UNIVERSITÉ FRANÇOIS RABELAIS TOURS

École Doctorale MIPTIS

Équipe BdTln – LI Tours

THÈSE présenté par :

Jakub WASZCZUK

soutenue le : 26 Juin 2017

pour obtenir le grade de : Docteur de l'Université François Rabelais Tours

Discipline: Informatique

Leveraging MWEs in practical TAG parsing: towards the best of the two worlds

THÈSE DIRIGÉE PAR :

SAVARY Agata	Maître de Conférences, HDR, Université François Rabelais Tours

RAPPORTEURS :

KALLMEYER Laura	Professeur, Heinrich-Heine-Universität Düsseldorf, Allemagne
WINTNER Shuly	Professeur, University of Haifa, Israël

JURY :

ANTOINE Jean-Yves	Professeur, Université François Rabelais Tours
DUCHIER Denys	Professeur, Université d'Orléans
KALLMEYER Laura	Professeur, Heinrich-Heine-Universität Düsseldorf, Allemagne
NASR Alexis	Professeur, Université d'Aix-Marseille
PARMENTIER Yannick	Maître de Conférences, Université d'Orléans (co-encadrant)
SAVARY Agata	Maître de Conférences, HDR, Université François Rabelais Tours
VILLEMONTE DE LA CLERGERIE Éric	Chargé de Recherche, INRIA Paris
WINTNER Shuly	Professeur, University of Haifa, Israël

Acknowledgments

First of all, I would like to thank Agata Savary and Yannick Parmentier, my PhD advisors, for their enthusiasm, their availability, and their expertise, for helping me out with both scientific and everyday problems, for guiding me towards realistic goals, for telling me when to stop, for all the work they put in this thesis themselves and, last but not least, for having made this experience very enjoyable from start to finish.

I would like to also express my gratitude to the jury members for having accepted to participate in the defence of this thesis.

I would like to thank all the members of the BdTln team and all the people associated with the computer science department in Blois for the very friendly atmosphere I experienced during my stay there. In particular, I would like to thank Verónika Peralta, Béatrice Bouchou, Patrick Marcel, Jean-Yves Antoine, Nathalie Friburger, and Agata for giving me the opportunity to teach by their side, which allowed me to occasionally catch a breath from my PhD-related activities. Many thanks to François Laurand for always being ready to help me with any technical issues I might have. I am also grateful to Jean-Yves for his help in guiding this thesis towards a happy end, and to Denis Maurel for his constant interest in our work.

I would like to thank all the members of the PARSEME action, from whom I learned a great deal about both MWEs and parsing, which significantly facilitated my work on this thesis. A greater part of this manuscript is either inspired by or based on their work. My thanks go to Simon Petitjean and Timm Lichte for hosting me during my stay in Düsseldorf, for their inspiring work on XMG, and for their help in making ParTAGe – the tool developed as a part of this thesis – more usable. Many thanks to Chérifa Ben Khelil for taking her time to stress test ParTAGe with her grammar.

I am also grateful to Agnieszka Mykowiecka and Małgorzata Marciniak, who made me discover the domain of natural language processing and showed

how fascinating this subject can be. My thanks go also to the other members of the NLP team in Warsaw – Adam Przepiórkowski, Agnieszka Patejuk, Marcin Woliński, Elżbieta Hajnicz, Maciej Ogrodniczuk, among many others – to whom I owe, to a great extent, my decision of starting a PhD.

Special thanks go to all the students who occupied office 323 during the course of my stay in Blois – particularly to Anaïs Lefeuvre-Halftermeyer, Chedlia Chakroun, Julien Aligon, Mahfoud Djedaini, Adnan El Moussawi, Caroline Pasquer, and El Arby Sidi Aly. I am glad that I had the possibility of sharing the unbelievable experience of a PhD thesis with others and in such a friendly environment. My thanks go also to Aneeksha Brigemohun and Christine Farmer for helping me to keep up my English when my French was not even good enough to properly buy a baguette.

Finally, I am grateful to my family for their love and support. My thanks go in particular to Zuzia for her strength and patience, and for accepting to turn our life upside down so that I can have fun with parsing MWEs. Many thanks to my kids, Ola and Asia, for their humour, for the many sleepless nights we spent together, and for making this PhD an even more incredible experience. I am also grateful to my parents, Jadzia and Jarek, for all the help they provided us with both before and during this thesis.

Contents

Chapter 1

Introduction

The class of multiword expressions (MWEs) contains irregular expressions with unpredictable properties, crossing boundaries of different linguistic representations. Some MWEs are irregular already at the level of morphology, e.g. (FR) *grands-mères* 'grandmothers', whose components do not agree in gender. Others have irregular structure, e.g. (EN) *by and large* 'in general', a surprising coordination of a preposition *by* and an adjective *large* which yields an adverbial modifier. At the semantic level, MWEs show a varying degree of non-compositionality, e.g., *to pull strings* is semantically opaque but can be understood compositionally if the components themselves are interpreted in an idiomatic way (*to pull* as 'to use', and *strings* as 'one's influence'). In fact, the spectrum of the irregular, MWE-related properties is so wide that it is hard to pinpoint a unified specification of what MWEs actually are.

However, one thing is clear – MWEs are exceptional and, as such, they are not always easy to deal with. Even though, globally, they can account for more than 40% of lexical items in a natural language, individually MWEs are relatively scarce (Savary, 2014). This makes it difficult to: (i) empirically investigate their irregular properties, (ii) compile MWE-dedicated lexical resources, (iii) investigate their distributional characteristics (important for statistical processing), among others.

A practical NLP application where the influence of MWEs appears as evident is machine translation. Because of their non-compositional semantics, MWEs cannot be translated literally, word-by-word, but only as a whole. For instance, (FR) *à la va-vite* should be translated to 'hastily, carelessly' rather than to (lit.) *to the go-quickly*, while (PL) *wystawić [kogoś] do wiatru* means to 'fool somebody' rather than to (lit.) *put [somebody] to wind*. Before

MWEs can be translated, though, they have to be identified, and MWEs are often hard to detect due to their fine-grained, non-trivial idiosyncratic requirements and, often, discontinuous nature. Moreover, MWEs can be ambiguous with either literal readings or accidental co-occurrences of their component words (Nasr et al., 2015; Constant et al., 2017). For instance, (FR) *Est-ce ainsi que les hommes vivent?* 'is it so that people live', might, at first sight, seem to contain the MWE (FR) *ainsi que* (lit. *so that*) 'as well as', but the co-occurrence of the words *ainsi* and *que* in this sentence is accidental and the syntactic relation between them is different than in their true MWE occurrences.

Interpretation-oriented NLP tasks, such as semantic calculus or translation, call for MWE-dedicated procedures. In this work, we focus on syntactic parsing, which often underlies such tasks. MWEs exhibit properties of both words and syntactic expressions, hence it is not clear what is an appropriate model to handle them in parsing, nor how they should be represented in syntactic treebanks (Rosén et al., 2015). The crucial issue is at which point the MWE identification should take place: before (Nivre and Nilsson, 2004; Arun and Keller, 2005; Constant et al., 2013a; Korkontzelos and Manandhar, 2010), after (Constant et al., 2012; Nagy T. and Vincze, 2014), or during syntactic parsing (Green et al., 2013; Candito and Constant, 2014; Le Roux et al., 2014; Nasr et al., 2015; Constant and Nivre, 2016). The last, joint approach deals best with the circular dependencies between MWEs and syntax, but it is also the most challenging one to implement, because it requires extending the existing methods with MWE-dedicated mechanisms.

Statistical parsing approaches typically focus on robustness more than on precision. High precision, indispensable to account for the idiosyncratic properties of MWEs, is achievable in symbolic parsing approaches. Within this context, MWEs challenge the traditional grammar/lexicon distinction, adopted to different extent in different symbolic frameworks. When syntactic rules are distinguished from lexical entries, as e.g. in Lexical Functional Grammar, LFG (Dalrymple, 2006), it is unclear to which of these two levels MWEs should belong. In some frameworks, e.g. in Combinatory Categorial Grammar, CCG (Lewis and Steedman, 2014), MWEs can be only modeled compositionally, which is counter-intuitive from the lexicographic point of view. MWEs are modeled compositionally also in Head-Driven Phrase Structure Grammar, HPSG (Sheinfux et al., 2015; Bargmann, 2015). Finally, it is non-trivial to model the idiosyncratic properties of MWEs and to account for their various restrictive and defective properties, syntactic

configurations, subcategorization requirements, etc., even within the context of formalisms which provide a first-class support for MWEs, such as Tree Adjoining Grammar, TAG (Joshi and Schabes, 1997). Existing solutions to these issues are typically based on grammar engineering methods – macros (LFG), meta-grammars (TAG), type-hierarchies (HPSG), etc. – and often rely on MWE-aware lexicons, difficult to compile due to the scarcity of MWEs.

In the domain of symbolic parsing, a growing interest is dedicated to parsing strategies which allow to compute the most plausible parse tree(s) without having to generate the space of all the grammar-compliant solutions in advance. Such strategies enable efficient parsing within the context of large grammars and/or complex symbolic formalisms (Angelov and Ljunglöf, 2014). They were also shown to finely combine with probabilistic supertagging methods, which can be used to pre-score the individual solutions and, hence, guide the parser to quickly find the most probable one(s) (Lewis and Steedman, 2014). Bangalore and Joshi (1999) claim that, once supertagging is correctly performed, the remaining step of determining the corresponding syntactic structure is trivial (Nasr and Rambow, 2010). Using a parsing strategy in combination with supertagging allows then to backtrack from eventual mistakes made by the supertagger. However, the existing symbolic parsing strategies rarely pay attention to MWEs, even though the latter are ubiquitous and potential MWE occurrences can help the parser to make better disambiguation decisions (Wehrli, 2014).

One of the main goals of this thesis is to remedy this deficiency by investigating the question of how to express the idea of promoting MWEs (Wehrli, 2014) in terms of a parsing strategy (Lewis and Steedman, 2014), as well as to verify the impact of such a strategy, applied to several types of MWEs, on the accuracy and speed of symbolic TAG parsing. Having the goal of efficient, MWE-aware TAG parsing in mind, we design an architecture which allows to benefit from standard grammar compression techniques, thus making it less sensitive to MWE-induced changes in grammar size. It also allows us to investigate the usefulness of promoting MWEs in a context closer to a real-world situation where, clearly, different parsing optimizations should ideally combine to provide an optimal solution.

The remainder of this document is structured as follows. In Ch. 2, we summarize the main approaches to representing structural relations – namely, dependencies and constituencies. In Ch. 3, we look more closely at MWEs and at the types of their defining characteristics. In Ch. 4, we investigate the relations between MWEs and syntax by looking at how MWEs are represented

in the existing MWE-aware syntactic treebanks. In Ch. 5, we look at the particular issues and challenges MWEs introduce in the domain of syntactic parsing, as well as at how MWEs are accounted for in the existing parsing systems, both purely statistical and symbolic ones.

Ch. 6 includes a description of TAG, the formalism we chose as the basis for our work due to its first-class support for MWEs, and a range of TAG-related definitions which we rely on in the following chapters. A description of our TAG parsing architecture, followed by a formalization of the A*-based TAG parsing strategy which enables promoting MWEs, can be found in Ch. 7. An experimental evaluation of the benefits and drawbacks of the MWE-promoting strategy is described in Ch. 8. All the experiments are performed with ParTAGe,[1] a parser for TAGs which extends the above-mentioned architecture with facilities enabling real-world applications, such as grammar compression and feature structures (detailed in Sec. 7.5). Finally, we describe our ideas for future work in Ch. 9 and conclude in Ch. 10.

[1] Available at `https://github.com/kawu/partage` under an open license.

Chapter 2

Syntactic structure

2.1 Constituents

Words in natural languages can be characterized by grammatical classes (i.e. as nouns, verbs, adjectives, etc.) and one can empirically observe that words belonging to the same class will often appear in similar contexts. In English, for instance, an adjective often precedes a noun, but not a verb; an adverb can be often found in the company of a verb, but it is not so likely to follow a preposition.

The idea of constituency emerges from the observation that the above behavior can be generalized from words to word groups, also called *phrases* or *constituents*. They also can be classified – roughly, on the basis of the grammatical classes of the main words they contain – as noun phrases, verbal phrases, adjectival phrases, etc., and they also exhibit preferences regarding the company they tend to keep. Moreover, phrases are to a lesser or a greater extent inseparable. While a noun phrase can very well precede a verb (cf. Ex 2.1), it is not true for each of its components separately (cf. Ex. 2.2), nor can one split a noun phrase and put half of it before the verb and the other half behind (cf. Ex. 2.3, see (Jurafsky and Martin, 2008, ch. 12, p. 421) for other examples).

(2.1) [All the particularly black cats]$_{NP}$ [bring]$_{V}$ misfortune.

(2.2) *Particularly bring misfortune.

(2.3) *All the particularly bring black cats misfortune.

In free-word-order languages, e.g. Polish, such constraints seem to play a less important role. For instance, it is possible to separate, by a verb, an adjective from the noun it modifies (cf. Ex. 2.4), since their relation is morphologically marked (through case and number, in this particular case). Nevertheless, hard word-order constraints also occur – the preposition must obligatorily precede its noun complement, which renders the sentence in Ex. 2.6 ungrammatical.

(2.4) Zgubę czarne przynoszą koty (PL)
Affliction.SG.ACC black.PL.NOM bring cats.PL.NOM

(2.5) Myślę [o niebieskich migdałach]$_{\text{PP}}$ (PL)
Think.SG.1 [about blue almonds]$_{\text{PP}}$
'I think about nothing in particular, I idle'

(2.6) *Myślę niebieskich migdałach o (PL)
Think.SG.1 blue almonds about

This empirical behavior of word groups leads to the idea of representing syntactic relations in natural languages not as dependencies between words, as is the case in dependency grammar (see Sec. 2.2), but rather as dependencies between both words and phrases. Phrases can compose together to build up complex, recursively embedded structures, which calls for an appropriate, sufficiently expressive formal representation. A formal representation typically adopted in constituency grammars is a tree. Fig. 2.1 shows an example constituency-based analysis of the sentence from Ex. 2.1, in which phrasal nodes constitute internal nodes of the tree and words of the sentence are placed in its leaves.

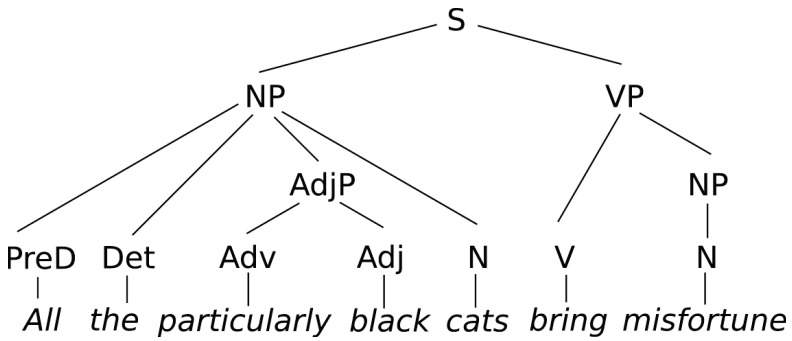


Figure 2.1: A constituency-based analysis of the sentence from Ex. 2.1.

An important property of syntactic structures – both dependency-based and constituency-based – is the so-called *projectivity*.

Definition 1 (Projectivity)**.** *Let $w_1, w_2, \ldots, w_n$ be a sequence of words, where n is its length, and $<$ be a binary transitive relation representing their ordering in the sentence. A constituent structure is said to be* projective *if it satisfies the property that, for any two words w_i and w_j which are in the* yield[1] *of a given phrasal node, every word w_k such that $w_i < w_k < w_j$ also belongs to its yield. Fig. 2.1 provides an example of a projective constituent tree.*

It is sometimes argued that constituency trees are inevitably projective.[2] It is clearly true for trees which can be expressed in context-free grammars (CFGs), one of the first constituency-based formalisms used for syntactic analysis. CFG has been nevertheless shown to be too weak to account for certain syntactic phenomena in natural languages, precisely because it does not allow to construct non-projective structures which are required to model certain sentences, e.g. the one in Ex. 2.4, where any reasonable constituency-based analysis should group *czarne* 'black' and *koty* 'cats' under a single constituent NP, which, at the same time, should not contain the verb *przynoszą* 'bring' in its yield (otherwise it would have to be a VP because it is the noun *koty* which is bound by the verb *przynoszą*).[3]

One possible solution to this issue is to use formal representations which exceed the modeling power of CFGs, e.g., which relax the assumption that a syntactic structure must be projective, and which therefore allow a certain degree of crossing dependencies. Fig. 2.2 (a) shows an example of a non-projective tree which could be assigned to the sentence from Ex. 2.4. It is also possible to relax the assumption that the syntactic structure is a tree whatsoever, in order to account for the so-called gapping constructions, an example of which can be found in Fig. 2.2 (b).

Another solution, used in formalisms like LFG or HPSG – which adhere to the projective, tree-based representations of constituents – consists in representing non-projective relations between words by using a dual representation

[1] A word is in the yield of a phrasal node if it is reachable from this node via a transitive closure of the parent-child dominance relation. For instance, in Fig. 2.1, the word *black* is in the yield of the phrasal node S, but not in the yield of the node VP.

[2] Kahane (2012, p. 269) claims this indirectly by writing "*the boy* is not a constituent [in *the little boy*] because these two lexical items are not adjacent."

[3] See (Kallmeyer, 2010, p. 19) for more general arguments relying on the properties of the string languages which can be generated with CFGs.

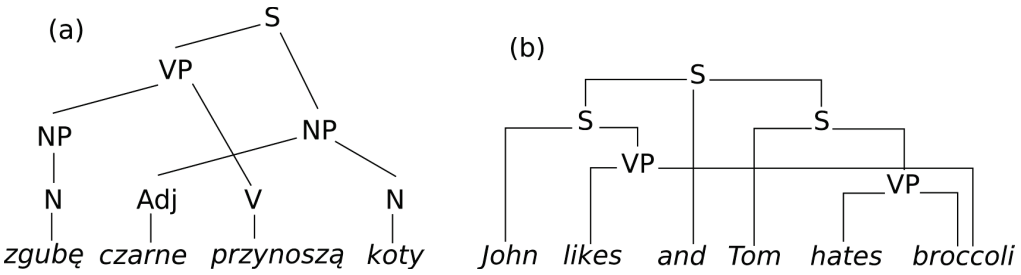


Figure 2.2: (a) A constituent analysis with non-projective constituents. (b) An example of a gapping construction where *broccoli* is analysed as an element of two VPs, one over *likes*, the other over *hates* (example (b) borrowed from (Kallmeyer, 2010, p. 5))

based on the so-called *feature structures*. More information on such structures can be found in Sec. 2.3.

Finally, constituent structures inform about groupings of words, but they do not, by default, inform about head-dependent relations between them, which are nevertheless closer to semantic representations than constituents. For instance, both the verb *likes* and the proper noun *John* are in the yield of the left-most sentential node in Fig. 2.2 (b). In the standard approach to semantic modeling, *likes* is considered a predicate and *John* its argument, but this relation is not present in the constituent structure. The simplest way to add this kind of information is to use the so-called *head* annotations, i.e., mark selected edges of the graph as heads. Fig. 2.3 (a) presents a headed version of the constituent tree shown in Fig. 2.1, while Fig. 2.3 (b) presents the corresponding graph of head-dependent relations. This latter structure is, in fact, a dependency graph, an example of the alternative approach to syntactic modeling.

2.2 Dependencies

While the constituency-based approach dominated during the second half of the 20th century, in recent years the trend appears to be reversed. The dependency-based approach is now appreciated as an appropriate way to represent syntactic structures (Kahane, 2012) and it has been also shown to provide an effective backbone for syntactic parsing models. Nivre (2005) postulates that the latter may be related to potential usefulness of bilexical

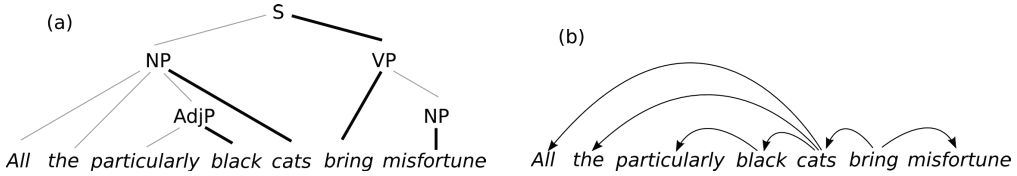


Figure 2.3: (a) A head-annotated version of the tree from Fig. 2.1, with the head annotations highlighted in bold; phrasal nodes directly dominating words were omitted for clarity. (b) The corresponding graph of head-dependent relations between words (arcs lead from heads to their dependents), which can be obtained using the procedure described e.g. in Kahane (2012).

relations[4] in syntactic disambiguation, as well as to a more restricted search space when processing a given sentence. Indeed, from a formal point of view, the same dependency tree can be obtained from many different (headed) constituency trees (Kahane, 2012). A constituency parser needs to learn to distinguish all these trees, while from a dependency-oriented perspective they all represent the same set of syntactic relations and the work performed to differentiate them appears futile. Other reasons may be that the constitutency-based approach was popularized together with context-free grammars, known to be too weak for syntactic modeling, and that vanilla constituent structures do not represent head-dependent relations, which are closer to semantic representations than constituent structures, and which are emphasized in the dependency grammar approach (Kahane, 2012, p. 260).

The common core of assumptions underlying the various approaches to dependency grammar is that a syntactic structure consists of *lexical nodes* linked via asymmetric binary relations called *dependencies* (Nivre, 2005). The goal of dependency analysis consists therefore in determining the connections between lexical units, their types, and their directions. Formally, the resulting structure is a *labeled directed graph*, which is typically required to be *connected* (so that the analysis covers the entire sentence) and, especially in practical applications like parsing, it is assumed to be a *tree* rather than a general graph (which cuts down the number of possible structures per sentence).[5] We have already seen an example of a dependency tree in Fig. 2.3 (b). Fig. 2.4 shows the same tree enriched with labels specifying the types of the individual

[4]Relations occurring between words associated with each other through a direct syntactic relationship, e.g., of one word being a subject or an object of the other one.

[5]Another property mentioned in (Nivre, 2005) is *acyclicity*.

head-dependent relations.

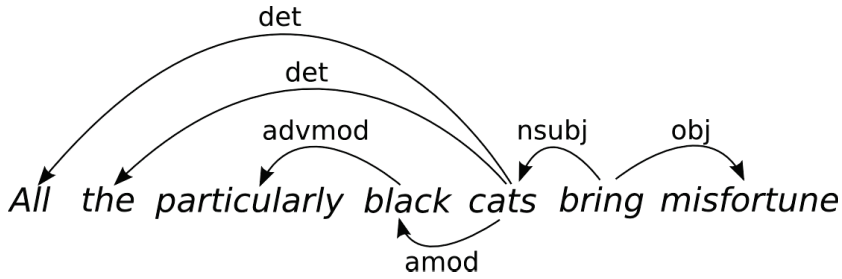


Figure 2.4: Tree from Fig. 2.3 (b) enriched with labels consistent with the Universal Dependencies tagset: `nsubj` for nominal subject, `obj` for object, `amod` for adjectival modifier, etc.

Directions specify the head-dependent relations between words and are not always easy to ascertain. The choice is particularly problematic in *exocentric* constructions, where syntactic properties emerge from either none or more than one of its lexical elements.[6] For instance, there is no consensus on how to model grammatical function words (articles, complementizers, auxiliary verbs) or prepositional phrases (Nivre, 2005, p. 6).[7] Coordination, even though *endocentric*[8], poses a similar problem, because neither the choice of the conjunction (which cannot syntactically replace the entire construction) nor one of its complements (both coordinated elements have the same status) is linguistically satisfactory (Nivre, 2005, p. 11). Fig. 2.5 shows an example of two possible coordination analyses in the dependency framework.

There are many extensions of dependency grammars. Some approaches allow underspecification of directions, which may help to better represent exocentric structures. One can relax the by-default assumption that dependency structure is a tree, and use dependency graphs to allow a given lexical unit to be governed by several heads, as exemplified by the plain and the dashed arcs in Fig. 2.5 (b). Other extensions propose to distinguish different types of relations – semantic from syntactic and morphological, endocentric from exocentric – or the use of stratified structures, where several dependency

[6]Note that this issue emerges not just in dependency grammars, but, in general, in formalisms which rely on the notion of headedness.

[7]Although recent efforts – e.g. the collaboration around the Universal Dependencies project (Nivre et al., 2016) – strive to provide a unified answer to these questions.

[8]As opposed to exocentric constructions, syntactic properties of an endocentric contraction emerge from *one* of its lexical components, called its *head*.

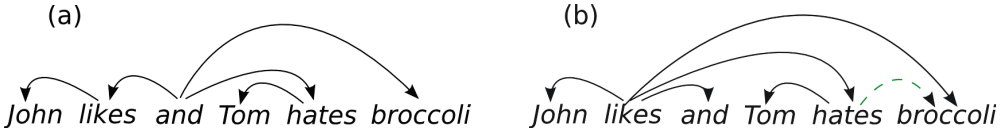


Figure 2.5: (a) An analysis which assumes that conjunction *and* is the head of the coordinated verbs. (b) The first of the coordinated verbs, *likes*, constitutes the root of the entire construction, while both the conjunction *and* and the second verb *hates* are its direct dependents. The second analysis is consistent with Universal Dependencies. Moreover, (b) contains an additional, dashed arc highlighted in green, which represents the verb-object relation between *hates* and *broccoli*, but whose addition makes the structure a general graph rather than a tree.

layers are constructed over a given sentence.[9] An example of the former is the theory of Tesnière (1959), where *connections*[10] play the role of dependencies, *transfer* relations connect elements of exocentric constructions (often of a more morphological nature, e.g. a case marking preposition with its noun complement), and *junction* relations link coordinated elements (Nivre, 2005, p. 7).

The stratified methodology has been used in designing and constructing the Prague Dependency Treebank (PDT) (Bejček et al., 2012). It follows the theory of Functional Generative Description (FGD) (Sgall and Hajicová, 1986), where the *analytical* layer is used to represent more surface-oriented syntactic relations and the *tectogrammatical* layer can be considered as a deep syntactic/shallow semantic representation (Nivre, 2005, p. 7). In this theory, for instance, an exocentric prepositional phrase is headed by its preposition in the analytical layer, but this very same preposition is not present in the tectogrammatical layer, where the entire construction is represented by its semantic head (noun) directly dependent on the verb (Nivre, 2005, p. 11).

Extensions allow to solve many of the issues emerging in the basic dependency grammar approach. As we will see, some of them turn out useful within the context of MWEs. However, while most dependency theories are expressive, most parsing frameworks adopt only the basic set of assumptions

[9]Some stratified dependency theories are even more complex, e.g. in the *bubble* dependency grammar lexical nodes form hierarchical structures and dependencies can connect more than two nodes at the same time (Kahane, 2012).

[10]Kahane (2012) designates by *connections* undirected binary relations.

and do not handle multi-stratal representations, general graph structures, or non-projective trees.[11]

2.2.1 Dependencies vs. constituents

The apparent difference between dependency- and constituency-based approaches is the lack of phrasal nodes in the former (Nivre, 2005). On the other hand, lexical relations inherently present in dependency structures are specified in constituency structures only if the latter contain appropriate *head* annotations (Kahane, 2012). Moreover, while phrasal nodes are typically annotated with phrase type annotations (VP for verbal phrase, NP for noun phrase, S for sentence etc.), arcs in dependency structures are typically annotated with grammatical functions (SUBJ for subject, OBJ for object, MOD for adjunct, etc.) or thematic roles as in PDT. Thus, these two approaches to syntactic modeling differ not only in the hierarchical structure they use to represent syntactic information, but also in the domain of the corresponding type annotations, which play different roles in both approaches.

According to Kahane (2012), the main reasons to prefer dependency grammars over constituency grammars include the following: (i) dependency structures do not imply implicit ordering constraints, thus they may be better adapted for free-word-order languages,[12] and (ii) constituency structure alone (i.e., without important extensions) does not allow to represent non-projective relations. Apparently, Kahane (2012) refers to structures stemming from the Chomskian family of constituency-based formalisms. Non-projective structures can be easily represented by allowing the yields of the individual phrases to be interleaved with each other, as exemplified in Fig. 2.2. This approach is adopted in the family of TAG-related formalisms, e.g. in linear context-free rewriting systems (Kallmeyer, 2010, p. 4, fig. 1.2). It is nevertheless true that formalisms like LFG or HPSG adopt a dual syntactic representation with a projective, constituency-based backbone,[13] additionally enriched with feature structures, which allow to represent potentially non-projective, functional dependencies between words (subject, object, etc.).

[11] Projectivity for dependency structures follows Def. 1 under a slightly modified definition of *yield* which includes all the words reachable from a given lexical node by means of a reflexive transitive closure of the head-dependent relation.

[12] For example, FGD assumes that tectogrammatical structures are projective, while analytical structures not necessarily (Nivre, 2005).

[13] More precisely, a context-free-grammar backbone.

It seems therefore that the crucial question in the dependency vs. constituency debate is simply whether dependency structures are not only *necessary* but also *sufficient* to model syntactic structures of natural languages (Nivre, 2005, p. 8). According to Rambow (2010), constituencies and dependencies can be both seen as syntactic content-independent approaches to syntactic modeling, in the sense that the same syntactic content (transitive verbs, multiword expressions, control constructions, etc.) can be representated using either of these representation types. Still, phrasal nodes – even if not necessary – could turn out simply *too convenient* for syntactic modeling to be abandoned. And while the dependency approach gains in popularity both in linguistics and in parsing, the constituency approach is still more widespread as far as symbolic grammars and symbolic parsing are concerned, which may confirm the above intuition. Besides, it is sometimes claimed that constituent structures allow to support compositional semantics more naturally than dependency structures (Crabbé, 2014).

2.3 Composite representations (LFG)

Syntactic approaches which rely on more elaborate linguistic theories – and thus adopt more assumptions, but also provide richer descriptive devices – also exist. In this section we will focus on one of them, called *lexical functional grammar*, or LFG (Dalrymple, 2006), which combines the properties of the constituency and dependency approaches to syntactic modeling. Note, however, that similar arguments apply to other formalisms: HPSG (Sheinfux et al., 2015), TAG (Joshi and Schabes, 1997), CCG (Lewis and Steedman, 2014), etc.

Constituent structure. The syntactic analysis of a given utterance is represented in LFG by a heterogeneous construction consisting of a *constituent structure* and a *functional structure*. A constituent structure (c-structure) is a regular constituency tree, a structure representing groupings of words into phrases and of phrases into more complex phrases. Constituent trees in LFG are required to be *projective*. As we have seen before, for certain expressions non-projective trees are arguably more appropriate, which can be experienced especially within the context of free-word-order languages. However, c-structures in LFG are sometimes seen as performing a secondary role – they are only needed to build a scaffolding over which the functional structures can be conveniently defined. Within the scope of functional structures, much

more expressive than constituent trees, the constraint of projectivity does not apply.

$$eats \rightarrow \text{V}: \begin{bmatrix} \text{LEX} & eat \\ \text{SUBJ} & \begin{bmatrix} \text{NUM} & sg \\ \text{PERS} & 3 \end{bmatrix} \\ \text{OBJ} & \begin{bmatrix} \text{CASE} & acc \end{bmatrix} \end{bmatrix} \qquad Tom \rightarrow \text{N}: \begin{bmatrix} \text{LEX} & Tom \\ \text{NUM} & sg \\ \text{PERS} & 3 \end{bmatrix}$$

$$mice \rightarrow \text{N}: \begin{bmatrix} \text{LEX} & mouse \\ \text{NUM} & pl \\ \text{PERS} & 3 \end{bmatrix} \qquad a \rightarrow \text{D}: \begin{bmatrix} \text{LEX} & a \\ \text{NUM} & sg \end{bmatrix}$$

Figure 2.6: Potential part-of-speech and AVM representations of selected analyses of several English words. Each AVM consists of a list of feature-value pairs, where the value is either an atomic value (e.g. *sg* or *acc*) or a nested AVM. The verb's lexical analysis specifies grammatical requirements over its arguments – subject and object. Verification of these requirements should be handled by grammar rules. Note that, in LFG, lexical entries are described by functional descriptions rather than feature structures. The latter are only used to represent their analyses.

Functional structure. From the technical point of view, functional structures (f-structures) are represented with the so-called *feature structures* (FSs) and they constitute the source of the high expressiveness of LFG grammars. FSs in LFG are defined in a similar way as in general unification grammars (Francez and Wintner, 2012). In particular, a feature structure can be seen as a directed (potentially cyclic) graph in which two types of nodes can be distinguished:

- Frontier nodes with atomic *feature values*,
- Empty internal nodes, from which edges labeled by *features* may lead to other internal or frontier nodes[14].

[14] An internal node with no outgoing edges can be seen as a frontier node, but – because it does not contain any feature value – it can unify both with internal and with frontier, value-saturated nodes.

$$\begin{array}{lcccc} \text{S} & \rightarrow & \text{N} & & \text{VP} \\ & & (\uparrow \text{SUBJ}) = \downarrow & & \uparrow = \downarrow \\ \\ \text{VP} & \rightarrow & \text{V} & & \text{N} \\ & & \uparrow = \downarrow & & (\uparrow \text{OBJ}) = \downarrow \end{array}$$

Figure 2.7: A simple LFG grammar consisting of two CFG-like rules.

Feature structures are often represented graphically as the so-called *attribute-value matrices* (AVMs), well-known within the domain of linguistics, in which re-entrancies are represented with coreference identifiers. Figure 2.6 shows how feature structures could be used to represent grammatical interpretations of several English wordforms.

The correspondence between c-structures and f-structures is ensured by CFG-like grammar rules, as the ones shown in Fig. 2.7, which specify that (the upper rule) a sentence can be composed of a noun and a verb phrase, and that (the bottom rule) the latter can be composed of a verb and a noun. The annotations placed below the rules' right-hand side elements specify that in this particular word-order configuration the first noun provides a subject for the verb, while the second noun provides its object. Technically, $\uparrow$ designates the f-structure of the resulting construction (constituent), while $\downarrow$ refers to the f-structure of the corresponding subconstituent (of the verb phrase if it is placed under VP, of the noun if placed under N, etc.). Thus the $\uparrow = \downarrow$ equation used in both rules specifies that the same f-structure is assigned to the verb, to the verb phrase, and to the entire sentence – in other words, that the verb is the *head* of this construction. The annotation $(\uparrow \text{SUBJ}) = \downarrow$ attached to the noun element of the upper rule, on the other hand, specifies that this noun (here: *Tom*) is the subject of the verb. Similarly, the relation between the verb and its direct object is specified in the bottom rule.

Application of these two rules to the sentence *Tom eats mice*, assuming the lexical interpretations taken from the above lexicon, leads to the composite syntactic representation illustrated in Fig. 2.8. To emphasize the relationship between f-structures and dependency structures, grammatical functions are represented as arcs, while the morphosyntactic attributes are depicted using the AVM notation.

LFG grammars can thus produce composite constituency/dependency structures, and the latter, represented by FSs, can form potentially cyclic

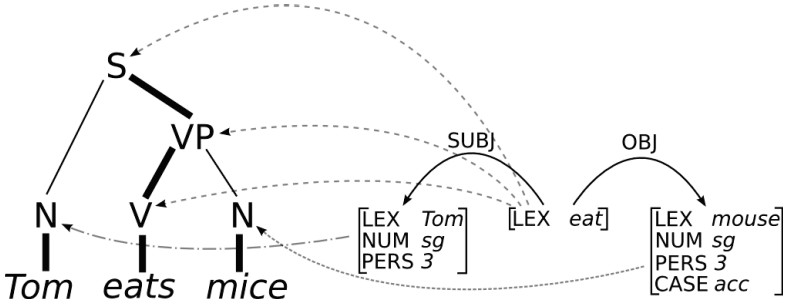


Figure 2.8: An LFG-based analysis of the sentence *Tom eats mice* based on the lexicon from Fig. 2.6 and the grammar from Fig. 2.7. The tree on the left represents the c-structure resulting from the application of the grammar rules. The dependency-like tree on the right depicts the resulting f-structure, constructed according to the f-structure-related annotations placed below the rules' body elements. The gray arrows between the two structures show to which constituent nodes the individual dependency nodes correspond. These relations also stem from the head-annotations, shown in bold.

graphs and are not restricted to projective relations, which attests the high expressive power of LFG and other related formalisms. Incidentally, this also shows that the role of dependencies in formal grammars and in syntactic parsing, even if somewhat implicit, has been significant at least since the beginnings of LFG in the late 1970s.

Chapter 3

Multiword expressions

This chapter is dedicated to a general description of multiword expressions and their various irregular properties. In Sec. 3.1, we first try to define what multiword expressions (MWEs) are. In Sec. 3.2, we describe the various idiosyncratic properties which motivate their special treatment. In Sec. 3.3, we propose an extended definition of MWEs and finally, in Sec. 3.4, we show an approach which satisfies the requirements of this definition and which consists in describing MWEs in the form of MWE-aware syntactic lexicons.

3.1 Basic definition

A question of what exactly a multi-word expression (MWE) is doesn't seem very hard to answer at first sight. Numerous examples of MWEs are known and certain types of expressions are widely acknowledged to be a part of this broad family. On the other hand, the family of MWEs is a large one – it contains idiomatic expressions, noun compounds, verb-particle constructions, light verb constructions, fixed expressions, etc. This heterogeneity does not make it easy to design a minimal and consistent set of requirements which could be used to unambiguously recognize MWEs among other, more regular objects of the language.

A seminal work on multi-word expressions by (Sag et al., 2002) defines them rather roughly as "idiosyncratic interpretations that cross word boundaries (or spaces)". While very short, this definition evokes three important points – MWEs are defined here as *interpretations* and not as linguistic objects per se. Furthermore, these interpretations are meant to be *idiosyncratic*, which is the

key property of MWEs – idiosyncratic expressions exhibit properties which are *inconsistent* with what would be the result of their regular, compositional treatment. The third point brings up the multi-word characteristic – MWEs need to span over multiple words, typically interleaved with spaces.

In this work, we start with a tentative definition similar to the one adopted by Mel'čuk (2012), where MWEs are lexical objects defined on top of lexemes.

Definition 2 (lexeme, following (Woliński, 2014))**.** *A* lexeme *is a model of a word which groups together its various* grammatical forms*. Moreover, it provides information about the morphological features of its individual forms.*

For example, *cat* and *cats* can be analysed as two different grammatical forms, with different morphological features (singular and plural number, respectively), of the single English lexeme *cat*.

Definition 3 (multiword expression)**.** *A* multiword expression *is a model of a linguistic expression which is formed by several (at least two) lexemes and which exhibits some idiosyncratic properties.*

The above definition does not specify which expressions should be considered as idiosyncratic and which should not – we will focus to this issue in Sec. 3.2. Let us only note here that, in the extreme case where no linguistic properties are deemed as regular, any given linguistic expression formed by several lexemes can be considered a MWE.

In contrast to (Mel'čuk, 2012), we do not assume that the individual components of a MWE are formed in a syntactically regular way. Such definition would leave out all the various syntactically irregular MWEs, whose components cannot be reasonably linked using standard dependency relations, such as *by and large*.

3.2 Different flavours of idiosyncrasy

The definition of MWEs (see Def. 3) relies on a somewhat blurred notion of *idiosyncrasy*, which assumes an underlying distinction between what is regular in the language and what is not. The goal of this section is to show several examples of idiosyncratic properties of MWEs which motivate their special status and treatment in linguistic modeling and NLP applications.

Following Lichte et al. (2017), we divide (most of the) idiosyncratic properties into two broad categories of *restrictive* and *defective* properties.

The former restrict the number of the possible surface realizations with respect to the corresponding literal interpretations, while the latter relax some of the normally occurring linguistic constraints, thus blocking literal interpretation in certain configurations. Defective properties in particular can lead to quite unexpected surface realizations and syntactic configurations, and we will thus call MWEs which exhibit such properties (syntactically, morphologically, etc.) **irregular**.

The first type of MWE-related idiosyncrasy is the **lexical idiomaticity** (Baldwin and Kim, 2010). It occurs when one of the components of a MWE does not function in the language model as a rightful lexical unit (lexeme) on its own, as is often the case with foreign expressions such as (EN) *ad hoc*.[1]

Definition 4 (multitoken word)**.** *Let e be an expression such that one of its components does not function in the underlying language model as a lexeme. Then, we call e a* multitoken word*.*

Multitoken words are not strictly speaking MWEs, since they do not satisfy Def. 3. They are, nevertheless, interesting from the NLP point of view and exhibit common properties with syntactically irregular MWEs such as *all of a sudden*.

The property which motivates the very definition of MWEs as sets of lexemes is the **restrictive lexical selection**, which imposes particular lexical realizations of certain syntactic arguments (which we will also call **lexically constrained**), e.g. (EN) *to pull someone's leg* requires the head verb *pull* with a direct object headed by *leg*: #to pull one's arm/member. Mel'čuk (2012) defines *free phrases* as expressions whose individual lexical components are chosen by the speaker in an unconstrained way – "each [component] is selected strictly for its meaning and in conformity with its linguistic properties but independently of the lexical identity of other components". MWEs are not free in this sense, hence the need to describe them as lexeme sets.

Morphological idiosyncrasy is a property of MWEs which do not behave as regular units at the level of morphology, e.g. yield only a limited set of surface forms by constraining certain grammatical categories to specific values, for example in the morphologically restrictive MWE (PL) *krokodyle łzy* (lit. *crocodile tears*) 'superficial sympathy' which does not occur in

[1]Each example is preceded by its language code in parentheses. The hash ('#') character signals the loss of the idiomatic reading due to a missing property, while the star ('*') means ungrammaticality.

a singular form. While this type of idiomaticity occurs in English, it is more diverse in languages with richer morphology in which the number of grammatical categories which can be potentially constrained is higher. The nominal compound (PL) *czerwony pająk* (lit. *red spider*) 'post-communist' is morphologically defective – it may occur either in the masculine inanimate or in the human masculine gender form (and, consequently, inflect differently within certain contexts), even though *pająk* 'spider' alone does not occur in the latter (unless used metaphorically) (Czerepowicka and Kosek, 2011).

The MWE (PL) *zjadłbym konia z kopytami* (lit. *I would eat a horse with its hooves*) 'I am very hungry' can only occur in conditional mood and is thus **morphosyntactically idiosyncratic** (more precisely, morphosyntactically restrictive): *#zjem konia z kopytami* 'I will eat a horse with its hooves'. MWEs often require additional agreement, e.g. (EN) *to cross one's fingers* imposes agreement in person, number and gender between the possessive pronoun and the subject: *#I cross his fingers*. Finally, the pragmatic MWE (PL) *Bóg (ci za to) zapłać* (lit. God_{NOM} *(*you_{SG} *for this)* $pay_{\text{IMPT.SG.2}}$) 'God bless you' is morphosyntactically defective, since *Bóg* occurs in nominative even though the expression has the imperative mood, which normally entails the vocative case of the subject.

Yet another flavor of idiosynrasy is the **syntactic idiosyncrasy**. MWEs are quite often syntactically restrictive: they can (i) constrain the choice of determiners or modifiers, e.g. (FR) avoir raison (lit. to have reason) 'to be right' allows neither a determiner nor a modifier of the nominal component: *#avoir (une) raison évidente* 'to have an obvious reason', (ii) constrain the set of the corresponding syntactic configurations, e.g. (EN) *to kick the bucket* 'to die' cannot be passivised while (FR) *les carottes sont cuites* (lit. the carrots are cooked) 'the situation is hopeless' only allows passive voice: *#on cuit les carottes* (lit. one cooks the carrots), (iii) exhibit linearization constraints, e.g. (EN) *drink and drive* requires the strict order of its coordinated verbs and violating this constraint leads to the loss of the idiomatic reading: *#drive and drink*, etc.

The class of syntactically defective expressions includes, e.g., (EN) *by and large* 'roughly', which is an irregular coordination of a preposition and an adjective, and (EN) *attorney.*N *general.*A, where the adjective follows the noun it modifies. Another interesting syntactic property is the defective subcategorization, i.e. imposing a subcategorization frame which the MWE headword does not admit outside MWEs, e.g. (PL) *dobrze mu z oczy patrzy* (lit. *well him looks from eyes*) 'he looks like a good person' prohibits a subject:

**uczciwość dobrze mu z oczy patrzy* (lit. *honesty well him looks from eyes*), while patrzy 'looks' as a standalone verb always requires one.

A prevalent property of MWEs is the **semantic non-compositionality**. The standard definition of semantic *compositionality* in linguistics states that "a compound expression is compositional if its meaning is a function of the meanings of its parts and of the syntactic rule by which they are combined" (Savary, 2014). Expressions such as *to kick the bucket*, whose meaning is completely opaque and is not related to the meaning of any of its components, are therefore regarded as non-compositional.

Not every occurrence of semantic idiosyncrasy manifests itself as non-compositionality. Certain idiomatic expressions, such as (EN) *to spill the beans* 'to reveal (a) secret(s)', are widely considered as semantically **decomposable**. Once the individual components of this expression are assigned the corresponding figurative meanings – *reveal* and *secret(s)*, respectively – the meaning of the entire phrase can be obtained compositionally. It doesn't change the fact that the required figurative meanings of *spill* and *beans* are not readily available without the full lexical context of this particular MWE.

Another, even more subtle type of semantic idiosyncrasy can be exemplified by a seemingly trivial (EN) *bus driver* 'a person driving a bus' but not e.g. 'a piece of software which enables computer to communicate with a bus'. While here again the meaning of the compound can be derived from the appropriate meanings of its components and thus is perfectly compositional, this particular compound could be theoretically analysed in several different ways using the compositional methodology. The alternative meanings should be probably blocked in a linguistically precise semantic model.

The above definition of semantic compositionality should be taken with a grain of salt. It relies on the assumption that simplex wordforms are equipped with sets of their potential meanings – it is notoriously the case that a wordform can be analysed as many different lexemes and that a lexeme has several possible semantic analyses. The choice of how the set of the potential meanings for a given simplex wordform should be constructed is, in fact, quite arbitrary. If we choose all its possible meanings, regardless of the context it occurs in, even the idiomatic expression *to kick the bucket* could be modeled compositionally, as presented by Bargmann (2015). However, such a treatment can lead to a significant increase in the number of potential semantic interpretations of the individual occurrences of simplex wordforms, most likely to an overgeneration of semantic interpretations of complex expressions and, finally, does not seem to model very well the mental process of a native

English speaker who, most likely, does not think about dying every time he utters or hears the word *kick*.

Pragmatic idiosyncrasy is a property exhibited by expressions related to specific situations or real-life contexts, outside of which they are not typically used. Pragmatically idiosyncratic expressions include greetings, e.g. (EN) *Good morning*, politeness formulas, e.g. (FR) *Veuillez recevoir, Monsieur, l'expression de mes sentiments distingués* (lit. please receive, sir, the expression of my sentiments distinguished), etc.

The last and possibly the weakest flavor of idiocynrasy mentioned in the definition of MWEs presented above is the **statistical** (also called distributional) **idiosyncrasy**. Certain word combinations tend to co-occur significantly more often than other possible formulations of the same message and/or more often than the assumption of statistical independence between their individual components would make us believe. In fact, most of the MWEs which exhibit any symptoms of the idiosyncrasies described above tend to be also statistically idiosyncratic, but the implication doesn't work the other way around. An example of an expression which can be classified as idiosyncratic only at the level of statistics is (EN) *immaculate performance* (Baldwin and Kim, 2010). There is nothing special about its morphology, syntax or semantics, yet its two component words occur together with a marked frequency. Expressions idiosyncratic at the statistical level are called **collocations**.

3.3 Extended definition

Def. 3 abstracts away from the details of the idiosyncratic properties of MWEs, even though such properties provide the very motivation to model MWEs as objects of a lexical nature. A lexeme tells us what are the morphological features of its individual grammatical forms – by analogy, is seems reasonable to expect that a MWE tells us what are the properties of its individual forms. Since MWEs are syntactic expressions, such forms can be expected to have a syntactic nature as well.

Definition 5 (multiword expression)**.** *Following Def. 3, a* multiword expression *is a model of a linguistic expression which is formed by several (at least two) lexemes and which exhibits some idiosyncratic properties. Moreover, a MWE groups together its various* syntactic forms *– i.e., syntactic structures*

built on top of the corresponding lexemes – and, for each of these forms, provides information about its morphological and syntactic properties.

The above definition does not specify the nature of the syntactic structures, nor does it determine the form in which information about the properties of the individual syntactic forms should be given. Concerning the structure, the underlying linguistic theory could rely on either dependencies or constituencies. Concerning the methodology of description, the individual syntactic forms and their properties could be explicitly listed one-by-one, they could be described in a higher-level generative meta-language, or they could be expressed in terms of constraints which have to be satisfied by the individual syntactic forms. In any case, any precise answer to these questions would clearly have to commit to a certain linguistic theory of morphology and syntax. The following section, Sec. 3.4, describes a possible answer to these questions, based on the notion of MWE-aware syntactic lexicons.

3.4 MWEs in syntactic lexicons

We now focus on two related (even if independently developed) formalisms which serve to describe MWE-aware syntactic lexicons: PDT-Vallex and Walenty. Their extensive comparison with respect to phraseology can be found in (Przepiórkowski et al., 2017).

PDT-Vallex is a valency dictionary accompanying the development of the Prague Dependency Treebank, PDT (Bejček et al., 2012). PDT and PDT-Vallex are strongly related – each occurrence of a predicate word (mainly verbs, but also nouns, adjectives, and a few adverbs) in the tectogrammatical (shallow semantic) layer of PDT is expected to be annotated with its "word meaning", represented by the corresponding *valency frame*. A valency frame essentially describes all the expected complements of a given word, each complement potentially enriched with the corresponding morphosyntactic or lexical constraints. The dependents of the individual complements can be further specified in a recursive way.

PDT-Vallex includes two classes (dependency labels) dedicated to phraseological complements. `CPHR` (Compound Phraseme) is used to represent dependent objects of light verbs, while `DPHR` (Dependent Phraseme) is used for verbal idioms in general. An example of two valency frames assigned to the verb (CZ) *uzavřít* 'to close' can be found in Fig. 3.1. The first one, in particular, represents the light verb (CZ) *uzavřít smlouva* 'to close a contract'.

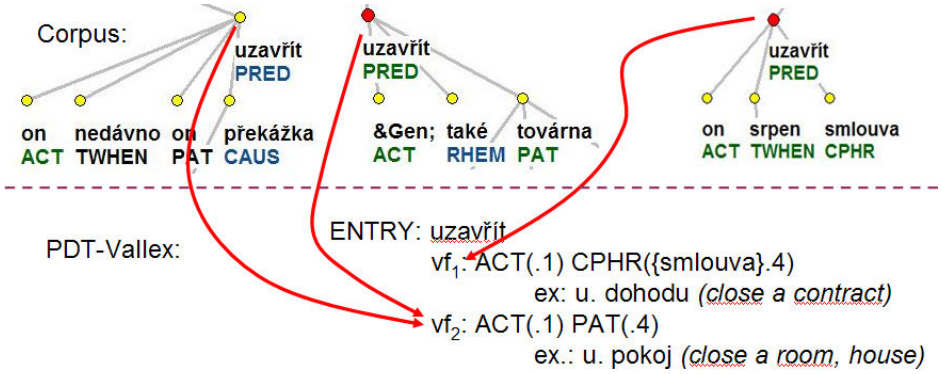


Figure 3.1: (Below) Two valency frames assigned to the Czech verb *uzavřít* 'to close'. The individual complements are marked with *functors* which designate the corresponding dependency labels. `ACT` stands for actor and is typically realized by the subject, `PAT` is the patient which is usually a direct object, and `CPHR` is the direct object of a light verb. The complements are also decorated with numbers 1 and 4 which designate the nominative and the accusative case, respectively. (Above) The links leading from the tectogrammatical nodes in PDT to the corresponding frames. Frame-specified complements can be determined on the basis of the dependency labels which should correspond to functors described in the valency entries. Figure borrowed from (Urešová, 2009, p. 7).

The PDT-Vallex lexicon is in fact related to both the tectogrammatical (TL) and the analytical layers (AL) of the PDT treebank. An occurrence of a given valency entry is a flat subtree on the TL level, a subtree whose individual children are annotated with the corresponding functors (actor, patient, etc.) coming from the lexicon. On the AL level, the same TL subtree can be represented by several different AL subtrees, of depth potentially greater than 1. AL subtrees can contain morphosyntactic and lexical constraints in their nodes, and can be obtained using various globally defined transformations (passivisation, for example) (Hajič and Urešová, 2003).

Walenty is a Polish large-scale valence dictionary of about 50000, 3700, 3000, and 1000 subcategorization frames (in its 2015 version) for Polish verbs, nouns, adjectives, and adverbs respectively. Its encoding formalism is probably more theory-neutral than PDT-Vallex, since the latter is strongly coupled with PDT and both PDT and PDT-Vallex are based on the Functional Generative Description framework (Sgall and Hajicová, 1986). Walenty

includes an elaborate phraseological component (Przepiórkowski et al., 2014). Thus, above 8,000 verbal frames contain lexicalized arguments of head verbs, i.e. they describe verbal MWEs. For instance the idiom highlighted in example (3.1) is described in Walenty as shown in Fig. 3.2. Each component separated by a '+' represents one required verbal complement with its lexical, morphological, syntactic, and (sometimes) semantic constraints. Here, the subject is compulsory and has a structural case (`subj{np(str)}`), which notably means that it normally occurs in nominative, but turns to genitive when the head verb is nominalized. The subject being a required argument in a verbal frame does not contradict the fact that it can regularly be omitted in Polish[2], as in (3.1).

(3.1) Nie umiem w tych sprawach **trzymać języka za zębami**.
Not know$_{\text{SG.1}}$ in these affairs hold$_{\text{INF}}$ tongue$_{\text{SG.GEN}}$ behind teeth.
(lit. *I cannot hold my tongue behind my teeth in such cases*) 'I cannot hold my tongue in such cases'

The second required argument is a direct object realized as a nominal phrase in structural case, i.e. normally in accusative but turning to genitive when the sentence is negated, as in (3.1). The lexicalized object's head has the lemma *język* 'tongue', should be in singular (`sg`) and does not admit modifiers (`natr`). The third complement is a prepositional nominal phrase (`prepnp`) headed by the preposition *za* 'behind' governing the instrumental case (`inst`) and a lexicalized non-modifiable (`natr`) noun with the lemma *ząb* 'tooth' in plural (`pl`).

```
trzymać: subj{np(str)}+
   obj{lex(np(str),sg,'język',natr)}+
   {lex(prepnp(za,inst),pl,'ząb',natr)}
```

Figure 3.2: Description of *trzymać język za zębami* 'hold one's tongue' in Walenty

The theoretical underpinnings of Walenty and PDT-Vallex are quite similar. Both provide a description language which allows to describe, roughly,

[2]This property is to be distinguished from impersonal verbs, which prohibit a subject, as in *dobrze mu z oczu patrzy* (lit. *looks him from eyes well*) 'he looks like a good person'. The same applies to Czech; however, elided arguments are restored in the tectogrammatical level of PDT, the level where PDT-Vallex frame occurrences are annotated.

connected dependency subtrees whose nodes can be potentially enriched with morphosyntactic and lexical constraints. Each valency frame encodes, therefore, a set of the corresponding, underspecified dependency tree fragments, even though Walenty descriptions encode also information about constituent structures. An occurrence of a fragment in a particular dependency structure can be then seen as a potential instance of the corresponding valency frame.[3]

While entries of either Walenty or PDT-Vallex can be seen as dependency subtree descriptions, their interpretation is not 100% formalized. Especially in case of Walenty, this can be seen as a by-product of the intentional theory-neutrality of the formalism. Walenty should be in principle usable with many different dependency-related formalisms which can, e.g., adopt different accounts of coordination (see also Sec. 2.2 for a short description of different issues related to dependency-based representations). A particular interpretation function, which relates valency entries with the corresponding dependency fragments, can be then defined within the context of a particular grammatical formalism. Patejuk (2015) showed that Walenty entries can be automatically transformed into the corresponding lexical entries of POLFIE, the LFG grammar for Polish. More precisely, the individual Walenty descriptions map into definitions and constraints over the corresponding LFG f-structures.

Przepiórkowski et al. (2017) describe certain limitations of the two formalisms with respect to the representation of MWEs. One of the challenging examples they provide is the idiom (PL) *biec swoim torem* 'run its course', where the dependent object *torem* 'course' must be modified by either an adjectival modifier or a genitive NP but, even though any number of adjectival modifiers is allowed, there can be at most one genitive NP in this idiom. Neither Walenty nor PDT-Vallex allows to express such a complex constraint. As a general solution to this and other issues, they propose to enrich the respective description formalisms with regular expression operators – Kleene star, Kleene plus, and optionality.

[3]One of the reasons why such instances are only potential is that two different valency frames can possibly match one and the same dependency fragment.

Chapter 4

MWEs in treebanks

According to Rosén et al. (2015), there is currently little agreement on how MWEs should be annotated in treebanks. This may be a striking observation, given that structures present in treebanks should provide a paragon of how they should be represented in the output of parsing systems. This shows that the question of "how to parse MWEs" is a doubly complicated one, since it is not only unclear how to handle MWEs in parsing, it is not even clear how to represent them in its results.

Constant et al. (2017) distinguish several main representations of MWEs in treebanks[1], some of which also emerge from the comparative study carried out by Rosén et al. (2015) who looked at the most commonly annotated types of MWEs – prepositional MWEs, verb-particle constructions and multiword named entities – and their representations in 18 different treebanks.[2]

In this work we propose our own typology, presented in Tab. 4.1, in which we divide MWE-dedicated representations into classes depending on their MWE-related characteristics: structural restrictions and the type of interface between MWEs and the morphosyntactic/syntactic layers. More specifically, we consider four different MWE-related characteristics:

- INTERNAL STRUCTURES take two possible values, depending on whether the given approach allows (*yes*) to represent the internal structure of MWEs or *no*t. If *yes*, we do not assume any specific constraints over

[1] Among which, the choice of not representing MWEs at all, which we omit here.

[2] Many of the examples shown in this section originate from the overview of MWE annotation in treebanks carried out by the working group 4 of the PARSEME action, see `http://clarino.uib.no/iness/page?page-id=MWEs_in_Parseme`.

Classes of MWE representation approaches	INTERNAL STRUCTURES	DISCONTINUOUS STRUCTURES	MWE/MORPHOSYNTAX INTERFACE	MWE/SYNTAX INTERFACE
Words with spaces	no	no	embedded	below
Chunks	yes	no	above	below
Subtree	yes	yes	above	embedded
Asynchronous	yes	yes	above	parallel
Overlay	yes	yes	above	above
Bidirectional	yes	yes	above	intertwined

Table 4.1: Main classes of MWE representation approaches in treebanks (rows) and their MWE-related characteristics (columns).

the type of the corresponding structure, even though it tends to be flat in some approaches and recursive in others.

- On top of that, the approach can either allow (*yes*) or *not* to represent DISCONTINUOUS STRUCTURES.

- We also consider two flavors of the MWE/MORPHOSYNTAX INTERFACE: MWEs can be either directly *embedded* in the morphosyntactic layer – essentially meaning that MWEs are represented as tokens, just as regular words – or constructed *above* this level.

- With respect to the MWE/SYNTAX INTERFACE, MWEs can be (i) placed *below* the layer of syntactic structures, which is equivalent to saying that MWEs form lexical nodes on top of which syntactic structures are constructed, (ii) *embedded* in syntactic trees, meaning that connections between MWE components are described as syntactic, (iii) put *above* syntactic structures, meaning that they refer to syntactic nodes but not the other way around, (iv) *intertwined* with syntactic structures, meaning that they form a separate layer of representation but references in either direction – from syntax to MWEs (and elements of their internal structure) and from MWEs to syntax – are allowed, and finally (v) *parallel* to syntactic structures, in the sense that their correspondence with syntactic structures is not guaranteed.

It is mainly on the basis of the two last characteristics – MWE/MORPHOSYNTAX INTERFACE and MWE/SYNTAX INTERFACE – that we define six different classes of MWE representations, presented as rows in Tab. 4.1. The classical *words-with-spaces* approach relies on the token-based representation of MWEs. Since individual component words are not distinguished in this

approach, it is not possible to describe relations between them (i.e., the internal structure) either. The *chunks* class of representations is similar to words-with-spaces in the sense that chunks are placed below syntactic structures. However, this approach allows to represent the internal structure of MWEs, thanks to the fact that it keeps the individual MWE component words separated. The remaining four representation classes – *subtree*, *asynchronous*, *overlay*, and *bidirectional* – correspond to the *embedded*, *parallel*, *above*, and *intertwined* types of the SYNTAX/MWE INTERFACE, respectively.

It is also mainly the SYNTAX/MWE INTERFACE characteristic which distinguishes our typology from, e.g., the one described in (Constant et al., 2017). One of the clear differences is that we consider an approach to be an instance of the *chunks* family of approaches only if chunks provide lexical nodes over which syntactic structures are constructed. The more traditional point of view on chunks does not put up such requirement. We think, however, that the question of the MWE-syntax interface is crucial, not only from the point of view of treebanking but also of MWE-aware parsing, which is the origin of this slightly less orthodox typology presented here. Approaches which represent MWEs independently from syntactic structures – whether they use chunk-like structures or recursive graphs – are thus of lower interest to us and we classify them all as *asynchronous*.

4.1 Words with spaces

The first approach, typically called the *words-with-spaces* approach, consists in concatenating together (using a special character as a separator, e.g. a space) all the components of a given MWE to form a single graphical word. It is motivated by the fact that certain continuous MWEs have a clearly atomic interpretation. MWEs represented with this methodology are essentially indistinguishable from simplex tokens. In constituency treebanks, words with spaces are thus represented as terminal leaves, while in dependency treebanks – as lexical nodes on top of which dependencies are defined.

This approach is e.g. used to represent prepositional MWEs in the NorGramBank (Rosén et al., 2015) (see Fig. 4.1 (a)), based on the motivation that prepositional MWEs are typically fixed and thus undergo neither internal modifications nor morphological variations. Constant et al. (2017) mention several drawbacks of this approach related to its limited expressiveness – it cannot be used to represent discontinuous MWE, and it hinders semi-

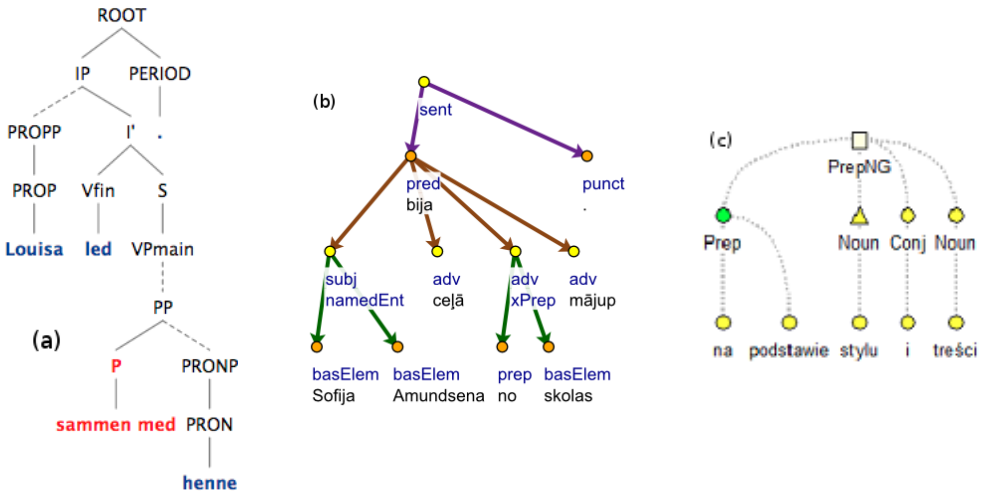


Figure 4.1: The words-with-spaces and chunking approaches in practice: (a) A constituent tree with words with spaces representation of the complex preposition *sammen med* 'together with' in the NorGramBank. (b) A chunking representation of the named entity *Sofija Amundsena* in the Latvian treebank. The MWE-related subtree can be seen as a chunk because its root does not refer to any lexical item, as nodes in dependency trees normally do, but rather groups together MWE components in a chunking-like manner. (c) A complex preposition *na podstawie* 'on the basis of' represented as a syntactic word on top of which a prepositional group is defined (an example from the National Corpus of Polish).

compositional semantic analysis since it makes it impossible to retrieve internal structure of MWEs. From this one can conclude that the words-with-spaces approach may be appropriate to represent the so-called multitoken words (cf. Def. 4, p. 27), written accidentally with spaces (or other separators) inside but otherwise semantically non-compositional, continuous, and (up to morphosyntactic variations) fixed, but it is inappropriate for the representation of other types of MWEs.

4.2 Chunking representation

The chunking representation consists in representing MWEs as *chunks*, i.e., as simple, continuous phrases constructed directly over words. In the chunking

approach MWE representation is typically flat (we, however, do not require this property in our typology) and, as defined in Tab. 4.1, chunks form lexical nodes of syntactic structures, which makes it akin to the words-with-spaces approach. On the other hand, it allows the individual component words to be assigned e.g. part-of-speech information. One could also imagine discontinuous chunks to form lexical nodes of a syntactic tree, but such design is rarely used in practical applications or treebanks.

An example of this approach is the Latvian Treebank, which is essentially dependency-based but employs special, phrase-style constructions for person names and, in general, multiword named entities without clear syntactic structure (Rosén et al., 2015) (see Fig. 4.1 (b) for an example).[3] A superficially similar solution is used in the ssj500k Dependency Treebank for Slovene, where multiword named entities are also marked as chunks but, in contrast to the Latvian Treebank, they are internally analysed at the syntactic level, as any other nominal group, which to our eyes makes it rather an instance of the asynchronous approach.[4]

Another treebank which employs the chunking approach is the National Corpus of Polish (NCP). It models prepositional MWEs as *syntactic words*, i.e., in a layer on top of which *syntactic groups* are defined (see Fig. 4.1 (c)). The same approach is used for continuous adverbial (e.g. *na czczo* 'on an empty stomach') and conjunctive (e.g. *a zatem* 'therefore') MWEs. Interestingly, discontinuous conjunctive MWEs – as the one shown in Ex. 4.1 – are also joined to form single syntactic words in NCP, in spite of their discontinuity. This suggests that some forms of discontinuous MWEs could also be treated as chunks (or words-with-spaces) in parsing systems, even if we are aware of no experiments performed in this direction so far.

(4.1) **zarówno** wczoraj, **jak i** dziś (PL)
both yesterday, how and today
'both yesterday and today'

The chunk-based approach is also used in the French Treebank (FTB), where continuous MWEs are (by default) uniformly represented as flat trees,

[3]Similar representation is sometimes used in dependency parsing systems. For instance, irregular MWEs form lexical nodes in the transition-based dependency parser of Constant and Nivre (2016).

[4]There seems to be no mechanism to ensure that multiword named entities actually form nominal groups in this annotation scheme.

both in the constituency version (Le Roux et al., 2014) and in the dependency version (Candito and Constant, 2014).[5] While it can be classified as an instance of the subtree approach (Constant et al., 2017), covered in the following section, it exhibits all the properties of the chunk-based approach as defined in Tab. 4.1, even if MWEs following it are represented directly in the syntactic layer.

According to Constant et al. (2017), a standard way of representing chunks in tagging systems is the IOB encoding method. While straightforward in its basic formalization – tag B(egin) is assigned to the chunk's left-most token, tag (I)nside to other tokens covered by the chunk, and tag (O)utside to the remaining tokens – this approach is in general quite powerful and allows to represent not only discontinuous (Schneider et al., 2014), but also recursively embedded (Waszczuk et al., 2013) MWEs. Ex. 4.2 shows how the discontinuous *zarówno jak i* from Ex. 4.1 can be encoded as a single, discontinuous chunk in this approach.

(4.2) **zarówno** wczoraj, **jak i** dziś (PL)
B O I I O

While IOB encoding is used in tagging and chunking systems, it is not used in treebanks, thus it can be seen as a practical encoding rather than an annotation scheme. Finally, even if it were used as an annotation scheme, we would rather classify it as an instance of the asynchronous approach, since typically IOB encodings are not guaranteed to correspond to syntactic structures.

4.3 Subtree analysis

The *subtree analysis* essentially consists in representing MWEs as connected subtrees (or substructures, in general) in the corresponding syntactic structures (either phrasal or dependency-based, depending on the character of the underlying treebank). A *subtree* should be understood here in its mathematical sense – as a tree whose graph vertices and edges form subsets of the sets of vertices and edges of the underlying tree, respectively.

The defining characteristic of this approach is that MWEs are not explicitly marked but rather distinguished from regular syntactic structures through

[5]Even though in the first version of FTB, as mentioned by Constant et al. (2017), continuous MWEs were represented using the words-with-spaces approach.

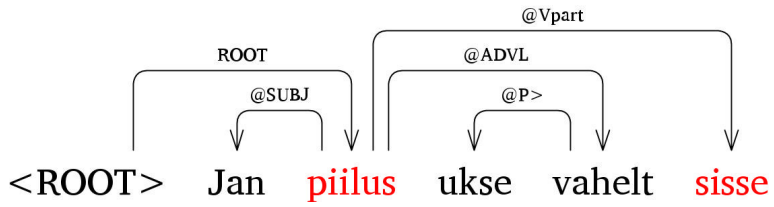


Figure 4.2: An analysis of the verb-particle construction (EE) *piilus sisse* (lit. $peek_{pst.3.sg}$ $in_{particle}$) 'to peek' in the Estonian treebank. Both words are connected using a dedicated `@Vpart` dependency.

the use of special phrasal or dependency labels. In other words, labels must join the syntactic and the MWE-dedicated pieces of information. It allows to represent the internal structure of MWEs, which is especially convenient within the context of syntactically regular, semantically compositional MWEs.

This approach is sometimes used to model verb-particles, e.g. by using a special dependency label connecting a particle with its head verb (as e.g. in the Estonian and Hungarian dependency treebanks, see Fig. 4.2 for an example), or a special phrasal label for the nodes directly dominating the corresponding particles (as e.g. in the German TIGER constituency treebank, see Fig. 4.3). In NorGramBank, based on the LFG formalism, the relation between the verb and its particle is marked in the *f(unctional)-structure*, which essentially describes functional dependencies between the individual lexical units of the sentence. More specifically, information about the particle is integrated into the predicate of the resulting verb-particle construction, provided that the construction is effectively semantically non-compositional.

This last point brings up an important property of verb-particles – some of them can be, in fact, analysed as semantically compositional. The expression *put up*, for example, can have both idiomatic and compositional occurrences, exemplified by Ex. 4.3 and Ex. 4.4, respectively[6].

(4.3) To *put up* a flag

(4.4) To *put up* a friend for the night

To properly annotate verb-particle MWEs in treebanks one should therefore use a dedicated label to distinguish the idiomatic from the compositional

[6]Example borrowed from the PARSEME shared task annotation guidelines, see `http://parsemefr.lif.univ-mrs.fr/guidelines-hypertext/`.

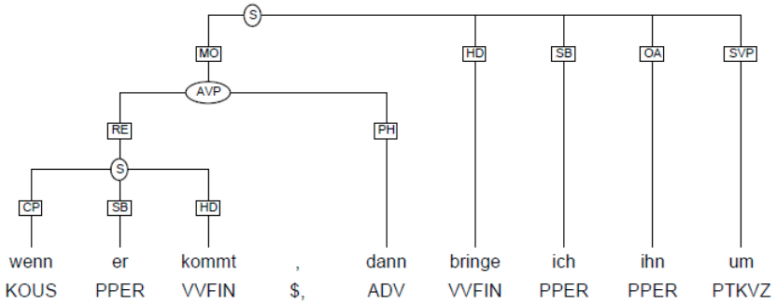


Figure 4.3: An analysis of the verb-particle construction (DE) *bringe um* (lit. $bring_{1.sg}$ *at/for/around*$_{particle}$) 'to kill' in the German TIGER treebank. The particle is marked using a dedicated `SVP` label, reflecting its relation with the head verb of the entire construction.

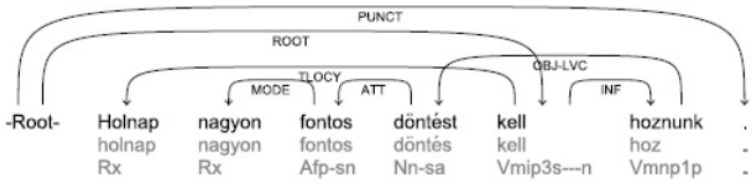


Figure 4.4: Analysis of the light-verb construction (HU) *döntést hoznunk* (lit. $decision_{acc}$ $bring_{inf.1.pl}$) 'make decision' in the Hungarian treebank. Dependency label `OBJ` relating both words is enriched with the `-LVC` suffix.

occurrences. Technically, such a solution is used by Vincze et al. (2013) to annotate light-verb constructions (LVCs) in the Hungarian dependency treebank. More precisely, their representation consists in enhancing the corresponding dependency relations with information about such constructions (e.g., `OBJ` → `OBJ-LVC`).

Candito and Constant (2014) experimented with the subtree approach in order to represent continuous yet syntactically regular MWEs in the dependency version of the French Treebank. In this representation dependency labels are enriched with a suffix containing information about the MWE status of a given expression and its part-of-speech. They also considered an alternative representation where information about MWEs is not preserved in labels but rather as values of dedicated features attached to the individual lexical nodes.

This approach is also sometimes used to model prepositional MWEs (e.g.

in the SQUOIA Spanish treebank or in the Dutch LASSY Small Treebank), but in this case – prepositional MWEs are typically fixed – annotations are flat and information about the MWE is marked in the root of the tree encompassing the entire expression. Similar approach is used in the French Treebank, where continuous MWEs are (by default) uniformly represented as flat trees. As mentioned in Sec. 4.2, such a solution exhibits all the properties of the chunk-based approach but, in fact, it also satisfies the properties of the subtree approach as defined in Tab. 4.1. Flat MWE-dedicated subtrees can be seen as placed below syntactic trees proper, or as embedded in syntactic trees.

The subtree method can be thus seen as an amalgam of three more specific MWE-representation approaches. If the subtrees used to represent a given MWE are also subtrees in the information-science sense – i.e., all the nodes reachable from the MWE's nodes via the parent-child relation form a part of this MWE – then such MWE can be represented using chunks. Alternatively, if by stripping away the MWE-related markings valid, purely syntactic structures are obtained, then such a representation can be seen as an instance of the overlay approach. Finally, if the relations between MWEs and syntactic structures are intertwined, then we may be dealing with the bidirectional approach.

4.4 Asynchronous annotation

The main property of the asynchronous approach, as defined in Tab. 4.1, is that MWE annotations are independent from syntactic structures and, if they even actually overlap with them, they are not formally required to. This approach does not allow to reuse structures present at the syntactic level when describing MWEs, which entails additional and potentially unnecessary (since already carried out) work, nor does it allow to reuse MWE annotations (related to fixed, irregular MWEs, in particular) in syntactic structures. Another pitfall of the asynchronous approach in general is that it makes it non-trivial to search for patterns involving several annotation layers (Bejček et al., 2012). On the other hand, this design choice has the advantage of making the *process* of annotating MWEs independent from the process of annotating syntactic structures, which on the one hand simplifies it, and on the other hand allows later to update the two layers independently. It also allows to annotate one of the layers even when the other layer is not even present. Finally, synchronic

(as apposed to asynchronous) annotation can cause suboptimal representation choices: wrong decisions concerning the structures adopted in one layer can negatively influence the corresponding representations in the other layer, a phenomenon which can be seen as related to the issue of *error propagation* in the pipeline-based parsing systems. Thus it seems that in the short term the asynchronous approach can be advantageous, but in the long run it fails to represent potentially important connections between MWEs and syntax.[7]

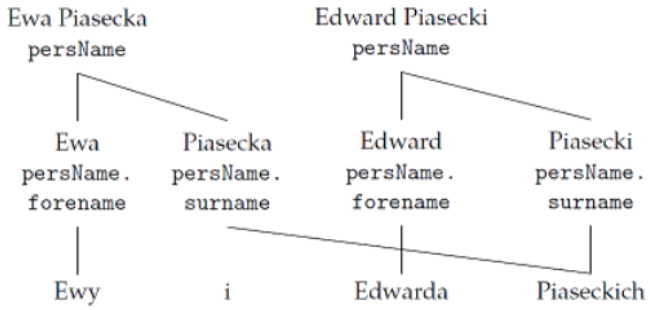


Figure 4.5: Graph-based analysis of two coordinated named entities, *Ewa Piasecka* and *Edward Piasecki*, in the National Corpus of Polish. Note that the family name *Piaseckich* occurs only once on the surface.

Yet, due to its practical advantages, it is commonly used to annotate MWEs in treebanks. It is used, e.g., in the ssj500k Slovene Dependency Treebank to annotate multiword named entities at the morphological level. These annotations take the form of chunks, which are nevertheless given an independent, internal analysis at the syntactic level, as any other nominal group. Another example is provided by the National Corpus of Polish (NCP), in which multiword named entities are represented in a separate layer, independent from the (shallow) syntactic layer, and which allows to describe recursively embedded structures. Moreover, the NCP annotation scheme allows to represent coordinated named entities (see Fig. 4.5), which are formally characterized by graphs rather than trees (Savary and Waszczuk,

[7]Note that we do not advocate at this point any of the alternative representation approaches, but rather suggest that relations between syntactic objects and MWEs exist and should be ideally represented in treebanks, on the one hand, and also that syntactic structures and MWEs should "align", which is not guaranteed in the asynchronous approach, on the other hand. Otherwise, the task of semantic analysis – which in principle should take place over both the syntactic structures and MWEs – would be impossible to perform.

2012, p. 141). This also shows that the asynchronous approach does not enforce any particular structure of the MWE representations – they can be represented by chunks, as in the Slovene Treebank, or by recursive constituent graphs, as in the National Corpus of Polish.

Finally, the asynchronous approach is used not only in treebanks, but also in MWE-aware syntactic parsing systems. In (Constant et al., 2016), MWEs are represented as dependency trees independent from syntactic trees, and the parser does not guarantee any alignment between them. In (Le Roux et al., 2014), continuous MWEs take the form of IOB-encoded chunks, which the parsing system attempts (but does not guarantee) to keep synchronized with the corresponding syntactic structures.

4.5 Overlay

The main characteristic of the overlay approach, as defined in Tab. 4.1, is that MWE annotations are defined on top of syntactic structures. Clearly it is appropriate only for MWEs with regular syntactic structures. Its advantage, in comparison with the subtree approach, can be seen in the fact that it provides a conceptual separation between syntactic and MWE annotations. Both approaches can sometimes lead to equivalent representations, as exemplified by LVC annotations in the Hungarian treebank (see Fig. 4.4) which, even though annotated directly in syntactic dependency trees, could be very well removed from this layer by stripping the appropriate LVC-related suffixes and putting them in a separate layer containing references to the corresponding LVC-related dependencies.

A notable example of the overlay approach is the Prague Dependency Treebank (Bejček et al., 2012) (PDT). PDT annotates multiword named entities, foreign and numerical expressions, etc., on top of tectogrammatical (deep syntactic/shallow semantic) trees. More specifically, MWE annotations take the form of subtrees enriched with information about the corresponding MWE type. As shown in Fig. 4.6, the annotated MWEs can be internally modified, even though they are required to be continuous. This is possible because lexical units – even when discontinuous on the surface or in the shallow analytical level – become continuous in the semantic layer. On top of that, the tectogrammatical layer can contain nodes not present on the surface at all, thus facilitating the analysis and annotation of ellipsis-stricken MWEs.

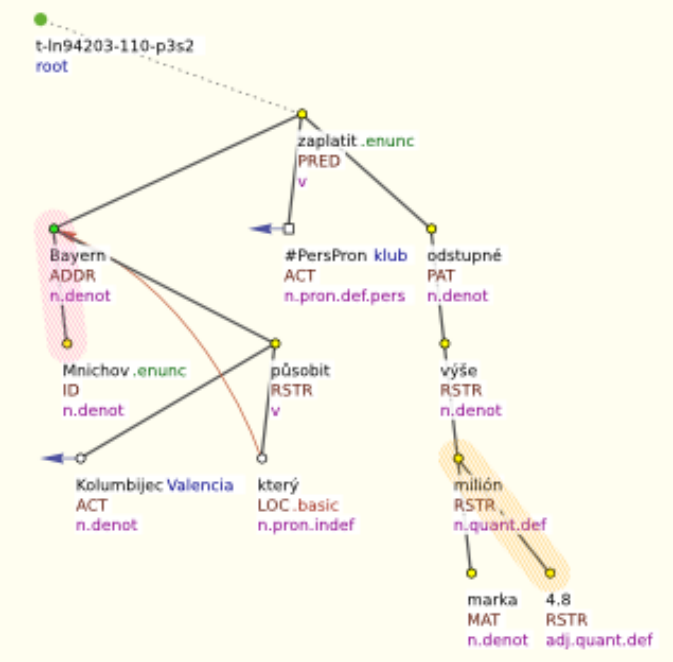


Figure 4.6: A dependency tree assigned in the Prague Dependency Treebank to the sentence shown in Ex. 4.5. It contains two multiword annotations – the named entity *Bayern Mnichov* and the numerical expression *4.8 millión* '4.8 million' – highlighted in red and orange, respectively, over the continuous tectogrammatical subtrees.

As explained in Sec. 3.4, PDT contains valency annotations which specify, for a given tectogrammatical node – in particular, for every occurrence of a verb – which of its direct dependents are bound (or required) in this particular context[8] and which are optional.[9] There is a valency lexicon, called PDT-Vallex (Hajič et al., 2003; Urešová, 2009), accompanying the development of the PDT treebank, and its individual entries – *valency frames* – can be seen as flat tectogrammatical subtrees whose annotation consists in finding their individual occurrences in the tectogrammatical layer of the treebank. This point of view reveals certain similarities between MWEs and valency frames, since both apply to continuous substructures at the tectogrammatical layer.

[8]The context determines the word sense, which in turn determines the number and character of the bound terms.

[9]Let us note that certain arguments – e.g. subjects – are commonly omitted in Czech on the surface level, but they are restored in the tectogrammatical level.

Moreover, MWEs of an important subtype – verbal idioms – are actually described in PDT-Vallex and thus annotated in PDT in a different way than e.g. named entities, which raises the questions of whether (i) the methodology of valency annotations in PDT could be applied to different types of MWEs, and (ii) annotations of MWEs and valency frames should (or could) not be represented in a unified manner.

(4.5) **Bayernu Mnichov**, v němž Kolumbijec působil, zaplatí odstupné
Bayern Munich in which Colombian operated will-pay gratuity
ve výši **4,8 miliónu** marek.
in amount 4.8 million marks

'They will pay a gratuity *4.8 million* marks to *Bayern Munich*, where the Colombian has operated.'

4.6 Bidirectional

The bidirectional approach is based on the *intertwined* SYNTAX/MWE INTERFACE, in which syntactic structures and MWEs rely on each other. This type of MWE representation can be often observed in treebanks based on specific, more elaborate linguistic theories. Fig. 4.7 shows an LFG-based analysis of the sentence from Ex. 4.6, which contains a verbal idiom, in the NorGramBank. Information about the verbal MWE (VMWE) is marked in the PRED feature of the f-structure, which represents the predicate-argument relations between the main verb and its complements. In this particular case, the predicate is complex and it is composed of the component words of the VMWE (`holde#øye*med`), in order to distinguish it from other predicates whose main verb is `holde`. Between the `<...>` brackets the SUBJ and the OBL-TH arguments are specified. The direct object is indicated outside of the brackets, signifying that it is subcategorized for but semantically void (the semantics it brings is already accounted for in the complex predicate).

(4.6) Samtidig **holdt** han umerkelig et skarpt **øye med** jentungen.
simultaneously kept he unnoticeably a sharp eye with the girl child

'At the same time he furtively kept a close eye on the girl.'

In the so-called "PRED-only" view, where only predicate words and relations between them are preserved, f-structures take the form of dependency

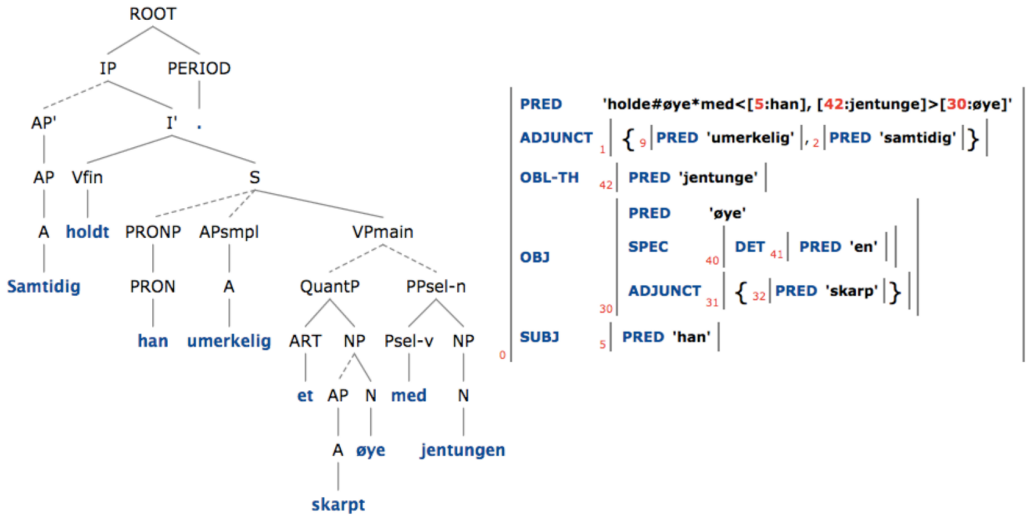


Figure 4.7: The c-structure and the f-structure (simplified) assigned to the sentence from Ex. 4.6. Both the example and the figure borrowed from Dyvik et al. (2017).

graphs whose nodes are either simplex words or MWEs, and whose arcs are annotated with grammatical functions (subject, object, adjunct, etc.). Looking from this perspective, MWEs can be seen as lexical nodes of syntactic structures, which reveals the similarities between the LFG-based representation of MWEs and the chunks approach and, consequently, the bidirectional character of the former.

4.7 Conclusions

Various MWE characteristics can influence the choice of an appropriate MWE representation approach. In this work, we introduce a novel typology of MWE annotation in treebanks, based mainly on the MWE/syntax interface and, consequently, the issue of syntactic irregularity emerges as a factor with a dominant impact on this decision.

Two representation approaches stand out as particularly appropriate in dealing with syntactically regular MWEs – overlay and bidirectional. The choice between the two seems to lie in the underlying linguistic theory. In LFG, for example, predicate-argument structures are accounted for at the syntactic

layer (in f-structures, more precisely) and since it is not possible to describe them properly without describing MWEs, too, the intertwined interface turns out indispensable. In PDT, MWE and valency annotations are put on top of tectogrammatical trees and this solution seems to satisfy all the desirable properties of a MWE representation approach with respect to syntactically regular MWEs. On the other hand, tectogrammatical descriptions and PDT-Vallex entries are mutually constrained, which brings the question whether it is not really an instance of the bidirectional approach.

Conversely, even if a given syntactically irregular MWE could be formally described using the regular syntactic objects (dependencies, labels, phrases, features) underlying a given syntactic formalism, the subtree solution has certain disadvantages. In particular, it may lead to abuse of syntactic notation in inappropriate contexts which, in turn, may lead to less clear interpretation of the meaning of the syntactic objects. For instance, while *by and large* could be modeled as a coordination of a preposition and an adjective, such an interpretation is unconvincing. Moreover, one may deduce from it (and this is highly probable in case of automatic syntactic analysis methods) that prepositions can actually coordinate with adjectives, which obviously is not a viable generalization. Another example is (PL) *do rany przyłóż* (lit. *to wound apply*) 'gentle', which performs a function of a predicative adjective, which is not immediately obvious – and should not be deduced in general – from its internal preposition-noun-verb structure.

It therefore follows that amongst the methods presented in Tab. 4.1, it is the chunks approach which seems best equipped to represent syntactically irregular MWEs. Of course, in both dependency and constituency frameworks, the words-with-spaces approach could substitute the chunking approach. The former is very restrictive, though, and should be probably used with caution. Nevertheless, it seems safe to apply it to totally frozen expressions whose elements cannot be individually co-referenced from outside – complex prepositions (e.g. *together with*), particularly irregular MWEs (e.g. *all of a sudden*), foreign expressions (*ad hoc*), etc.

Chapter 5

MWEs and parsing

In this chapter we look at the particular issues and challenges MWEs introduce in the domain of syntactic parsing, as well as at how MWEs are accounted for in the existing parsing systems, both purely statistical and symbolic ones.

5.1 Selected issues in parsing

In this work we understand the notion of *parsing* as a process which leads from an unstructured text – a sequence of characters – to a structured representation which includes:

1. Division of the unstructured text into paragraphs and sentences
2. Segmentation of the individual sentences into tokens
3. Morphosyntactic structure which provides, for each token, information about its grammatical class (or part-of-speech – verb, noun, adjective, etc.) and grammatical categories (number, gender, person, case, etc.) within the context of the sentence
4. Syntactic structure of the sentence, which may represent relations between its words (dependencies) or their groupings (constituents)
5. Semantic structure of the sentence, in correspondence with its syntactic structure

MWEs, as linguistic objects with idiosyncratic properties, interfere with all these levels of linguistic representation (perhaps apart from the first one

– division into paragraphs), and especially with the level of the semantic structure. Semantic non-compositionality is a quasi-universal property of MWEs and, therefore, semantic structures cannot be properly handled without an appropriate account of MWEs – i.e., if MWEs are not recognized and their structure linguistically represented.

In this work we are mainly concerned with the layers of *segmentation*, *morphosyntactic structure* and *syntactic structure*, with a particular focus on the latter. An example of a division into layers is the Prague Dependency Treebank, where the *morphological layer* provides information about morphosyntactic structures of individual sentences in the underlying corpus, the *analytical layer* provides surface-syntactic relations between lexical (morphosyntactic) units, and the *tectogrammatical layer* – defined directly on top of the analytical layer – provides deep syntactic (shallow semantic) representations.

5.1.1 Pipeline architecture

An important property of the division into linguistic layers shown above is that each representation layer depends on (i.e., is constructed on top of) the layer immediately below. Thus segmentation relies on the division into sentences, since it is performed within the context of a sentence. Morphosyntactic structure can be determined only if a particular segmentation is chosen, since grammatical classes and categories are assigned to particular segments. Syntactic structure is built over lexical nodes, and if segmentation is unknown one cannot tell what fragments of the sentence should play their (i.e., the nodes') role. Similarly, the syntactic structure provides a scaffolding for a semantic layer.

This conceptual separation of linguistic representations into layers and their sequential ordering leads to the idea of pipeline approaches to parsing. Such approaches basically consist in constructing the individual layers one by one, in a sequential manner, starting from the lowest level – division into paragraphs and sentences – and gradually moving up the stack, to semantic analysis. One of the main ramifications of this processing architecture is that, once a given layer is constructed, it is not possible to change one's mind and revise the decisions made before. For instance, in Ex. 5.1, assuming the words-with-spaces MWE representation approach (cf. Sec. 4.1), if the *tokenizer* decides that both *by and large* occurrences are MWEs, then the *syntactic parser* will not be able to reassess their classification as MWEs and,

consequently, to correctly analyse the sentence.

(5.1) While we were passing *by and large* clouds were gathering over our heads, my daughter said that this is *by and large* a nice place.

The main advantages of the pipeline approach are that it simplifies the parsing architecture and that, while focusing on one representation layer at a time, the parser can consider a significantly restricted space of possible structures. For instance, when the module of *syntactic parsing* is employed to determine the syntactic structure of the given sentence, already tokenized and tagged morphosyntactically, it only has to consider the possible syntactic structures which can be built on top of the lexical units given in advance, and it can limit the possible syntactic structures to those consistent with morphosyntactic information present in the previous layer. The main disadvantage of the pipeline approach is that, as already mentioned above, it does not allow to recover from errors made in the already processed layers, and sometimes information from the following layers needs to be considered in order to properly process the current one.

5.1.2 Symbolic vs. probabilistic parsing

Syntactic parsing techniques are often considered to be divided in two broad families of *symbolic* and *probabilistic* parsing. They can be characterized by the following set of contrastive descriptions.

Symbolic parsing is mainly concerned with *syntactic analysis*, the process of determining all the syntactically valid structures of a given sentence. Probabilistic parsing, on the other hand, is mainly concerned with *syntactic disambiguation*, i.e., the process of determining the most plausible syntactic analysis of a given sentence[1]. At first glance, this distinction may not seem very clear – one may wonder why there should be more than one syntactically valid structures for a given sentence. However, this is often the case in symbolic approaches, and at least for two reasons. Firstly, syntactic analysis methods are often ignorant of semantics and pragmatics, without which it is often impossible to correctly disambiguate between two different syntactic interpretations (as in the famous Ex. 5.2).

[1]The term *probabilistic* actually fits rather poorly to this description, since machine-learning methods in general can be used to perform syntactic disambiguation, and not all machine-learning methods are based on probabilities.

(5.2) One morning I shot an elephant in my pajamas. How he got into my pajamas I'll never know.

Secondly, symbolic methods can sometimes *overgenerate*, i.e they can produce more syntactic structures than a syntactic oracle (e.g. a native speaker and expert in linguistics) would deem as syntactically valid. This issue is related to several factors: the expressive power of the underlying symbolic formalism (it may not be powerful enough to express certain syntactic relations) or robustness (a robust parser should be able to parse sentences even when they violate some of the syntactic rules), among others.

Symbolic systems are typically based on the notion of a *syntactic grammar*, which determines which syntactic analyses are valid and which are not. Probabilistic systems typically make out without any explicit notion of syntactic grammar, which also means that they typically cannot be used for syntactic analysis (as defined above), just as purely symbolic systems cannot deal with the task of syntactic disambiguation. However, this apparent weakness of the probabilistic approaches is often seen as their main strength, since it makes them more robust.

The third axis differentiating these two approaches is related to the notion of *constraints*. Symbolic methods can be seen as constraint-oriented in the sense that the parser is supposed to retrieve all and only the structures satisfying certain constraints, typically stemming from the underlying grammar. In probabilistic parsing, the role of constraints is less apparent. Of course probabilistic parsers also adopt assumptions which delimit the space of permitted structures. An example from dependency parsing is the assumption that dependency structure is often required (i) to be a tree, and (ii) to be a projective tree (see Sec. 2.2). This kind of global constraints is clearly present also in symbolic approaches, which nevertheless make use of additional, local, grammar-driven constraints.

Nevertheless, it is worth noting that the two approaches are far from being contradictory. Indeed, even if the ultimate goal in the context of a particular application is to perform syntactic disambiguation, one can easily (theoretically speaking) use a syntactic analysis module to delimit the set of permitted structures in order to, for example, make the parsing disambiguation either faster or more accurate (or both). Conversely, even if syntactic analysis is to be regarded as the core function of the syntactic parser, practical NLP applications typically prefer to obtain only a single, most plausible syntactic structure for a given sentence, and one can regard

syntactic disambiguation as a subsidiary process which allows to satisfy this technical constraint. For instance, Dyvik et al. (2017) note that, because of the lexical and the syntactic ambiguity in the NorGram LFG grammar, parsing often results in many different syntactic analyses, which renders the step of syntactic disambiguation necessary. Finally, symbolic grammars can be typically enriched with probabilities. For instance, CFGs can be extended to probabilistic CFGs by (i) assigning to each rewriting rule a particular probability and (ii) making sure that the probabilities of all the rules rewriting a particular non-terminal sum up to 1. Other symbolic frameworks – TAGs, CCGs, HPSGs, etc. – can be similarly enhanced (Resnik, 1992; Clark and Curran, 2007; Miyao and Tsujii, 2008). Such probabilistic extensions lead to hybrid systems, which can be used for both syntactic analysis and disambiguation.

5.2 MWE-related issues in symbolic parsing

This section is dedicated to some of the MWE-related issues which interfere with the grammar-based view on syntactic parsing and, in particular, with the task of syntactic analysis, i.e., the task of determining all syntactic structures consistent with the underlying grammar and with the given input sentence. Below we describe several properties of MWEs which play an important role within this context.

5.2.1 Semantic non-compositionality

While often not related to syntactic analysis at first sight – semantic analysis takes place, conceptually, on top of syntactic structures – semantic non-compositionality actually provides the very motivation for including the MWE recognition results in the results of symbolic parsing. If MWEs are not accounted for, it is not possible to carry out correct semantic analysis, since MWEs are (almost universally) semantically non-compositional.

Conversely, if semantically compositional exressions are described as MWEs in a given symbolic grammar, the problem of spurious derivations may arise. Namely, it may be then possible to analyse a given sentence using two structures identical on both the syntactic and the semantic level.

5.2.2 Manifold linguistic status of MWEs

The linguistic status of MWEs is manifold – they exhibit properties of both words and expressions. It is thus not clear whether they should be handled in parsing with a mechanism dedicated to words, or rather to expressions. The manifold linguistic status of MWEs is first and foremost a challenge with respect to their representation in treebanks and Ch. 4 deals with this issue in detail. But, of course, this issue is related to the task of the syntactic analysis too – a wrongly chosen representation will, in principle, prohibit an appropriate analysis of MWEs in symbolic grammars.

5.2.3 Extended domain of lexical dependencies and morphosyntactic constraints

MWEs exhibit a property which can be called an *extended domain of lexical dependencies*.[2] Each MWE binds together, within a single semantic predicate, lexical nodes which can be quite far from each other in the corresponding dependency structure. Fig. 5.1 shows a surface syntactic dependency subtree which can be assigned to the expression (PL) *poruszyć niebo i ziemię żeby* 'move heaven and earth (to do something)', assuming the UD-based treatment of coordination and that the complementizer *żeby* 'so that' is analysed as a dependent of the main verb of the subordinate clause it introduces (which is also consistent with the UD scheme). The extended domain of dependencies can be further aggravated by the so-called *long-distance dependencies* which may involve, depending on the underlying linguistic theory, syntactic relations across an unbounded number of dependency arcs.

Just as MWEs can entail lexical dependencies in the corresponding dependency fragments, they can enforce morphosyntactic constraints on their elements. An example of this is (FR) *casser sa pipe* (lit. *break one's pipe*) 'to die', where the possessive pronoun must agree in number and person with the subject, even though the pronoun is not connected with the subject by a direct dependency relation.

[2]Note that this is a MWE-related property, to be distinguished from the (loosely related) *extended domain of a locality*, which is a property of linguistic formalisms (cf. Sec. 5.5.2).

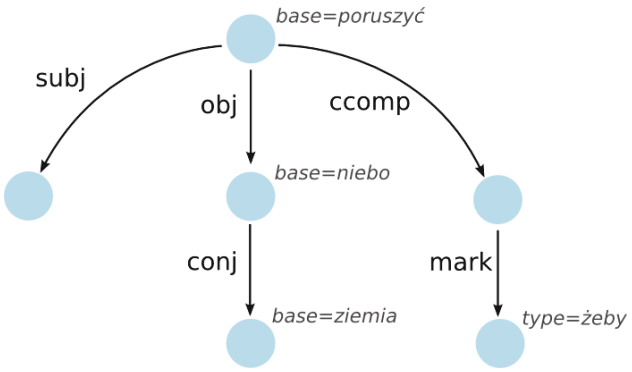


Figure 5.1: Possible dependency subtree for the MWE *poruszyć niebo i ziemię* 'move heaven and earth' assuming UD-based treatment of coordination. The conjunction *i* 'and' is not represented in the tree since it is not obligatory – it is possible to use other conjunctions (e.g. correlative conjunction ***ani*** *nieba* ***ani*** *ziemi* 'neither heaven nor earth' under the scope of negation (Przepiórkowski et al., 2017)), or to simply use a comma to separate the coordinated objects. In contrast to other lexically specified nodes, the complementizer is constrained to be a member of a specific lexical class, which includes not only *żeby* but also *aby*, *by*, *po to by*, etc., all meaning roughly 'so that'.

5.2.4 Discontinuity

Many MWEs can be discontinuous, which can be seen as a subcase of the issue of the extended domain of lexical dependencies. The challenge of discontinuity with respect to symbolic parsing is that discontinuous MWEs cannot be analysed as words (i.e., words-with-spaces or chunks). Therefore, in formalisms which adopt the grammar/lexicon separation, their idiosyncratic properties need to be accounted for in the lexical entries corresponding to their individual components words. In different terms, it can be non-trivial to describe a given potentially (or obligatorily) discontinuous MWE in a lexicon because the description may need to be split between several lexical entries rather than to be put within the scope of a single lexical description.

5.2.5 Irregular syntax

The property of irregular syntax applies to MWEs whose internal syntactic structure does not follow the regular grammar rules of the language. Such a property can be difficult to handle directly in syntactic formalisms which assume a distinction between syntactic and lexical rules, the latter originating from simplex words only. Modeling irregular syntax via generic syntactic rules can be inconvenient, in the best case, and impossible in the worst case. Alternatively, syntactically irregular MWEs can be modeled as chunks (provided that the underlying symbolic system allows it), which should alleviate the problem to some extent.

5.2.6 Variability

The issue of variability of MWEs within the context of syntactic analysis is mainly of a technical nature, but it is not negligible. A MWE can be often expressed in a multitude of different syntactic and morphosyntactic configurations. For instance, *to spill the beans* can be passivised (*the beans he has spilled*) without losing its idiomatic interpretation. This particular VMWE can be transformed as freely as regular verbal expressions, but MWEs which block some or all syntactic transformations are not uncommon. The MWE *casser sa pipe* mentioned above cannot be passivised, for example.

All the admissible configurations should be represented in the corresponding grammar/lexicon entries. This brings up a challenge related to the task of lexicon engineering. Appropriate factorization methods should be used to

capture similar syntactic/morphosyntactic behavior of different MWEs, so that a grammar/lexicon developer does not have to enumerate all the different configurations of a given MWE separately, a task which would clearly be tedious and inefficient given the large quantity of MWEs and their complex behavior on various linguistic levels.

On the other hand, when all the lexical, morphosyntactic, and syntactic variants of a given MWE are accounted for in a given symbolic grammar, issues related to parsing efficiency may arise.

5.3 MWE-related issues in statistical parsing

The same MWE-related issues which impact the task of syntactic analysis and symbolic parsing also influence the task of syntactic disambiguation and statistical parsing. We start our description by those and provide information specific to syntactic disambiguation. Then we follow with a couple of other issues which seem particularly relevant with respect to statistical parsing.

5.3.1 Semantic non-compositionality

Just as within the context of symbolic parsing, semantic non-compositionality provides one of the major motivations to perform MWE recognition (MWER) in the first place. The fact that semantics and syntax are often modeled jointly entails that MWEs, often placed somewhere between these two layers, need to be accounted for and modeled jointly as well.

5.3.2 Manifold linguistic status of MWEs

The manifold linguistic status of MWEs raises the issue of their representations in treebanks, which are major resources used to train probabilistic parser, and different representations may entail different parsing strategies. While different representation approaches can be more or less adapted to different types of MWEs, no general guidelines which would address this issue exist.

5.3.3 Extended domain of lexical dependencies and morphosyntactic constraints

The issue of the extended domain of lexical dependencies is even more prominent in statistical parsing than in symbolic parsing. This is related to the fact that statistical systems often adopt strong assumptions, thus sacrificing expression power for the sake of a more reliable parameter estimation and more efficient parsing. For instance, the dependency subtree presented in Fig. 5.1 exceeds the expressive power of the practical state-of-the-art graph-based dependency parsing methods which, in practice, can take into account at most three adjacent dependency arcs at once (Carreras, 2007). This makes it impossible to express a predicate which would check all the corresponding lexical and morphosyntactic constraints expressed in this subtree, which contains six arcs.

An ensemble of the lexical, morphosyntactic, and syntactic constraints enforced by a given MWE cannot be typically checked in case of more complex MWEs. Statistical methods can bypass this problem by factorizing the predicate corresponding to a given MWE by a set of local[3] predicates which can be expressed in the underlying system as model features. While the factorized representation is not equivalent to the original predicate, it has its advantages – notably, it may be more robust because it allows to capture MWE occurrences which do not satisfy all the prototypical constraints.

In case of syntactically regular MWEs, it is reasonable to check complex lexical and morphosyntactic constraints over an already constructed dependency tree – i.e. to perform MWE post-recognition after syntactic disambiguation. However, this excludes the possibility of exploiting information about possible MWE occurrences during syntactic disambiguation. Such information can be helpful in choosing the correct syntactic structure for a given sentence.

5.3.4 Discontinuity

The issue of discontinuity may play an even more important role within the context of statistical parsing than it plays in symbolic parsing. Symbolic parsing methods exhibit, in principle, no preferences between long-distance (in the sense of the surface distance between the words) and short-distance

[3] In the sense that they apply to smaller dependency fragments than the dependency representation of the entire MWE.

analyses. What matters in syntactic analysis is their syntactic validity. The situation is different in statistical methods which, by nature, learn to disambiguate between long-distance and short-distance dependencies. In particular, the former are typically harder to capture using statistics, hence MWEs which often occur in discontinuous surface configurations can turn out more difficult to deal with.

5.3.5 Irregular syntax

In practice, irregular syntactic structures are often represented in treebanks in the same layer as regular structures (cf. Sec. 4.3), which may confuse a statistical parser. Because of such irregular fragments, the parser may fail to learn useful syntactic generalizations (e.g. that two adjectives can very well coordinate but a preposition and an adjective – not necessarily). As in the case of symbolic parsing, an alternative is to use chunks to represent syntactically irregular MWEs. This, in turn, may require from the statistical parser to effectively deal with segmentation ambiguities (if chunks are represented as single segments) or the adaptation of the parsing algorithm to jointly model chunks and regular syntactic structures.

5.3.6 Variability and scarcity

Variability of MWEs within the context of statistical parsing methods is inseparably related with the issue of *scarcity* of MWEs. The latter issue is a real challenge for statistical methods, which typically require large annotated data to achieve high performance. It stems from the distributional nature of MWEs which, taken as a group, are quite frequent in many languages, but individually many of them are rare and thus they may be under-represented in training data. The issue of scarcity is further aggravated by the aforementioned variability. MWEs can often take many different syntactic and morphosyntactic configurations, which makes it difficult to automatically learn to recognize all their occurrences.

A potential solution for the issue of scarcity is provided by *lexical resources* which contain descriptions of MWEs. Such resources can be either used to pre-identify certain word combinations as possible MWE candidates[4] or as a source of model features.

[4]This applies mainly to continuous MWEs, since identifying discontinuous MWEs typically requires syntactic information.

5.3.7 Ambiguity

Pre-recognition of MWEs based on lexical resources does not solve the problem of MWE recognition in a particularly accurate way. This is related to one of the major MWE-related issues in statistical parsing – the task of MWER is highly interconnected with the tasks of tokenization and syntactic disambiguation.

For instance, *by and large* is generally used as a MWE and, thanks to the fact that it is fixed, it can be easily recognized before syntactic disambiguation (or even analysis) takes place. Such a strategy will most likely lead to an efficient MWER with respect to this particular sequence of words. However, it may fail to recognize its non-idiomatic occurrences, as the one in Ex. 5.3, despite the fact that contextual, syntactic information which allows to reject the MWE interpretation is easily accessible in this particular sentence.

(5.3) And it was a fantastic protest – lots of interest from people passing by and large numbers of conversations which were uplifting and positive in the main.

Provided that irregular MWEs are represented as words-with-spaces[5] (see Sec. 4.1), similar interactions apply between the task of tokenization and MWER. A decision of representing a given *by and large* occurrence as a single token is then equivalent with admitting that this occurrence is a MWE. By extension, interactions between MWER and morphosyntactic analysis and disambiguation also arise: MWER impacts syntactic disambiguation, while the latter impacts morphosyntactic analysis and disambiguation.

The *by and large* example is particular because it belongs to the class of the syntactically irregular MWEs, but similar reasoning extends to syntactically regular MWEs and, in particular, to discontinuous MWEs. Let us consider the expression (EN) *to make a good leader/programmer/husband/etc.* The fact that the two underlined lexical components of this MWE occur in a given sentence does not mean that it is present there (cf. Ex. 5.6 vs. Ex. 5.5) or that the syntactic relation between the two words is of the same nature (cf. Ex. 5.6 vs. Ex. 5.7). It is also possible that the sentence contains another, partially overlapping MWE, e.g. (EN) *make money* (cf. Ex. 5.4).

(5.4) You can make some pretty good side money working nights and weekends.

(5.5) You can make some pretty good NLP applications working nights and weekends.

[5] Or using any other approach in which MWEs are placed *below* syntactic structures.

(5.6) You can make a pretty good husband working nights and weekends.

(5.7) If I make it myself, I'm good to go.

Ex. 5.7 illustrates what can be called an **accidental co-occurrence** of the lexical components of this MWE, while Ex. 5.5 can be seen as an example of its **literal occurrence** (Constant et al., 2017). The former are particularly relevant within the context of the ambiguity issue because accidental co-occurrence of words, if recognized as MWEs, can induce wrong syntactic analyses (cf. Ex. 5.7).

5.4 Handling MWEs in statistical parsing

MWEs exhibit properties of both words and syntactic expressions, hence it is not clear what is an appropriate model to handle them in parsing. The goal of this section, heavily inspired by the work of Constant et al. (2017), is to describe the existing techniques of dealing with MWEs in statistical parsing, as well as to take the view on the so-called orchestration question – at which point the MWE identification should take place: before, after, or during syntactic parsing.

5.4.1 Pre-recognition approach

The first possibility is to perform MWER *before* syntactic parsing. It is an approach which fits into the so called *pipeline architecture*, where the output of each module provides the input for the following one (see also Sec. 5.1.1). Pre-recognition entails certain constraints on the representation of MWEs – it is mostly compatible with the representation approaches which assume that MWEs are placed *below* the syntactic layer, i.e., that MWEs (just as regular words) constitute lexical nodes of syntactic structures. In our typology of MWE representations in treebanks (see Ch. 4), this constraint is satisfied by the *words-with-spaces* approach and the *chunks* approach.

In the **words-with-spaces** approach MWEs are pre-identified and their components concatenated before the stage of syntactic parsing. For instance, given the sentence *The prime minister made a few good decisions*, the MWEs *prime minister* and *a few* are supposed to be identified first, then concatenated to yield a re-tokenized sentence *The prime_minister made a_few good*

decisions,[6] which is then provided as input for a syntactic parser, both for learning and parsing (Constant et al., 2017).

Seminal studies on the impact of the perfect MWE pre-recognition on statistical parsing have been carried out with the words-with-spaces approach used to represent pre-recognized MWEs (Nivre and Nilsson, 2004; Arun and Keller, 2005). They have shown significant improvements in parsing quality, involving not only MWEs themselves but also the surrounding structures.

Later works, e.g. Constant et al. (2013a) for dependency parsing and Korkontzelos and Manandhar (2010) for (shallow) constituency parsing, proved that improvements can be also observed in a realistic setting where MWEs are identified and concatenated by the parsing system itself.

Only continuous MWEs can be easily handled using this approach, thus for the sake of consistency it seems reasonable to apply it exclusively to fixed MWEs and semi-fixed MWEs which do not allow internal modifiers. Discontinuous MWEs, such as *made decisions* in the example above, remain separated in the re-tokenized sentence. At the same time, the words-with-spaces approach deals with the issue of syntactic irregularity – which applies mainly to continuous expressions – reasonably well, because it handles such irregular MWEs in isolation from syntactic structures. Irregular morphosyntactic properties can be easily expressed at the level of the concatenated segments. With respect to the issue of ambiguity, this approach is imperfect even within the context of fixed MWEs which can coincide with word combinations that cross phrase boundaries, as in *after [all the preparation], we finally left*. Nasr et al. (2015) emphasize the difficulty of committing to a single segmentation at the stage when the syntactic structure of the sentence is not yet explicit. They provide arguments that non-trivial morphological, lexical and syntactic clues (e.g. subcategorization frame or mood) must be taken into account in order to correctly spot occurrences of the continuous ADV+*que* and *de*+DET constructions, very frequent in French.

The issue of the extended domain of lexical and morphosyntactic constraints does not apply to the types of MWEs which can be reasonably handled using this approach, since such dependencies remain local. Variability of contiguous MWEs can be very high in morphologically rich languages and concatenations can increase the scarcity of the resulting lexical models

[6]The character _ can be indeed used as a separator, with a tacit assumption that using it will not cause conflation of concatenated MWEs with existing, simplex words. However, this is just a technical trick which is not necessary when tokens are allowed to contain spaces.

significantly. One can imagine replacing the individual components by their base forms, which nevertheless provides only a partial solution to the problem of scarcity.

Another pre-recognition approach is the **substitution approach**, which consists in replacing the recognized MWEs by their lexical heads, for instance (FR) *alors que* (lit. then that) 'while' by *que* 'that' or *multiword expression* by *expression*. Weeds et al. (2007) obtained positive results by using this technique in the domain of biomedical research, where gene and protein names are often MWEs (Constant et al., 2017). The advantage over the words-with-spaces approach is the reduced scarcity, which makes it easier for a statistical parser to learn relations between lexical units. This advantage comes with a price of decreased linguistic precision, because substitution can conflate expressions with different lexical attachment preferences and make them indistinguishable from the parsing point of view. Already *alors que* mentioned above is an example of this phenomenon – it performs a different function than *que* alone. The substitution approach also deals poorly with syntactically irregular MWEs (what could be the head of *all of a sudden*?) and with exocentric constructions in general. While the words-with-spaces method aggravates the issues of scarcity and variability, the substitution method seems to go too far in trying to alleviate it. With respect to other MWE-related issues both methods exhibit similar characteristics.

Yet another pre-recognition technique is based on the **chunking approach**, which relies on the chunk representation of continuous MWEs. As explained in Ch. 4, the main advantage of chunks over words-with-spaces is that chunks allow to represent the internal structure of continuous MWEs and can be thus used to represent numerical expressions, addresses, etc., whose common property is that they follow rules of a subgrammar which does not necessarily correspond to regular syntactic rules. From the parsing point of view, this approach allows to handle irregular continuous MWEs in a particularly appropriate way, and at the same time it is better in dealing with the issue of scarcity and variability than the substitution approach. Since chunks have an internal structure, their individual components can be exploited as model features separately or jointly, thus they provides flexibility not present in either the words-with-spaces or the substitution approaches.

The capability of the pre-recognition techniques in general to reuse lexical resources – particularly those focused on continuous MWEs – should be high. Continuous MWEs, together with their inflected forms, can be represented using e.g. finite-state approaches (Savary, 2005), which allow to search for

potential MWE occurrences in untokenized texts. Al-Haj et al. (2014) describe a morphological processing strategy which allows to pre-recognize occurrences of discontinuous MWEs. Such occurrences can be subsequently used in a form of dedicated model features to support and possibly improve the MWE recognition decisions (Constant et al., 2013b).

5.4.2 Post-recognition approach

The alternative pipeline architecture consists in identifying MWEs *after* syntactic parsing. It relies on the observation that most MWEs follow the rules of regular syntax and therefore their syntactic structures can be safely identified using standard parsing machinery. MWE recognition, on the other hand, is greatly simplified when the syntactic structure of a sentence is already known, because (i) in constituency parsing, an MWE generally covers a syntactic constituent, and (ii) in dependency parsing it generally constitutes a continuous path in the dependency tree (Constant et al., 2017). While the opposite observation is also true – it is easier to parse a sentence once MWE occurrences are known (cf. Sec. 5.4.1) – one can point out that strong lexical relations exist between the individual components of MWEs and that statistical parsers are often designed to model and efficiently recognize such relations. This approach is therefore especially appropriate for syntactically flexible MWEs and naturally predisposed to handle MWEs with potential internal modifications – components of discontinuous expressions, difficult to recognize using pre-processing methods, become closely related in syntactic trees.

In comparison with the preprocessing approaches, MWE post-recognition should be more helpful in resolving the cases of the MWE-related idiomatic vs. compositional ambiguity, in recognizing discontinuous MWEs, and in capturing connections between the different variations of a given MWE. The post-recognition approach certainly makes it easier to exploit rich morphological and syntactic information which can be found in MWE dictionaries, thus it provides a reasonable way to overcome the scarcity issue. When syntactic trees are known in advance, searching for syntactic nodes satisfying the constraints described in the corresponding dictionary entries is greatly simplified (due to reduced search space) and it can lead to satisfactory MWER results, as shown in (Savary and Waszczuk, 2017), where we obtained the MWE post-recognition precision of 85%. In contrast, Candito and Constant (2014) experimented with a simple MWE pre-recognition method – choosing the

maximum number of non-overlapping MWEs in the given sentence according to the lexicons – and obtained precision of barely 50%[7].

MWE post-recognition shows advantages not only with respect to the MWE pre-recognition approach, but also in comparison with the methods where the task of MWE recognition is embedded into the task of syntactic parsing, a family of approaches which we consider in Sec. 5.4.5. Nagy T. and Vincze (2014) found out that the Bohnet dependency parser was able to correctly recognize only 58.16% of the verb-particle constructions (VPCs) in the Wiki50 corpus. Other particle occurrences were often incorrectly tagged as prepositions or adverbial modifiers of the verb. By using a machine learning-based post-processing recognition method over the candidates selected among the verb-particle, verb-preposition and verb-adverbial modifier pairs, they were able to improve recall from 58.16% to 76.79%, with the accompanying drop in precision from 89.04% to 85.7%, which, to sum up, increased the resulting F-score by more than 10%.

On the other hand, the post-recognition method seems poorly adapted to handle the cases of syntactic irregularity, as in *by and large*, which cannot be reasonably analysed using regular syntactic rules and can be expected to cause syntactic analysis errors propagating to subsequents processing phases. A related disadvantage of the post-processing approach is that the syntactic parser cannot benefit from information about MWEs in general, not only those which are fixed. Potential occurrences of MWEs can help the parser to make correct disambiguation decisions, which has been shown to improve the results of symbolic parsing (Wehrli, 2014) as well as statistical parsing (cf. Sec. 5.4.1).

5.4.3 Word-lattice approach

Certain versions of the pipeline approach allow to represent ambiguity throughout the parsing process and, therefore, defer the MWER and/or syntactic disambiguation decisions. At the interface between tokenization and syntactic parsing, this can be achieved through the use of the so called *word-lattices*. At the interface between parsing and MWE post-recognition, this can be accomplished by looking at n-best trees produced by the parser and selecting the

[7]Note that these two numbers are not really comparable, since they relate to different types of MWEs, different MWE resources, different treebanks, and the corresponding recall values are not given, but the large gap between the two numbers is suggestive all the same.

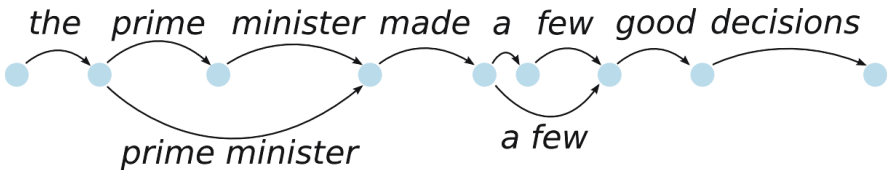


Figure 5.2: A word-lattice (DAG) representation of 4 different possible segmentations of the sentence *the prime minister made a few good decisions*, with two potential MWEs. Possible tokens are represented by DAG edges, and possible token break sites by DAG node.

most plausible one using re-ranking methods. The latter method is described in Sec. 5.4.4.

The word-lattice approach consists in representing the tokenization result in the form of a *directed acyclic graph* (DAG), which allows to preserve tokenization ambiguities (Woliński, 2006) and to delay their resolution to later processing stages. This is also a standard approach in speech processing, where transformation of the audio signal to a written representation is already too complex to perform it without contextual (syntactic, semantic) information, hence the need to express segmentation ambiguities under the form of a DAG (Nasr et al., 2011). Such a pre-processing architecture is also suitable to deal with noisy textual corpora which contain spelling errors and thus are not trivial to tokenize, and can be used to perform a non-deterministic recognition of continuous MWEs (Sagot and Boullier, 2005).

Within the context of MWE processing this approach is typically seen as an extension of the words-with-spaces approach, where MWEs are represented as separate tokens, i.e., as distinct edges in the segmentation ambiguity graph. Fig. 5.2 shows an example of such a graph for the sentence *the prime minister made a few good decisions*, in which 2x2 different paths can be taken from the left-most node to the right-most node, which represents the possibility of analysing both *prime minister* and *a few* either as MWEs or compositionally. In other words, in the word-lattice approach, MWER decisions are completely determined by tokenization decisions, but these decisions can be deferred virtually indefinitely in the processing pipeline, provided that the subsequent processing modules can handle and preserve such ambiguities. It is worth noting that the word-lattice approach could be very well adapted to the chunks representation of MWEs, although we are aware of no work within the context of MWE processing which would test this possibility.

Constant et al. (2013b) tested a variant of the word-lattice method in which n most probable paths – according to an underlying, sequential probabilistic model – are selected from the segmentation DAG, thus yielding a sub-DAG which is then processed by a syntactic parser. However, they obtained poor results using this approach, mainly due to their syntactic parser systematically preferring the shortest paths in the segmentation DAG, regardless of their probabilities. The PCFG-LA parser they used was preferring shorter paths due to its generative nature – intuitively, each additional segmentation arc was a source of an additional multiplier decreasing the resulting total probability. They obtained more satisfactory results when ambiguous segmentation was represented using the IOB-encoding, but this solution did not outpeform the baseline system working on the single-best segmentation either.

Constant et al. (2013b) relied on the sequential conditional random field, CRF (Sutton and McCallum, 2011) model to obtain the most probable paths in the segmentation DAG. Individual edges in this graph could be enriched with marginal probabilities – while CRFs assign probabilities to the individual paths, marginal probabilities of the individual edges – tokens – can be easily computed, too.[8] It seems plausible that they would have obtained much better results if they had used a discriminative syntactic parser, with no segmentation-length preferences, especially if such a parser could take advantage of the probabilities provided by the statistical tokenization module.

The word-lattice method deals with most MWE-related issues in a very similar way as the words-with-spaces approach. The notable difference is a potentially much better support for the MWE-related ambiguities, which is also the main drawback of the words-with-spaces method w.r.t. fixed MWEs. Thus, the word-lattice approach seems to provide a very promising way of handling fixed MWEs, since it allows to perform the more straightforward segmentation decisions early in the parsing pipeline, thus reducing unnecessary ambiguity early, while leaving the more difficult ambiguity cases for later processing units. Even for the latter cases, a tagger can provide information about the probabilities of different segmentation paths, thus simplifying the postponed disambiguation tasks.

[8]We implemented this functionality in a development version of Concraft, a morphosyntactic tagger for Polish (Waszczuk, 2012), thus verifying that it is not difficult to obtain.

5.4.4 Re-ranking approach

The re-ranking method consists in asking a statistical parser to produce the n most probable syntactic trees instead of the single best tree. Afterwards, a generic re-ranker is used to score the n best trees again, based on a more elaborated statistical model, and to choose the most plausible one. The advantage of this method is precisely that, in contrast to syntactic parsers, it works on a significantly pruned search space and can thus make use of more advanced statistical or machine learning machinery. It can be thus seen as an intermediate solution between performing MWER during syntactic disambiguation and MWE post-recognition. A potential disadvantage of this approach is that it may be hard to find the optimal value of n, and that the simpler parsing model can fail to include the correct solution in the heavily pruned output. Constant et al. (2012) employed this technique in their experiments, using a MWE-specific re-ranking strategy, and in comparison with other methods they tested – namely, a vanilla Berkeley parser and the Berkeley parser trained on concatenated (words-with-spaces) compounds obtained with a SOA CRF classifier – it turned out to be the best method for improving syntactic parsing accuracy using MWE-specific information.

Re-ranking performs the same function with respect to syntactic disambiguation as word-lattices with respect to tokenization. Looking from the perspective of MWE-related issues in parsing, both methods provide two different solutions to deal with MWE-related ambiguity (and ambiguity in general, in fact). The word-lattice method seems to be a reasonable approach for handling fixed, continuous MWEs, while re-ranking can be used to deal with attachment ambiguities arising within the context of flexible, possibly discontinuous MWEs. However, especially in case of re-ranking, the choice of the proper value of n is unclear. Syntactic ambiguity, especially in case of long sentences, can exclude the possibility of generating all the more plausible syntactic analyses, making the re-ranking approach either insufficiently accurate or impractical. An alternative could be to represent syntactic ambiguity in a compressed form – for example, a parsing hypergraph or a dependency shared forest – and to subsequently disambiguate thereupon.

Villemonte De La Clergerie (2013) uses the latter approach in the disambiguation step of syntactic parsing – possible derivations are first transformed into the corresponding shared dependency forest and, then, weighted rules are used to score the individual possible dependency structures encoded in the forest. While there is nothing MWE-specific in this solution, it can be seen

as an extension of the re-ranking approach, on the one hand, and it seems possible to adapt it to use MWE-related information to score the individual dependency trees, on the other hand.

5.4.5 Joint off-the-shelf parsing models

Yet another possibility is to recognize MWEs together with syntactic structures – neither before, nor after, but (conceptually) *at the same time*. This approach is taken up by the various so-called *joint* methods. It is motivated by the two previously mentioned observations – that MWER can improve parsing results and, the other way around, that knowledge about syntactic structures can improve MWER – from which one can conclude that neither of the simple pipeline approaches is sufficiently well adapted to efficiently perform MWER and syntactic disambiguation at the same time.

The easiest way to incorporate MWEs into statistical parsing is to directly add MWE annotations into a syntactic treebank used to train the parser. This solution is based on the *subtree* approach to MWE representation (see Sec. 4.3). Statistical parsers provide the output in the form they see in and learn from the training data. By adding MWE annotations, invisible from the parser's point of view in the sense that the parser is ignorant of their meaning, yet present in the same form as syntactic annotations, one can obtain a MWE-aware parser by simply training an off-the-shelf syntactic parser on syntactic data appropriately marked with MWEs. This approach was tested by Vincze et al. (2013); Candito and Constant (2014); Nasr et al. (2015), among others.

By adding a light-weight structure to treebank labels (without changing the syntactic trees themselves), so as to separate MWE-related information from syntactic information, it becomes possible to use MWE-dedicated model features and thus possibly handle MWEs more efficiently. However, we are aware of no work where this possibility would be investigated within the context of MWEs. Candito and Constant (2014), for instance, experimented with a structured representation where information about MWEs is preserved not within the labels but rather as values of dedicated features attached to lexical nodes. While convenient from the representation point of view, the graph-based dependency parser they used did not provide support for such features and thus it could not recognize regular MWEs for which this representation was used.

While the choice of using the existing parsing methods and algorithms to

jointly model syntax and MWEs comes with certain restrictions – notably, with the restriction that the internal structure of MWEs has to be represented according to the rules of syntactic representations, not necessarily well adapted to model all kinds of MWEs – it also enables the use of the common, generic and relatively powerful methods of feature-based engineering. The underlying idea of these approaches is that the score of the different parsing derivations can be factorized as a combination of the scores applying locally to their individual parts. For example, SOA graph-based dependency parsers adopt the following definition of the most probable dependency tree for a given sentence $W = w_1 \dots w_n$ (Nasr et al., 2015):

$$\hat{T} = \underset{T \in \mathcal{T}(W)}{\arg\max} \sum_{F \in \mathcal{F}(T)} s(F), \tag{5.8}$$

where:

- $\mathcal{T}(W)$ is the set of all possible dependency trees which can be constructed over sentence W,
- $\mathcal{F}(T)$ is the multiset of the model features[9] which together represent all the relevant properties of the given tree T, and
- to every feature the corresponding score $s(F)$ is assigned, which represents its impact (positive if $s(F) > 0$, negative if $s(F) < 0$, and neutral otherwise) on the probability of tree T, and which can be estimated during the parser's training.

This framework (which can be applied also to constituency parsing, see e.g. (Finkel and Manning, 2009) or (Le Roux et al., 2014)) allows to express relatively complex MWE-related predicates (under the form of model features), provided that they do not exceed the expressive power of the particular statistical model.

Nasr et al. (2015) show that occurrences of the construction *alors que* 'while' can be sometimes distinguished from the occurrences of the complementizer *que* 'that' which just happens to be placed next to *alors* 'then' (see Ex. 5.9) by looking at subcategorization of the governing verb. Some verbs (e.g. *manger* 'to eat') do not take as arguments subordinate clauses introduced

[9] In (Nasr et al., 2015) elements of $\mathcal{F}(T)$ are called *factors* and to individual factors many model features can apply. The definition we give here is thus its more abstracted version.

by the complementizer *que*. However, they can be very well modified by a subordinate clause introduced with *alors que*, as shown in Ex. 5.10. Such a rather complex correspondence can be expressed by binding – within the scope of a single, dedicated model feature – subcategorization information corresponding to a given verb together with the form of its direct complement. Based on training data, the parser should learn that, if the verb does not accept arguments introduced by *que*, it cannot be connected by a dependency link with *que* alone (see Ex. 5.11), but – on the contrary – it can very well govern a subordinate clause introduced by *alors que*.

(5.9) Je croyais alors que cela ne pourrait pas arriver à moi (FR)
I believed then that this could not happen to me

(5.10) Je mange alors que je n'ai pas faim (FR)
I eat while I have no hunger
'I eat even though I'm not hungry'

(5.11) * Je mange que je n'ai pas faim (FR)
* I eat that I have no hunger

As mentioned in Sec. 5.3.3, statistical parsing models typically take up strong assumptions regarding the form of model features, mainly for the sake of efficiency and reliability of the parameter estimation process. The latter is strongly related to the size of the feature space – the more parameters the model contains, the easier it is to fit it to the training data, but this typically leads to less robust models which work poorly on out-of-domain texts. In graph-based dependency parsers, efficiency depends roughly on the number of the adjacent dependency arcs (possibly disregarding their directions) a given model feature can bind. This number is called the *order* of the parser. For instance, the parsing time complexity is $\mathcal{O}(n^3)$ for 1st order[10] dependency parsing[11] and $\mathcal{O}(n^4)$ for 2nd order dependency parsing (Carreras, 2007). Practical graph-based dependency parsers are typically of the 2nd order. As a result, it is not possible to express all the lexical, morphosyntactic, and syntactic constraints of certain more complex MWEs (see e.g. Fig. 5.1) under the form of atomic model features. Again, as mentioned in Sec. 5.3.3,

[10] A 1st order parser can look at, at most, two adjacent dependency arcs at once.

[11] It goes down to $\mathcal{O}(n^2)$ if dependency trees are not required to be projective (McDonald et al., 2005), but this somewhat surprising behavior does not extend to higher-oder dependency parsing algorithms.

the totality of such constraints can be simulated in a factorized way, using a set of smaller constraints which together add up to roughly (even if not 100% faithfully) represent the characteristics of a given MWE, and such a factorized solution also has its advantages – notably, it should exhibit higher robustness.

In terms of the parsing quality, none of the MWE-related works mentioned above, based on the off-the-shelf approach to MWE processing, actually confirmed the intuition that MWEs can help syntactic parsing. The dependency parsing results of Vincze et al. (2013) were slightly worse, in terms of labeled attachment score (LAS), when LVCs were marked in dependency labels, doubtlessly due to increased scarcity. Nasr et al. (2015) did not observe important gains when evaluating their dependency parser on FTB, which had a very small number of the constructions they focused on. Finally, in (Candito and Constant, 2014), the joint solution was only slightly (i.e., insignificantly in most of their experiments) better than the system where regular MWEs were post-recognized. Interestingly, the latter approach was better in terms of MWER in general, while the fully joint approach performed better in recognizing irregular MWEs.

With respect to MWE-related issues, the subtree approach does not seem to be particularly well adapted to deal with syntactically irregular MWEs. Firstly, such MWEs should not be accounted for in syntactic structures in a fully regular way. Secondly, doing so may confuse statistical parsers, which may draw invalid conclusions from idiosyncratic structures seen in training data. It is possible to use special labels to distinguish idiosyncratic, non-syntactic dependencies (e.g., `dep_cpd` in (Candito and Constant, 2014) or `morph` in (Nasr et al., 2015)). This, however, can aggravate scarcity-related issues. All this can be alleviated to some extent by the use of appropriate feature engineering methods but, as we have seen, not all MWE-related properties are easy to express in the form of model features in statistical parsers, notably because of the extended domain of constraints exhibited by MWEs.

The subtree approach deals naturally with MWE-related ambiguities. Its big advantage is that it allows the two tasks of MWER and syntactic disambiguation to influence each other. The issue of MWE-related discontinuity disappears in syntactic structures where, most of the time, MWEs become continuous fragments.

Finally, the issues of variability and scarcity may be hard to deal with in this approach. Training data on which statistical syntactic disambiguation

methods rely are limited and can be expected to contain only a small portion of MWEs occurring in a given natural language. And even those which occur in the training treebank may occur only in a limited number of morphological forms and syntactic configurations. In some of the works mentioned above, the joint models suffered from the scarcity issues resulting from the concatenations of the syntactic dependency labels with the MWE-related annotations. A potential response to that is the use of lexical resources – MWE-aware lexicons, in particular – but, as mentioned above, it is restricted by limitations of the expressive power typically adopted in statistical parsers.

5.4.6 MWE-aware parsing models

This section contains a description of another family of joint solutions which, in contrast to the strategies described in Sec. 5.4.5, allow to use a representation of MWEs separate from syntax (or even several different representations for different types of MWEs). This strategy does not enforce any particular MWE/syntax interface, but it implies (as joint methods generally do) that the two levels are processed synchronously. In contrast to pipeline-oriented solutions, this strategy assumes that both syntactic and MWE structures should be modeled jointly. This approach is more challenging than the one described in Sec. 5.4.5 – since syntax is structurally separated from MWEs, dedicated algorithms for parsing and learning need to be designed in order to use it in practice. On the other hand, it allows to use structures better adapted to represent MWEs.

Three recent works adopt this approach to joint MWE/syntax modeling. In terms of representation, Le Roux et al. (2014) (who focus on continuous MWEs only) propose to use a MWE representation completely decoupled from syntactic structures. MWER takes the form of MWE-aware tokenization decisions and continuous MWE segments are represented via a dedicated IOB encoding (see also Ex. 4.2, p. 40). The system of Constant et al. (2016) also assumes that syntax is decoupled from MWEs, but it adopts a representation where lexical segmentation takes the form of a dependency tree, independent from the syntactic dependency tree. This representation allows to capture deep lexical analyses like nested MWEs, e.g. *I will (take a (rain check))*, and it can handle discontinuous MWEs, as shown on Fig. 5.3 (a). Such a representation is in fact equivalent to a phrase-structure forest-based representation where the individual, potentially embedded and potentially discontinuous MWEs are represented using phrasal nodes (see Fig. 5.3 (b)). The dependency tree-based representation can be seen as an encoding of such a phrase-structure forest.

Constant and Nivre (2016), in contrast, propose a mutually linked representation in which the syntactic dependency tree is built over lexical nodes, the latter representing either simplex words or irregular, continuous MWEs. Regular MWEs, on the other hand, remain decoupled from the syntactic structure. This representation can also handle discontinuous and nested MWEs. Its main difference in comparison with the representation of Constant et al. (2016) is that (i) irregular, continuous MWEs form lexical nodes of the corresponding syntactic structures, and (ii) lexical segmentation is not represented as a dependency tree, but rather as a phrase-structure forest

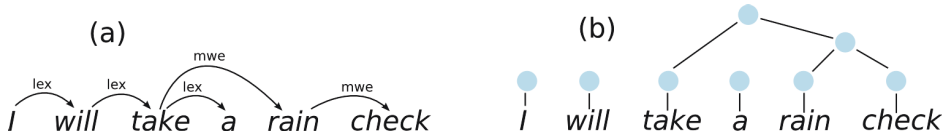


Figure 5.3: On the left (a), lexical segmentation represented as a dependency tree using the method of Constant et al. (2016). Adjacent lexical units (either MWEs or simple tokens) are connected with left-to-right arcs labeled with `lex`. Each dependency subtree corresponding to a MWE is rooted in its left-most token and arcs from this token, labeled with `mwe`, lead to all its lexical subtrees (either embedded MWEs or simple tokens). A special `submwe` label (not shown here) is used to allow the left-most token to be embedded in a subtree itself. On the right (b), an equivalent phrase-structure forest-based representation of the same lexical segmentation.

similar to the one shown in Fig. 5.3 (b). Point (i) means in particular that, while Constant et al. (2016) use the parallel MWE-representation approach to all types of MWEs, Constant and Nivre (2016) reserve it to syntactically regular MWEs, while fixed MWEs are represented using chunks.

Different mechanisms can be used to ensure the correspondence between the MWE-dedicated and the syntactic structures. One of the more elegant and extensible approaches consists in designing NLP modules in a way which makes them conceptually separated and which, at the same time, allows to plug them together in order to obtain a joint system. When parsing, such a system strives to find a joint solution which satisfies interface constraints between the individual modules. A prominent example of this strategy being applied to MWE-aware parsing can be found in (Le Roux et al., 2014). Their method is based on several MWE-aware tokenization modules, which individually allow to recognize continuous MWEs, part-of-speech tagging modules, and parsing modules. All these modules are implemented within the framework of conditional random fields (CRFs) and the choice of a unified framework allows to combine them together using a method called *dual decomposition*.[12] More precisely, the individual models are plugged together within a joint parsing (decoding) iterative algorithm which mutually constrains the models

[12]The authors observe that the use of a linguistically rich model for a given task may lead to data sparseness problems, while simpler models may suffer from insufficient linguistic precision, hence they propose to use several models, with different fine-grainedness levels, per task.

by promoting solutions which satisfy well-formedness constraints – in this particular case, the MWE-related segmentation constraints. For instance, if the MWE-aware tokenization module proposes a segmentation assuming that *bien que* is a MWE, than the parser should also adopt this choice. Otherwise, the method imposes scoring penalties on joint solutions in which differences in the segmentation choices exist. Importantly, the individual models are trained independently, which means that only the decoding algorithm needs to be adapted to model MWE-segmentation and syntactic structure jointly in the *dual decomposition* strategy, while the learning algorithm remains intact.

In the system proposed by Constant et al. (2016) no priority between the tasks of lexical (MWE-aware) segmentation and syntactic parsing is assumed. Both linguistic layers are parsed at the same time and features from both are available at any step of the parsing process, which allows the tasks to help each other mutually. An atomic parsing action takes the form of combining two adjacent dependency trees – or rather, their corresponding roots – which can be linked by the relation of dominance in two possible ways. At each step of the parsing process several actions can be applied, and the parser chooses the most plausible one (with the highest score). The subsequent decisions are constrained by the previous ones. The idea underlying this parsing strategy, called *easy-first*, is that the actions easier to determine should be performed early, and the more difficult ones should be performed later, hoping that the easy actions will help to resolve the difficult ones as a by-product. The exact flow of information is to be learned by the system itself. The disadvantage of the method is that it does not guarantee any correspondence between the two dimensions and the resulting lexical and syntactic layers can be out of sync.

In the solution proposed by Constant and Nivre (2016), irregular MWEs (which form lexical nodes together with simplex words) and syntactic trees are directly linked with each other, in the sense that the latter build up on the former. Regular MWEs do not directly correspond to syntactic structures, but the authors propose to model them jointly in the hope that recognizing them side-by-side with syntactic structures will have positive impact on syntactic parsing results.

In terms of MWER, in the experiments carried out by Le Roux et al. (2014), the system combining several CRF-based MWER modules outperformed all the CRF-based modules used individually, as well as a simple majority-wins voting system based on these modules. By adding the parsing component to the system, MWER results were increased even further, supporting the claim that syntactic information is needed to correctly handle ambiguity of

continuous MWEs. The findings of the other two works were less positive in this respect. Neither Constant et al. (2016) nor Constant and Nivre (2016) observed positive impact of the syntactic dimension on the predictions in the lexical, MWE-aware layer. Constant et al. (2016) postulated that this might be related to low frequency of discontinuous MWEs in their treebanks, and those which were present in the treebanks mostly had a gap of one token only.

In terms of syntactic parsing accuracy, Le Roux et al. (2014) obtained a very competitive parser out-performing the best system submitted to the SPMRL 2013 shared task. It was better than all the other architectures they tested in their experiments, including the pipeline-based ones. The authors believed that their system was more accurate overall because their method is more resilient to an error from one of its components. Constant et al. (2016) observed that lexical information tends to help syntactic prediction, but they observed no significant improvement when using the joint approach over the pipeline approach where the lexical dependency tree is constructed first and supplied to the syntactic parser in the form of model features. While Constant and Nivre (2016) observed significant improvements over a baseline, a joint system in which MWE information is concatenated with dependency labels, their dependency parsing results were significantly below those obtained with the state-of-the-art parsing systems. The authors noted, however, that their system could be potentially improved through the use of several advanced techniques related to transition-based parsing in general.

Concerning the MWE-related issues, the MWE-aware parsing models deal with them in a similar way as the joint, off-the-shelf parsing models described in Sec. 5.4.5, since both strategies adopt the joint point of view on syntactic parsing and MWER. The main differences between them are that the MWE-aware parsing approaches (i) allow to use separate data structures to represent MWEs, and (ii) promote conceptual separation of the two tasks. The latter is also important because one can not forget that syntactic parsing and MWER are not the only two tasks existing in the domain of NLP. They are also not the only ones which influence each other in a kind of a feedback loop either. It is hard to imagine that off-the-shelf syntactic parsers could be universally used to solve all kinds of NLP processing tasks, considering that such tasks can entail a virtually unlimited number of smaller sub-tasks which need to be solved. This is why the use of joint methods which assume some level of conceptual separation between the individual sub-modules seems very important in the long run. This is also why such methods, applied to the particular task of joint MWER and syntactic parsing, seem very promising

– they can be potentially reused in larger NLP systems, including not only MWER and syntactic parsing but also subsequent processing tasks.

MWE-parsing models can also provide a more adapted approach to deal with different types of MWEs. For instance, the treatment of regular and irregular MWEs in (Constant and Nivre, 2016) differs entirely, which is well motivated by the fact that they are very dissimilar on several linguistic levels.

Finally, concerning the issues of variability and scarcity, separation of the MWE layer from the syntactic layer permits to imagine joint, MWE-aware methods in which the MWE-dedicated component provides an extended domain of locality[13] with respect to MWE-related lexical and morphosyntactic constraints. While MWEs are often complex in this respect, individually they are also relatively infrequent and their potential occurrences are easy to spot. The latter is important because it allows to pre-filter potential MWEs present in the given sentence using simple, supertagging-like techniques, and thus significantly prune the search space of the MWE-dedicated parsing component. Handling MWEs with higher-order parsing models could be thus potentially tractable and could lead to both more efficient and more accurate MWE-aware parsing systems.

5.4.7 MWEs in statistical parsing: conclusions

Tab. 5.1 compares the different strategies of handling MWEs in statistical parsing based on their capabilities of dealing with the individual MWE-related issues introduced in Sec. 5.3.[14]

The methods described in the 4 top rows of the table do not provide appropriate mechanisms for dealing with the issue of discontinuity and, by extension, they cannot deal with the extended domain of dependencies in general either. In contrast, the post-recognition and the re-ranking approaches cannot deal with the issue of irregularity.

The immediate conclusion is that dedicated, MWE-parsing methods can

[13]To be understood here as a property of models (see also Sec. 5.5.2) which, among others, allows to account for the extended domain of MWE-related dependencies (cf. Sec. 5.3.3).

[14]We omit two of the issues mentioned in Sec. 5.3. The semantic non-compositionality provides a motivation for including MWEs in parsing results, but investigating the connections between parsing approaches and semantic modeling is beyond the scope of this section. As to the manifold linguistic status of MWEs, it is a challenge related primarily to the question of the MWE/syntax interface (cf. Ch. 4), and interacts with the tasks of MWER and syntactic parsing via the other issues considered in Tab. 5.1.

MWE-handling strategies in statistical parsing	extended domain of dependencies	discontinuity	irregular syntax or morphosyntax	variability, mitigating scarcity	ambiguity
Words-with-spaces	-	-	+	-	-
Substitution	-	-	-	+	-
Chunking	-	-	+	+	-
Word-lattice	-	-	+	+	+
Post-recognition	+	+	-	+	~
Re-ranking	+	+	-	+	+
Off-the-shelf	~	+	~	~	+
MWE-aware	+	+	+	+	+

Table 5.1: Comparison of different strategies of handling MWEs in statistical parsing based on their capabilities of dealing with the various MWE-related issues. To each strategy/issue pair one of the following values is assigned: + if it deals with the issue reasonably well, ~ if it is able to handle it but not as well as other comparable strategies summarized here, and - if it does not provide an appropriate mechanism to deal with the issue.

potentially deal with MWEs in the most optimal way. A less apparent conclusion is that by joining the word-lattice approach with the re-ranking approach we could also obtain an optimal system. However, it seems that only the MWE-aware parsing models can conveniently deal with the (arguably rare) examples of discontinuous and yet (morpho-)syntactically irregular MWEs, such as (PL) *Bóg zapłać* (see Sec. 3.2). It is also only the MWE-aware class of parsing solutions that allows to give a unified account to different types of MWEs, both regular and irregular.

5.5 Handling MWEs in symbolic parsing

As we have seen in Sec. 5.1.2, the main points of divergence between symbolic and statistical parsing methods are (i) syntactic grammars, not present in purely statistical systems, and (ii) the focus on syntactic analysis in symbolic sytems and on syntactic disambiguation in statistical systems. In this section we are going to see a couple of symbolic approaches to syntactic parsing and to MWE-aware syntactic analysis in particular.

The different goals of syntactic analysis and syntactic disambiguation entail different MWE-related issues. While within the context of disambiguation we are mostly concerned with the question of which methods work best in

discriminating more plausible from less probable syntactic analyses, in the symbolic setting we want to answer a more clearly defined question – which analyses are syntactically valid and which are not. While this question seems easier to answer, it entails that the focus is put on precise and linguistically motivated grammar-based descriptions of morphology, syntax, and other layers. Thus, from the perspective of MWEs, the goal of symbolic methods is to give a precise account of the various idiosyncratic properties of MWEs, a goal which is often beyond the scope of interest of statistical methods.

5.5.1 Mechanisms of marking MWEs

To begin with, it is important to note how different symbolic frameworks actually mark MWEs, i.e. inform about their occurrences in syntactic structures assigned to a given sentence.

In LFG, flexible MWEs are analysed as complex predicates which incorporate their individual lexical components and which describe their argument structure (Dyvik et al., 2017). It is the former which is specific to MWEs, since the latter applies to predicate words in general. For instance, the verb-particle *look sth. up* can be represented by a dedicated predicate `look*up` which takes a subject and a direct object as arguments. In the so-called "PRED-only" view, where only predicate words and relations between them are preserved, f-structures take the form of dependency graphs whose nodes are either simple tokens or MWEs, and whose arcs are annotated with grammatical functions (subject, object, adjunct, etc.; see also Sec. 2.3). Thus, LFG marks MWEs as lexical nodes in predicate-argument dependency structures, even if their descriptions come from lexical entries attached to simplex words.

In TAGs, introduced in more detail in Ch. 6, we assume the approach described by Abeillé and Schabes (1989), where MWEs are represented using elementary grammar trees. MWEs are then directly visible in TAG derivations – namely, they constitute nodes of derivation trees, just as other, regular elementary trees. From the dependency perspective, if we ignore the internal structure of the individual elementary trees present in a given derivation (i.e. map them to the sets of their component words), we obtain a view which is very similar to what we see within the context of LFG: a dependency tree whose nodes are either MWEs or simplex words (provided that all participating trees are lexicalized). A notable difference is that f-structures in LFG can form general graphs, and that their edges are annotated with grammatical functions. It is possible to obtain functional annotations in

TAG derivations by specifying functional relations within the scope of the individual elementary trees, but this is not a part of the definition of the TAG framework.

5.5.2 Extended domain of locality of elementary grammatical units

A mechanism which allows to deal in symbolic frameworks with the issue of the extended domain of MWE-related lexical dependencies and morphosyntactic constraints (cf. Sec. 5.2.3) can be called *extended domain of locality of elementary grammatical units*. It essentially means that atomic units of a given grammatical formalism – e.g. elementary trees in TAGs, lexical entries in LFG or HPSG, etc. – can bind together lexical and morphosyntactic properties of lexical nodes potentially distant from each other in the corresponding dependency trees.

Lexical items in LFG, including MWEs, specify the subcategorization and morphosyntactic requirements of the individual words. These requirements are expressed in terms of equations on the corresponding f-structures. All such equations can take up quite complex forms and, importantly, they are not restricted to the immediate neighborhood of the f-structure node corresponding to the given lexical item.

For instance, in the syntactically flexible expression (FR) *NP vider DET sac* (lit. *NP empty DET bag*) 'to express NP's secret thoughts', the possessive determiner DET embedded in the direct object of the verb *vider* 'empty' must agree in person and number with the subject NP (Abeillé and Schabes, 1989). Otherwise, the idiomatic meaning is lost, e.g. *ils ont vidé son sac* should only be interpreted as semantically compositional 'they have emptied his bag'. This expression can be represented by the following idiomatic description assigned to the verb *vider*:

$$\begin{array}{ll} (\uparrow \text{ OBJ PRED FN}) =_c \text{'sac'} & \\ (\uparrow \text{ OBJ SPEC POSS}) = f & \\ (f \text{ PERS}) = g & (f \text{ NUM}) = h \\ (\uparrow \text{ SUBJ PERS}) = g & (\uparrow \text{ SUBJ NUM}) = h \end{array} \tag{5.12}$$

The first line of the above description specifies that the name of the predicate assigned to the object must be *sac* 'bag'. The operator ↑ refers to the f-structure node corresponding to the verb *vider* in the f-structure assigned

to the underlying sentence. Further down, the description assigns the variable f to the possessive specifier of the object, where (↑ OBJ SPEC POSS) designates the f-structure node which can be reached from ↑ by following a path of three dependency-like arcs with the features OBJ, SPEC and POSS, respectively. Finally, the values of number (f NUM) and person (f PERS) of the possessive specifier are required to agree with the corresponding values assigned to the subject ((↑ SUBJ NUM) and (↑ SUBJ PERS), respectively).

Following (Sheinfux et al., 2015), an HPSG analysis of a particular MWE is distributed between the lexical entries specifying its individual lexical components. These components are mutually linked via a kind of a lexical chain. For instance, the lexical entry associated with the (idiomatic interpretation of the) head verb of the VMWE illustrated in Ex. 5.13 (Sheinfux et al., 2015, p. 125) requires a specific, lexically-constrained prepositional argument. This argument is lexically described in the entry assigned to the corresponding (idiomatic) preposition *me* 'from' which, in turn, selects for the particular, lexically constrained noun phrase 'the-tools'.

(5.13) dan hoci et dana me-ha-kelim (HE)
Dan took.out ACC Dana from-the-tools
'Dan made Dana lose her temper'

The extended domain of locality of TAGs is a well-known property which boils down to the fact that TAG elementary trees (ETs) are complex trees of unbounded size and that, thanks to adjunction, two nodes of a given ET can turn out arbitrarily distant from each other in a derived tree. Moreover, in feature structure-based TAGs, it is possible to attach feature structures (FSs) to the individual ET nodes. When a derived tree is attached to a given non-terminal node via the operation of substitution or adjunction, its FS is required to unify with the node's FS[15]. If unification fails, the operation is not allowed. This means that FS-based computations can be defined over the entire elementary grammar units and that potentially complex predicates can be verified within the scope of ETs. Fig. 5.4 shows an ET corresponding to the MWE *vider son sac*. The FSs attached to the individual nodes express the constraints described in Eq. 5.12.[16]

[15]We slightly simplify the picture by not adopting the typical distinction between `top` and `bottom` FSs.

[16]There are, in fact, two values of number in a `PossD` – one agreeing with the subject, and the other with the object. The latter is regular for possessive pronoun / noun pairs.

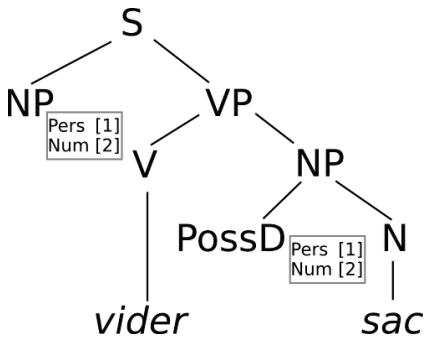


Figure 5.4: A TAG ET corresponding to the MWE *vider son sac*. FS decorations enforce that the subject agrees with the possessive pronoun. The example assumes that the determiner is modeled via substitution. Otherwise, an additional obligatory-adjunction constraint would have to be specified.

Since the size of ETs is not restricted, the distance between the mutually constrained nodes in the corresponding dependency structure is, in principle, not limited either. The design-related limitation is that all such constraints must be present in the same ET. Of course, technically, it is also possible to lexically constrain nodes by threading lexical information through feature structures attached to the individual nodes in FS-based trees. This alternative solution, however, is less convenient. It is also worth noting that, typically, FSs in TAGs are required to be flat, which limits their expressive capabilities.

5.5.3 Words-with-spaces

LFG assumes a distinction between the grammar and the lexicon, which makes it non-trivial to model syntactically irregular MWEs. Thus, fixed expressions are often modeled as words-in-spaces (cf. Sec. 4.1), for example in the NorGram LFG grammar (Dyvik et al., 2017) or in the Arabic LFG grammar of Attia (2006).

In lexicalized TAGs, the grammar and the lexicon are both represented in the form of the elementary grammar units, which allows to give a unified account to both fixed and flexible MWEs, even if it does not exclude the use of words-with-spaces.

A potential advantage of the unified representation is that, clearly, it can be used to represent semi-fixed MWEs as well. LFG requires that the status of such MWEs – as fixed expressions or not – is ascertained in advance. Some

MWEs can then be wrongly classified as fixed, which may complicate their further analysis. For instance, in the following short dialogue:

- Do you like **New York**?

- Not really, I like **the old one** better.

the words-with-spaces representation of *New York* makes it impossible to annotate *the old one* as a reference to its component word, *York*. On the other hand, syntactically irregular MWEs, if classified as flexible MWEs in LFG, need to be accounted for at the level of the (productive) grammar rules.

5.5.4 Dedicated grammar rules

Another technique which allows to account for the various MWE-related irregularities is the use of dedicated grammar rules. There is no lexicon/grammar distinction in lexicalized TAGs (LTAGs), thus this technique trivially underlies the LTAG-based descriptions of both MWEs and simplex words. In contrast, in formalisms which adopt certain lexicon/grammar distinction, e.g. LFG., its application to MWEs is clearly not self-evident – descriptions of MWEs, objects of a lexical nature, should be ideally limited to lexicons.

Verb-particle constructions can be seen as constructions which should not be handled by productive grammar rules. Only specific pairs, e.g. *look up* 'search for a reference' but not *look at* (in which case *at* is unambiguously a preposition), can be interpreted as MWEs. In the English ParGram grammar this issue is handled by the mechanism dedicated to subcategorization frames. The standard, transitive VP rule accepts a particle either before or after the direct object of the verb (VP →V PART OBJ | V OBJ PART, in simplified terms), while unification over the PRT-FORM feature, defined in both particle and verb lexical entries (e.g. (↑ PRT-FORM) = 'up' for the particle *up*) guarantees that only appropriate verb-particle constructions are recognized.

A similar solution is used to model correlative conjunctions such as *either* _ *or* _ and *both* _ *and* _. They are handled by a coordination rule with an optional 'preconjunction'. Every preconjunction (*both*, *either*, ...) specifies (as a value of the COORD-FORM feature) with which conjunction it combines and unification guarantees that only the corresponding pairs are recognized.

NorGram includes a set of special lexical and syntactic rules which allow to account for complex numerals, such as *hundre og to* 'one hundred two', which can be seen as a subtype of MWEs (Dyvik et al., 2017). It is plausible

that addresses – which are modeled e.g. in PDT following a special, dedicated annotation scheme (Bejček and Straňák, 2016) – should be modeled in LFG in a similar way.

Asudeh et al. (2013) provide an example of a Swedish traversal construction, (SW) *Jonas knuffade sig in i mängden* (lit. *Jonas pushed* SELF *in inside crowd.*DEF), which is distinguished by the requirement for the presence of a verb, a weak reflexive (coindexed with the subject), and a directional PP. It also exhibits a certain word-order peculiarity: the particle follows the direct object of the verb, while normally it would adjoin to the verb. Thus, the authors claim, the syntactic structure of the expression can be most elegantly modeled by a dedicated c-structure rule.

5.5.5 Templates and meta-descriptions

Grammar frameworks like LFG, HPSG or TAG typically come with mechanisms which allow to conveniently account for syntactic variations of different words, on the one hand, and to avoid descriptive redundancy, on the other hand. Such mechanisms, which are typically designed with simplex words in mind, often apply to MWEs as well and can be used to group together lexical descriptions of different types of MWEs.

In HPSG, lexical descriptions are grouped into inheritance hierarchies which enable them to share common properties, whether they describe MWEs or simplex words (Sheinfux et al., 2015). In LFG, templates play a similar role. They are essentially parametric functions which compute lexical descriptions, i.e. expressions – disjunctions, conjunctions, etc. – over functional equations. A typical design is to develop one template per subcategorization frame and call it from the corresponding lexical entries. The template can then call subordinate templates, which describe the different syntactic transformation a given word can undergo. Templates for different types of MWEs can be described in a similar way.

For instance, the NorGram LFG grammar uses the template shown in Eq. 5.14 – parametrized by the verb and the corresponding particle – to describe the class of Norwegian verb-particle constructions (Dyvik et al., 2017). The first line of the definition constructs the name of the predicate. `CONCAT` is just a template which allows to concatenate several values and put the result in a variable (here: `%FN`). Thus if we call `V-SUBJ-PRT-OBJ (look up)`, then the value of the `FN` variable will be set to `look*up`. The next line defines the predicate of the entire verb-particle construction. ↑ designates here the

corresponding f-structure, and ↑ PRED means that we define the feature value under the PRED path in this f-structure. In the last line, the template checks that the form of the particle (assigned to the feature PRT-FORM by grammar rules) actually corresponds to the particle given in the invocation of the template.

```
V-SUBJ-PRT-OBJ (P prt) =                                    (5.14)
  @(CONCAT P '* prt %FN)
  (↑ PRED)='%FN<(↑ SUBJ)(↑ OBJ)>'
  (↑ PRT-FORM) =c prt
```

The template V-SUBJ-PRT-OBJ can be then reused in the lexicon in the definitions of the other verb-particle constructions which subcategorize for a subject and an object. F-descriptions of the other types of MWEs can be factorized using templates in a similar manner.

In this formalization, all MWEs (as well as all simplex words) are "free" by default – they can be freely modified and undergo all the transformations accounted for in the grammar – but the individual entries can be easily constrained using additional functional equations. This allows, for instance, to specify that the MWE *to kick the bucket* cannot be passivised[17].

Templating is not the only LFG-related mechanism which can be used to account for the lexical and syntactic variation of words. Patejuk (2016) showed that it is possible to compile Walenty (cf. Sec. 3.4), a valency dictionary for Polish (Przepiórkowski et al., 2014), to the set of the corresponding LFG lexical entries compatible with POLFIE, a LFG grammar for Polish (Patejuk, 2015). These entries can be subsequently combined with the lexical entries stemming from Morfeusz, a morphosyntactic analysis tool (Woliński, 2014), and consequently used for LFG-based syntactic analysis of Polish sentences. This method does not rely on the templating mechanism of LFG (even though it can be very well combined with it), but rather on the high expressive power of the language of LFG functional descriptions which allows to express relatively complex functional constraints.

[17]This is one of the possible designs. Alternatively, passive and active forms could be described using separate templates, called from the corresponding lexical entries independently.

The idea of compiling high-level, factorized descriptions to concrete grammars has led to the development of the so-called meta-grammatical methods to grammar engineering. This approach, oftentimes used within the context of TAGs (Candito, 1999; Villemonte de La Clergerie, 2010; Crabbé et al., 2013), can be seen as a generalization of Patejuk (2015)'s method in the sense that (i) it applies not only to lexicon entries but also to grammar rules, and (ii) it adopts the high factorization of syntactic and lexical patterns as a goal in itself. While it has not yet been used extensively to describe MWE-aware grammars, the meta-grammar approach can be considered as a promising tool for the description of MWE-aware grammars as well as MWE-dedicated lexicons (Lichte et al., 2017).

LFG, HPSG, and TAG differ in their expressive power. Nevertheless, all three formalisms provide mechanisms which allow to deal with the MWE-related issues described in Sec. 5.2. We therefore omit a comparison table similar to Tab. 5.1 – it would trivially contain pluses in all its cells.

5.6 The choice of the grammatical formalism

In this work, we focus on the relations between MWEs and symbolic parsing, in general, and between MWEs and TAG parsing, in particular. Both symbolic and purely statistical methods have their advantages. We chose the former because of the notion of the grammar – typically underlying symbolic systems – which provides a human-understandable interface between data and the parser, an interface which is not explicit in purely statistical approaches. Such an interface allows a more fine-grained control over the process of syntactic analysis. Mistakes made by the parser can be analysed and resolved via further grammar refinement which, incidentally, can lead to better understanding of the internal workings of natural languages. In purely statistical methods it is often difficult to manually influence the behavior of the parser, which is typically a black-box. If it does not perform well, one can either extend training data in order to cover a phenomenon which causes the problem (a process which is costly), or engineer dedicated model features. However, neither solution provides a guarantee that the parser will learn to properly deal with the problem, and modifying model features may lead to unexpected changes in the behavior of the parser in other contexts.

With respect to MWEs, symbolic systems typically exhibit a significantly extended domain of locality in comparison with purely statistical systems.

As shown in Sec. 5.5.2, in both LFG and TAG elementary grammar units can be used to represent MWEs, and within the scope of such elementary units non-trivial morphosyntactic and lexical constraints can be specified, which allows to account for the various idiosyncratic constraints and requirements of MWEs. Furthermore, both LFG and TAG allow a perspective where MWEs are treated as semantically atomic units which can be composed together with other MWEs and simplex words via more or less regular semantic composition process. The difference with respect to purely statistical methods is that the latter rarely pay attention to provide a scaffolding over which semantic composition can be easily performed. For instance, while regular MWEs are recognized in the dependency parsing technique proposed by Constant and Nivre (2016), their output representation is decoupled from syntactic structures, and it is unclear how semantic analysis could be carried out on top of that. Finally, the use of a formal grammar as a basis for syntactic analysis does not exclude the possibility of performing subsequent disambiguation over the analysis' results.

As to the choice of TAG over e.g. LFG, it is the former which allows to treat MWEs as first-class citizens of a natural language, not distinguished from their regular counterparts – simplex words. In LFG, MWEs also can be described in the lexicon using the same machinery as regular words. However, LFG adopts a relatively strong lexicon/grammar distinction, and sometimes MWEs require the corresponding descriptions at the level of grammar rules as well, which may render the grammar engineering process more complex. Finally, the fact that MWEs are represented in TAG as elementary lexicon-grammar units entails a shift in perspective which allows to see syntactic parsing as a process of combining opaque – but supplied with appropriate syntactic and semantic interface – lexical objects, some of which happen to be MWEs. In other words, MWEs in TAG behave a bit like (potentially discontinuous) chunks (cf. Sec. 4.2) whose potential occurrences can be pre-recognized before syntactic parsing, which opens up interesting possibilities regarding the MWE-aware parsing architecture. This point of view (which underlies the proposition of the MWE-promoting strategy in TAG, described below) is obscured in LFG by the fact that MWE-related descriptions come from simplex words, and potential MWE interpretations are constructed during the process of parsing, not before.

It is also worth pointing out that the treatment of MWEs as elementary grammar units entails a representation which can be seen as an instance of the bidirectional approach (with certain properties of the chunking approach),

which is the most flexible of the approaches we considered in Ch. 4, and the only one which allows to conveniently handle discontinuous MWEs with irregular structure. TAG provides a lot of flexibility in specifying the internal structure of various types of syntactically irregular MWEs, and allows to specify dedicated semantic representations for them. Additionally, information about the subcategorization requirements and morphosyntactic constraints are all specified directly in TAG elementary grammar units. TAG provides a unified treatment of subcategorization and MWEs which, in light of the observation that valency and MWEs are related notions (cf. Sec. 3.4), seems like the right choice.

Chapter 6

Tree adjoining grammar

This chapter contains a formal definition of a tree adjoining grammar (TAG). Sec. 6.2 contains relatively standard TAG-related definitions, while in Sec. 6.3, we provide definitions relevant to our TAG parsing architecture, described in more detail in Ch .7.

Recall that we focus on the TAG formalisms because it provides a first-class support for MWEs. MWEs can be conveniently represented in TAG grammars as elementary grammar units (Abeillé and Schabes, 1989). This makes it easy to account for the morphosyntactically restrictive properties of MWEs (such as additional agreement constraints), among others. Practical TAGs are typically fully (or almost fully) lexicalized. Lexicalized TAGs adopt no lexicon/grammar distinction, which allows to handle syntactically irregular MWEs without too much trouble. This and the previous point are related to what is typically called an *extended domain of locality* of TAGs (see Sec. 5.5.2). Finally, the TAG-based treatment of MWEs allows to see them as potentially discontinuous chunks which can be pre-recognized before syntactic parsing using various, supertagging-related techniques, which opens up interesting possibilities regarding the MWE-aware parsing architecture.

6.1 Prerequisites

Let us first establish the basic notation.

Definition 6 (sequence)**.** *Let X be a set and $n \in \mathbb{N}$. We define a sequence over X, denoted $x = (x_i \in X)_{i=1}^{n}$, $(x_1, x_2, \ldots, x_n)$, or $x_1 \ldots x_n$ for short, as a function from $\{1, \ldots, n\}$ to X, i.e., as a set $\{(1, x_1), (2, x_2), \ldots, (n, x_n)\}$,*

where n is the length of x. We also denote by $|x|$ the length n of the sequence x, by X^* the set of all sequences over X, and by ϵ an empty sequence.

Definition 7 (sequence concatenation)**.** *Let x, y be two sequences over X. Then, we write $x \cdot y$ to denote a concatenation of x and y, defined as $(x_1, \ldots, x_{|x|}, y_1, \ldots, y_{|y|})$. The concatenation is a binary associative operation, we will occasionally omit the symbol $\cdot$ and write simply xy to denote a concatenation of x and y.*

Since the same word can occur more than once in a sentence or a tree, we will often need to manipulate multisets of words.

Definition 8 (multiset)**.** *For a set X, a* multiset *over X is a set of pairs $\{(x, k) : x \in X\}$, where $k \in \mathbb{N}_{>0}$ is called the multiplicity of x. For any set X, we define $\mathcal{M}(X)$ as the set of all multisets over X.*

We extend set notations and operators to multisets. For instance, $\{(a, 2), (b, 1)\}$ is noted as $\{a, a, b\}_{ms}$, and we have $\{a, b\}_{ms} \cup \{a\}_{ms} = \{a, a, b\}_{ms}$, $\{a, a, b\}_{ms} \setminus \{a, b\}_{ms} = \{a\}_{ms}$, $\{a, b\}_{ms} \subseteq \{a, a, b\}_{ms}$, $\{a, a, b\}_{ms} \not\subseteq \{a, b\}_{ms}$, $|\{a, a, b\}_{ms}| = 3$, etc.

6.2 Definition of a TAG

Let Σ and N be disjoint sets of terminal and non-terminal symbols.

Definition 9 (initial tree (IT))**.** *An* initial tree *is an ordered tree with non-terminals in non-leaf nodes and terminals/non-terminals in leaf nodes.*[1]

Definition 10 (auxiliary tree (AT))**.** *An* auxiliary tree *is similar to an IT but it has one distinguished leaf (usually marked with an asterisk), called a* foot*, containing a non-terminal.*

Definition 11 (TAG tree)**.** *A* TAG tree *is a tree over terminal and non-terminal symbols which is either an initial tree or an auxiliary tree.*

[1]Leaf non-terminal nodes are conventionally decorated with a down arrow so as to emphasize that they can be replaced by other trees using a substitution operation. In this work, we omit this notational convention and assume that every leaf non-terminal node is marked for substitution, unless stated otherwise.

Traditionally, the above-defined terms *initial tree*, *auxiliary tree*, and *TAG tree*[2] usually refer to TAG trees present in a TAG grammar. In this work, we use them instead to refer to trees which can be constructed over the sets Σ and N in general.

Definition 12 (proper auxiliary tree)**.** *Let t be an AT over Σ and N. Then, we say that t is* proper *if it contains the same non-terminal $x \in N$ in its root and in its foot.*

Definition 13 (tree adjoining grammar (TAG) (Joshi and Schabes, 1997))**.** *A* tree adjoining grammar (TAG) *is defined as a tuple $G = \langle \Sigma_G, N_G, I_G, A_G, S_G \rangle$ where Σ_G is a set of terminal symbols, N_G is a set of non-terminal symbols, I_G is a set of initial trees over Σ_G and N_G, A_G is a set of proper auxiliary trees over Σ_G and N_G, and $S_G \in N_G$ is the start non-terminal.*

We will occasionally omit the subscript $_G$ (and refer to Σ_G by Σ, to N_G by N, etc.) when its choice is unambiguous. We call the trees belonging to I *elementary initial trees (EITs)*, the trees belonging to A *elementary auxiliary trees (EATs)*, and the trees belonging to $I \cup A$ *elementary trees (ETs)*.

Definition 14 (lexicalized TAG tree)**.** *A* lexicalized TAG tree *(an* LTAG tree *for short) is a TAG tree which contains at least one node labeled with a terminal. We call all such terminal nodes of a given tree its* anchors.

For instance, in Fig. 6.1, trees anchored with *minister*, *made*, and *prime minister* are ITs, while *the*, *prime*, and *good* are ATs. For the sake of clarity, we will refer to ETs by their anchors henceforth.[3] Moreover, in order to make it easier to refer to the individual ETs and their subtrees, we assign a unique ID to each node of each ET.

A *derived tree* is created from EITs and EATs by *substitution* and *adjunction*. Given an IT t, and any tree t', substitution replaces a non-terminal leaf l in t' by t provided that labels in l and in t's root are equal. Given a proper AT t, and any tree t', adjunction replaces t's foot by a subtree t'' of t' and then inserts this modified t in place of t'' in t', provided that the root non-terminals in t and t'' are identical. A *derivation tree* keeps track of the operations and the elementary trees (ETs) involved in the creation of a derived tree.

[2]TAG in TAG *tree* denotes the formalism, not a particular grammar.

[3]All the examples are based on lexicalized TAGs. While the parsing algorithm discussed in Ch. 7 can be used with TAGs in general, the heuristic we are going to see later on is constrained to lexicalized grammars.

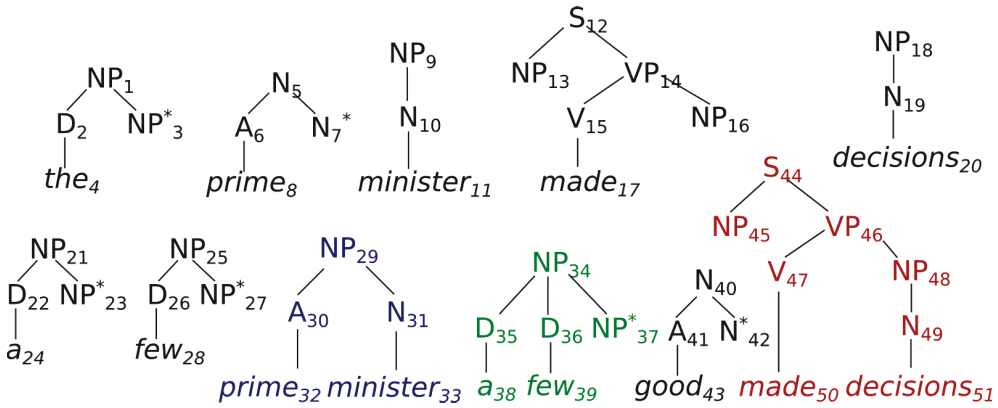


Figure 6.1: A toy TAG grammar. To each node a unique ID is assigned, placed in subscript on the right.

Definition 15 (derivation tree)**.** *Let G be a TAG. Then, we define a* derivation tree *as a rooted tree whose:*

- *nodes are labeled with G's ETs, and*
- *edges are labeled with addresses which unambiguously mark the attachment sites of the corresponding children ETs.*

Gorn addresses[4] are typically used to mark the attachment sites of the substitution and the adjunction operations. I.e., for each $\langle t_1, t_2 \rangle$ edge in the derivation tree marked with the address `addr`, `addr` points the node in the ET t_1 modified by t_2.

Fig. 6.2 illustrates a possible derivation which leads to the derived tree shown in Fig. 6.4. This derivation is based on the ETs present in the grammar shown in Fig. 6.1. A more traditional, but equivalent, way of representing the same TAG derivation tree is illustrated in Fig. 6.3 (b), while an alternative, compositional derivation (based on the same grammar) is illustrated in Fig. 6.3 (a).

Proposition 1. *Let δ be a derivation tree. Then, δ unambiguously determines the corresponding TAG tree, obtained by applying the individual substitution and adjunction operations over the δ's ETs.*

[4] 0 indicates the root node and ij indicates the j-th child of the node addressed with i.

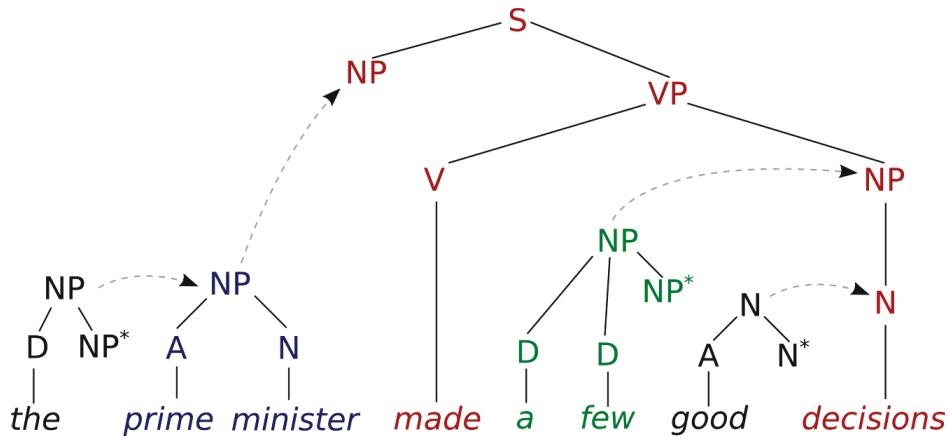


Figure 6.2: A derivation tree based on the grammar presented in Fig. 6.1. Adjunction and substitution operations are represented by dashed directed arcs leading to, respectively, internal nodes and leaf nodes in other ET trees.

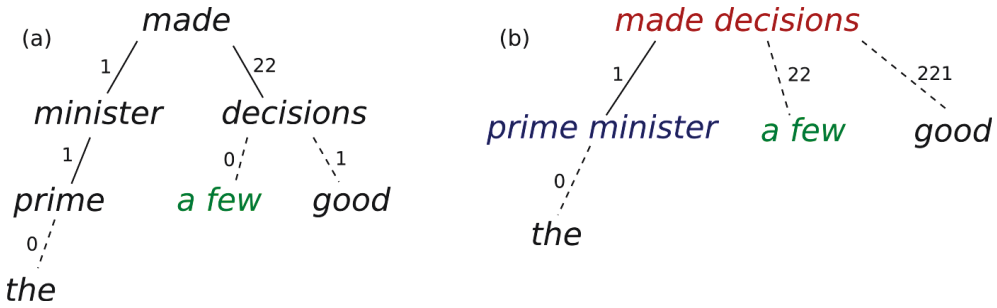


Figure 6.3: A traditional way of representing derivation trees. Nodes contain ETs (represented here by their anchors), while edges are labeled with Gorn addresses which indicate the modification sites. Substitutions and adjunctions are represented by plain and dashed edges, respectively. (b) A derivation tree corresponding to the derivation illustrated in Fig. 6.2. (a) An alternative derivation based on the same grammar.

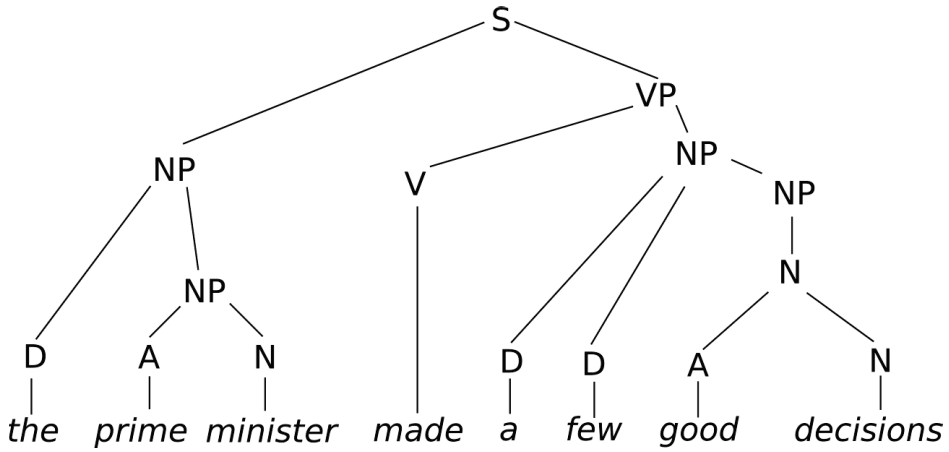


Figure 6.4: The derived tree corresponding to the derivation trees illustrated in Fig. 6.2 and Fig. 6.3 (b).

Note that the reverse does not necessarily hold, i.e., there can be several derivation trees corresponding to a given TAG tree.

Definition 16 (derived tree)**.** *Let δ be a derivation tree. We call the TAG tree determined by δ a* derived tree*.*

Definition 17 (complete tree)**.** *Let t be a TAG tree. Then, we say that t is complete, denoted $complete_G(t)$, iff all its leaves are labeled with terminals.*

Definition 18 (tree language)**.** *Let G be a TAG. Then, we define its* tree language*, denoted T_G, as the set of all S_G-rooted complete TAG initial trees which can be derived on the basis of the elementary trees in G.*

Definition 19 (yield of an initial tree)**.** *Let t be a TAG initial tree. Then, we define the* yield *of t, denoted $\gamma_G(t)$, as the left-to-right sequence of the terminals stored in t's terminal leaves.*

For instance, the yield of the tree illustrated in Fig. 6.4 is (the, prime, minister, made, a, few, good, decisions).

Definition 20 (yield of an auxiliary tree)**.** *Let t be an auxiliary tree. Then, we define the* yield *of t, denoted $\gamma_G(t)$, as a pair of sequences of terminals stored in t's terminal leaves on the left and on the right of the foot, respectively.*

Definition 21 (word language)**.** *Let G be a TAG. Then, we define its* word language*, denoted $L(G)$, as the set of the yields of the trees present in T_G. Formally:*

$$L(G) = \{\gamma_G(t) : t \in T_G\} \tag{6.1}$$

Given a TAG G and a sentence $s \in \Sigma_G^*$, the goal of parsing is, firstly, to determine whether $s \in L(G)$. Secondly, the parser should determine all the derived trees $t \in T_G$ (and, ideally, all the corresponding G-compliant derivation trees) such that $\gamma_G(t) = s$.

An important extension of TAG which we rely on in our work (particularly in Sec. 7.5.3) is a feature structures based TAG (FB-TAG). Restrictive morphosyntactic properties of MWEs can be most conveniently accounted for in TAGs using feature structures (Abeillé and Schabes, 1989). We refer the reader to (Vijay-Shanker and Joshi, 1988) for a detailed description of the FB-TAG formalism.

6.3 Custom definitions

Here, we present the TAG-related definitions particular to our work. Let us note that they are of no particular interest on their own, but rather serve to bridge the gap between the more traditional TAG-related definitions and the objects created by our parser (see Ch. 7). As such, these definitions might be easier to process and understand once the basics of the parser are acquired.

6.3.1 Elementary subtree

Definition 22 (child tree)**.** *Let t be a TAG tree, r be its root node, and c be a child of r*[5]*. Let also t' be a subtree of t in the mathematical sense*[6] *such that (i) the root of t' is c, and (ii) each node and edge reachable from c in t is also in t'. Then, we say that t' is a* child tree *of t.*

For instance, in Fig. 6.1, let t be the ET rooted in S_{44}, t_{VP} be the tree rooted in VP_{46}, and t_{NP} be the tree rooted in NP_{48}. Then, t_{VP} is a child tree of t, t_{NP} is a child tree of t_{VP}, but t_{NP} is not a child tree of t.

[5]Node c such that a directed edge leads from r to c (assuming that edges are oriented away from the root).

[6]The sets of nodes and edges in tree t' are subsets of the sets of nodes and edges in t, respectively.

Definition 23 (TAG subtree)**.** *Let t, t' be two TAG trees. Then, we say that t' is a* TAG subtree *(*subtree *for short) of t iff (i) t' is identical to t, or (ii) t' is a subtree of one of the children trees of t.*

Definition 24 (elementary subtree)**.** *Let G be a TAG and t be a TAG tree constructed over Σ and N. Then, we say that t is an* elementary subtree (EST) *in G iff a tree $t' \in I \cup A$ exists such that t is a subtree of t'.*

For instance, in Fig. 6.1, let t be the tree rooted in VP_{46} and t' be the tree rooted in NP_9. Then, both t and t' are elementary subtrees.

6.3.2 Derivation tree

We slightly modify the definition of a derivation tree so as to better reflect the correspondence between derivation trees and the items constructed in the process of bottom-up parsing.

Definition 25 (derivation tree)**.** *Let G be a TAG. Then, we define a* derivation tree *as a rooted tree such that:*

- *its root node is labeled with an EST and the non-root nodes are labeled with ETs,*
- *its edges are labeled with addresses which unambiguously mark the attachment sites of the corresponding children ETs,*[7] *and*
- *all non-terminal leaves of each EST in the derivation must be modified via substitution.*[8]

Thus, in contrast with Def. 15, (i) the root of a derivation tree has to contain an EST, but not necessarily an ET, and (ii) the derived tree corresponding to a given derivation tree is, by definition, complete. For instance, Fig. 6.5 (a) shows an example of a derivation tree consistent with Def. 15 but not with Def. 25, because the NP leaf of the root ET is not modified, while Fig. 6.5 (b) shows an example of a derivation tree consistent

[7] We adopt an extended model of TAG derivations, where an internal node can be adjoined to several times (Schabes and M. Shieber, 1994). We assume, henceforth, that children ETs attached to one and the same node are ordered.

[8] We abstract over the adjunction constraints, among others. Otherwise, we would also require here that all such constraints are satisfied.

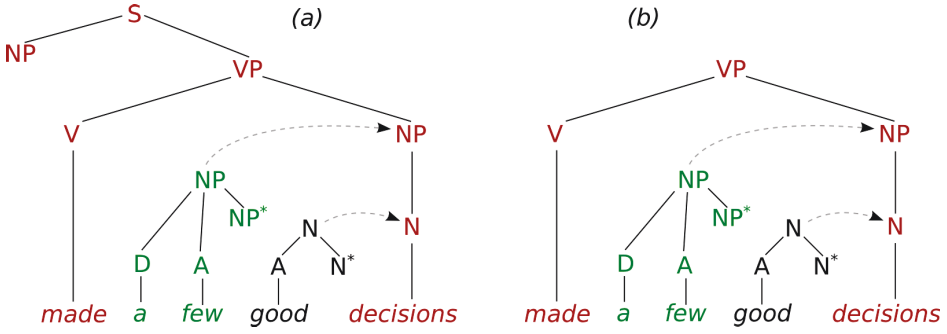


Figure 6.5: A comparison of a derivation tree (a) consistent with Def. 15 and a derivation tree (b) consistent with Def. 25.

with Def. 25 but not with Def. 15, because its root is not an ET. In the remaining of this work, we rely on Def. 25 rather than on Def. 15, unless stated otherwise.

Definition 26 (complete derivation tree)**.** *Let G be a TAG and δ be a G-compliant derivation tree. Then, we say that δ is a* complete derivation tree*, denoted $complete_G(\delta)$, iff the root of δ is an ET in G.*

For instance, the tree shown in Fig. 6.2 and both trees shown in Fig. 6.3 are complete derivation trees, while the tree shown in Fig. 6.5 (b) is not complete.

Proposition 2. *Let G be a TAG and $t \in T_G$ be a derived tree. Then,*

- *the set of derivation trees complying with Def. 15 and yielding the derived tree t, and*
- *the set of complete derivation trees complying with Def. 25 and yielding the derived tree t,*

are identical.

Definition 27 (auxiliary derivation tree)**.** *Let δ be a derivation tree whose root EST t is auxiliary. Then, we say that δ is* auxiliary *and we define the* foot *of δ as the t's foot.*

Definition 28 (yield of a derivation tree)**.** *Let δ be a derivation tree. We extend the definition of the* yield *to derivation trees, denoted $\gamma(\delta)$, as the yield (either a sequence or a pair of sequences) of the corresponding derived tree.*

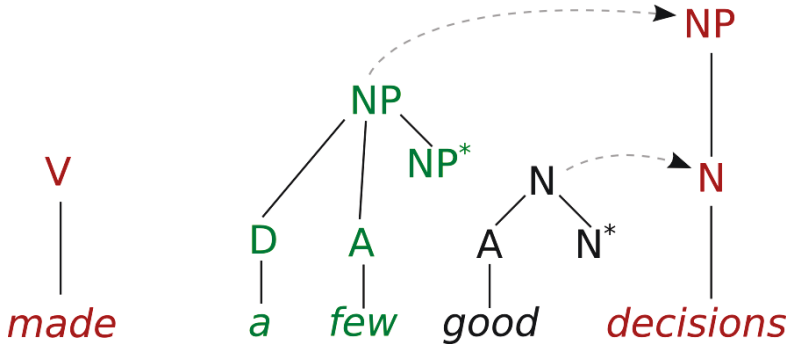


Figure 6.6: An example of a derivation grove consisting of two derivation trees. The left one contains one EST t anchored in *made*, while the right one, δ, is composed from three ESTs. t and the root of δ are adjacent children trees of the VP *made decisions* EST.

Definition 29 (set of derivations)**.** *Let G be a TAG. Then, we define Δ_G as the set of all the G-compliant derivation trees. For a given sequence of words $s \in \Sigma_G^*$, we also define $\Delta_G(s) \subseteq \Delta_G$ as the set of all derivation trees δ such that $\gamma(\delta) = s$, and $\Delta_G^{comp}(s) = \{\delta \in \Delta_G(s) : complete_G(\delta)\}$.*

Definition 30 (size of a TAG derivation tree)**.** *Let G be a TAG and $\delta \in \Delta_G$. Then, we define the* size *of δ, denoted $|\delta|$, as the number of δ's nodes.*

The size of a derivation tree corresponds, therefore, to the number of ESTs participating in this derivation.

6.3.3 Derivation grove

Definition 31 (derivation grove)**.** *Let δ be a sequence of derivation trees. Then, we say that δ is a* derivation grove *iff the individual derivation trees δ_i contain, in their roots, ESTs which are adjacent children of an EST t. Formally:*

- *δ_1's root must be the left-most child subtree of t, and*
- *for each $i = 2 \ldots |\delta|$, δ_i's root must be a child subtree of t placed immediately on the right of the δ_{i-1}'s root.*

Fig. 6.6 shows an example of a derivation grove consisting of two derivation trees.

Proposition 3. *Let δ be a derivation grove. Then, at most one of its derivation trees δ_i is auxiliary.*

The above proposition holds because, otherwise, an ET with more than one foot node would have to exist.

Definition 32 (auxiliary derivation grove)**.** *Let δ be a derivation grove which contains an auxiliary derivation tree δ_i. Then, we say that δ is* auxiliary *and we define the* foot *of δ as the δ_i's foot.*

Definition 33 (yield of a derivation grove)**.** *Let δ be a derivation grove. We extend the definition of the* yield *to derivation groves, denoted $\gamma(\delta)$, as the left-to-right concatenation of the yields $\gamma(\delta_i)$ of the individual derivation trees δ_i based on the following, associative binary operation which allows to append two yields:*

$$x \circ y = \begin{cases} \langle x_1, x_2 y \rangle, \text{ if } \langle x_1, x_2 \rangle = x \text{ and } y \in \Sigma_G^* \\ \langle xy_1, y_2 \rangle, \text{ if } \langle y_1, y_2 \rangle = y \text{ and } x \in \Sigma_G^* \\ xy, \text{ if both } x, y \in \Sigma_G^*, \\ \text{undefined, otherwise.} \end{cases}$$

Note that in the above definition, the fourth case (both yields are pairs) cannot occur because, otherwise, an EST would have to exist which contains more than one foot node.

Occasionally, we will refer to both derivation trees and derivation groves as *derivations* for short.

6.3.4 Derivation chain

Definition 34 (derivation chain)**.** *Let $\vec{\delta}$ be a non-empty sequence of derivation trees. Then, we say that $\vec{\delta}$ is a* derivation chain *iff:*

- *$\vec{\delta}_1$'s root is an initial EST,*
- *$\forall_{2<i\leq|\vec{\delta}|}$, $\vec{\delta}_i$'s root is an auxiliary EST (not necessarily proper) whose foot node is decorated with the same non-terminal as the root of the $\vec{\delta}_{i-1}$'s root EST.*

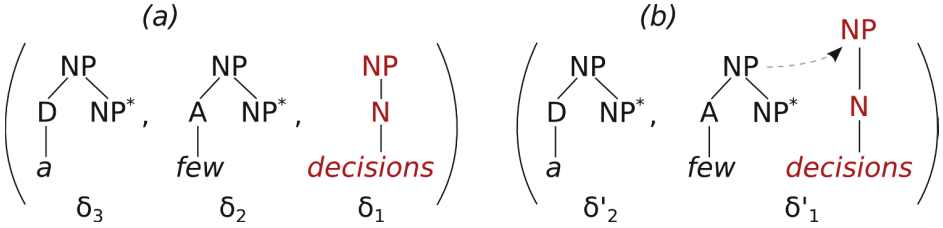


Figure 6.7: A graphical representation of two derivation chains, δ and δ'. Note that the order of elements is inverted in both chains.

Fig. 6.7 illustrates two examples of derivation chains. Fig. 6.7 (a) represents a derivation chain consisting of three singleton derivation trees, while (b) represents a chain consisting of two derivation trees. Derivation chains play an important role in the parsing algorithm which will be covered in Sec. 7.3.

Definition 35 (tip of a derivation chain)**.** *Let $\vec{\delta}$ be a derivation chain. Then, we define its* tip *as the root node of $\vec{\delta}_{|\vec{\delta}|}$'s root EST.*

Derivation chains are related to derivation trees. For instance, the two derivation trees of the derivation chain illustrated in Fig. 6.7 (b) could be combined using adjunction, which would result in a single derivation tree composed of two ETs and one EST. However, a transformation from a derivation chain to the corresponding derivation tree is, in general, not always possible. Firstly because the individual ATs present in a derivation chain are not necessarily elementary and only fully recognized EATs can be adjoined. Secondly because, if unification-like computations are assigned to the individual ETs, it may be that the derivation chain can be only trasformed into the corresponding derivation tree if it is extended with an additional derivation tree.[9] It is also worth noting that from a given derivation chain several derivation trees can be possibly constructed. For instance, in Fig. 6.7 (a), the left-most ET (i.e., the root of δ_3) can be adjoined either to the root node of the ET in δ_2, or to the root node of the EST in δ_1.

Definition 36 (yield of a derivation chain)**.** *Let $\vec{\delta}$ be a derivation chain. Then,*

[9]For instance if the top and the bottom FSs assigned to the root node of the root EST of the last derivation tree in a given chain are inconsistent and can be thus reconciled only if the root node is modified via adjunction.

we define the yield *of* $\vec{\delta}$*, denoted* $\gamma(\vec{\delta})$*, according to the following equation:*

$$\gamma(\vec{\delta}) = \begin{cases} \gamma(\vec{\delta}_1), \text{ if } |\vec{\delta}| = 1 \\ x \cdot \gamma(\vec{\delta}') \cdot z, \text{ otherwise, where } \vec{\delta} = \vec{\delta}' \cdot (\delta) \wedge \langle x, z \rangle = \gamma(\delta). \end{cases}$$

For instance, the yield of both derivation chains illustrated in Fig. 6.7 is equal to (a, few, decisions).

6.3.5 Multiset of terminals

Definition 37 (ET terminals)**.** *Let* t *be a TAG tree. Then, we define* $sub(t) \in \mathcal{M}(\Sigma)$ *as the multiset of terminals in tree* t*.*

For instance, in Fig. 6.1, let t be the ET *made decisions*, t_1 be its subtree rooted at NP_{45}, and t_2 be its subtree rooted at NP_{48}. Then, $sub(t) = \{\text{decisions}, \text{made}\}_{ms}$, $sub(t_1) = \emptyset_{ms}$, and $sub(t_2) = \{\text{decisions}\}_{ms}$.

Chapter 7

Parsing MWEs with TAGs

The first issue we try to tackle in this chapter is related to TAG parsing efficiency in general. Real-world, wide-coverage TAG grammars tend to be large, which leads to efficiency-related issues. The theoretical time complexity of TAG parsing is $\mathcal{O}(n^6)$, where n is the length of the input sentence (Satta, 1994). However, TAG parsing is also linear with respect to the size of the input grammar N. Clearly, parsing can become too slow for the use in practical NLP applications when either n or N becomes too big. While the length of the sentence theoretically plays a much more important role, sentences are typically much smaller than the grammar, thus the impact of N on parsing speed is not negligible either.

The TAG parsing architecture we describe in Sec. 7.3 lends itself to a variety of grammar compression techniques, detailed in Sec. 7.5.2, which enable parsing speed-up improvements. Even though this issue is not directly related to MWEs, adding MWE-related entries to a grammar clearly increases its size, even if it is not clear whether this phenomenon can, by itself, change significantly the scale of the problem.

The second issue is more specific to MWEs. Practical hybrid parsing systems often adopt a parsing architecture in which disambiguation is performed in parallel with syntactic analysis, in the sense that it is not required that all the valid (with respect to the underlying grammar) syntactic derivations are computed before disambiguation can take place (Klein and Manning, 2003; Angelov and Ljunglöf, 2014; Lewis and Steedman, 2014). The advantage is that a significant portion of the syntactic analysis search space can be left unexplored if only the best (or a couple of the best) syntactic derivations are searched for.

This strategy led to particularly encouraging results published in the work of Lewis and Steedman (2014), who used it within the context of CCG parsing. However, Lewis and Steedman (2014)'s proposal, even though quite generic, cannot be used within the context of a TAG grammar in which MWEs are represented as multi-anchored ETs, nor with any other strongly lexicalised symbolic grammar in which MWEs are represented as elementary grammar units. At the same time, Wehrli (2014) showed that promoting collocation-based derivations in symbolic parsing – MWEs are, in the vast majority, statistical collocations – is a potentially beneficial strategy in that it helps to deal with syntactic ambiguity. The second task we undertake in this chapter is, thus, to implement a MWE-promoting strategy (defined in Sec. 7.1) within the framework of A* TAG parsing (see Sec. 7.4). To this end, we define (in Sec. 7.4.3) a heuristic which allows to estimate the cost of parsing a given sentence fragment based on the potential MWE occurrences therein. In Sec. 7.5.1, we define a variant of this heuristic which handles adjunction more thoroughly and which has better theoretical properties. In Ch. 8, we experimentally verify the impact of the MWE-promoting strategy, applied to several types of MWEs, on LTAG parsing accuracy and speed.

Finally, certain idiosyncratic properties of MWEs – notably, restrictive morphosyntactic constraints – can be most conveniently accounted for in TAGs using feature structures (Abeillé and Schabes, 1989). Regardless of MWEs, feature structure decorations are also used in wide-coverage TAG grammars (Alahverdzhieva, 2008) and allow to formalize compositional semantics over TAG derivation trees (Gardent and Kallmeyer, 2003). In Sec. 7.5.3, we show how to extend the main parser implemented in ParTAGe so as to account for feature structure decorations and, more generally, for various unification-like computations defined over ETs.

It is worth noting another family of methods which allow to deal with the efficiency problems in TAG parsing. Such methods consist in constraining the formal power of TAG – and, thus, reducing the parsing complexity – while preserving most of the properties which make TAG relevant for linguistic modeling.[1] A well-known example of this approach is the *tree insertion*

[1]Another form of restrictions can be found in *spinal TAGs* (Shen and Joshi, 2005), where each elementary tree has to have the so-called *spinal* form. The goal in this case is not to reduce the parsing complexity, but rather to make the process of constructing TAG ETs take place in parallel with parsing. It is not clear to us, however, to what extent such mechanism could be relevant to MWEs and, more generally, to parsing heavily relying on linguistic resources.

grammar (Schabes and C. Waters, 1995), where wrapping adjunction is not allowed, thus decreasing the parsing complexity to $\mathcal{O}(n^3)$. Another formalism implementing this principle is osTAG (Swanson et al., 2013). The advantage of A* parsing over these methods is that A* does not require limiting the formal power of TAG in any way, yet potentially leading to comparable improvements.[2] It should be also possible to implement A* parsing for the two formalisms mentioned above, thus combinig the characteristics of both optimization families.

7.1 MWE-promoting strategy

The strategy of promoting MWEs can be formalized in TAGs in terms of a (partial) disambiguation strategy which selects those TAG derivation trees which contain MWE ETs. Consider again the derivation tree δ_{mwe} shown in Fig. 6.2 and an alternative, more compositional derivation δ shown in Fig. 7.1. Both trees are based on the same grammar (cf. Fig. 6.1), but δ_{mwe} contains more MWE ETs than δ and, because MWE ETs are multi-anchored, δ_{mwe}'s size is smaller ($|\delta_{mwe}| = 5 \wedge |\delta| = 7$). In order to make the parser promote the TAG derivations based on MWE ETs, it is thus sufficient to make it select the derivations which contain a smaller number of participating ETs.

Definition 38 (promoting MWEs)**.** *Let G be a TAG and $s \in \Sigma_G^*$ be a sentence. Then, we define a* MWE-promoting strategy *as the relation $\leq$ over $\Delta_G^{comp}(s)$ such that:*

$$\delta_1 \leq \delta_2 \iff |\delta_1| \leq |\delta_2|. \tag{7.1}$$

7.2 Probabilistic characterization

Since the MWE-promoting strategy can be seen as a partial disambiguation function, it can be expressed in probabilistic terms, i.e., in terms of a weighted TAG.

Definition 39 (weighted TAG)**.** *We define a* weighted TAG *as a pair $\langle G, \omega_G \rangle$ such that G is a TAG and ω_G is a weighting function from $I_G \cup A_G$ to $\mathbb{R}$.*

[2]We estimate that, under favorable circumstances – $\mathcal{O}(1)$ number of the optimal derivations per sentence, perfect estimations of the heuristic – and assuming (quite safely) that the size of a derivation in LTAGs is $\mathcal{O}(n)$, A* can reduce the complexity of TAG parsing to $\mathcal{O}(n^2)$.

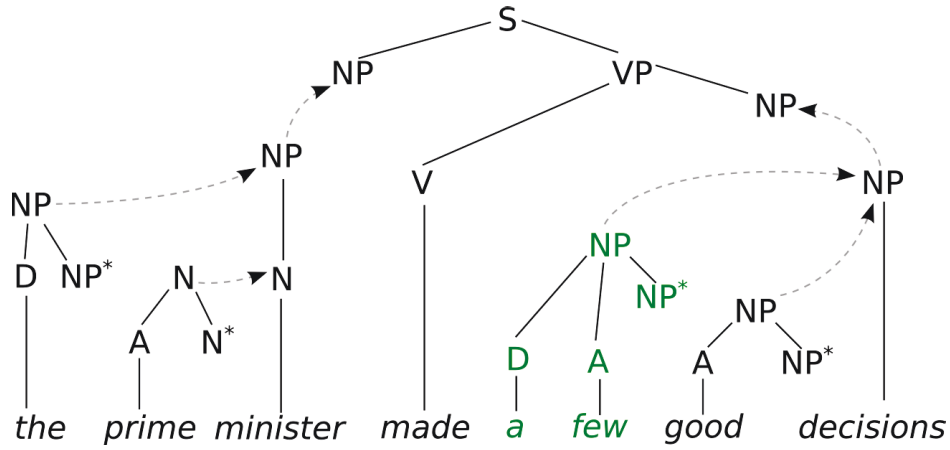


Figure 7.1: A "compositional" derivation tree based on the grammar presented in Fig. 6.1.

The smaller the number $\omega_G(t)$ assigned to a particular ET t in G, the more positive the impact of the presence of t in a derivation tree is. We implicitly extend to weighted TAGs the definitions applying to regular TAGs.

Such a characterization is inspired by the work of Lewis and Steedman (2014) who used it for CCG parsing with promising results. However, in Lewis and Steedman (2014)'s work, elementary grammar units can only analyse simplex words. This means, on the one hand, that they cannot be used to model MWEs in the same way as in TAG, but also that the probability of a derivation can be defined as a product of the probabilities of the elementary grammar units attached to the individual words in the sentence.

Definition 40 (multiset of ETs in a derivation)**.** *Let G be a TAG and $\delta \in \Delta_G$. Then, we define $mts(\delta)$ as the multiset of ETs participating in δ.*

In our case, ETs can analyse several words at once, as the *made decisions* ET in Fig. 6.2, which is attached to two non-adjacent words in the input sentence. In such a context, a probability of a derivation can be defined in a log-linear probabilistic framework instead:

Definition 41 (probability of a derivation)**.** *Let $\langle G, \omega_G \rangle$ be a weighted TAG, s be a sentence, and $\delta \in \Delta_G^{comp}(s)$ be a complete derivation tree. We define the probability of δ as:*

$$p_G(\delta|s) = \frac{\exp(\sum_{\langle t,k\rangle \in mts(\delta)}(-\omega_G(t) \cdot k))}{\sum_{\delta' \in \Delta_G^{comp}(s)} \exp(\sum_{\langle t,k\rangle \in mts(\delta')}(-\omega_G(t) \cdot k))}. \tag{7.2}$$

The above probability is well-defined only if the set $\Delta_G(s)$ is finite.

Definition 42 (finiteness of the derivation set in LTAG)**.** *Let G be an LTAG and s be a sentence. Then, $\Delta_G(s)$ is finite (Kallmeyer, 2010, p. 20).*

Definition 43 (weight of a derivation)**.** *Let $\langle G, \omega_G \rangle$ be a weighted TAG and $\delta \in \Delta_G$ be a derivation. Then, we define the weight of δ as:*

$$\omega_G(\delta) = \sum\nolimits_{\langle t,k \rangle \in mts(\delta)} \omega_G(t) \cdot k \cdot [t \in I_G \cup A_G]$$

where [] is the Iverson bracket.[3]

We do not adopt any notational distinction between $\omega_G \colon (I \cup A) \to \mathbb{R}$ and its extension $\omega_G \colon \Delta_G \to \mathbb{R}$.

The value of the denominator in Eq. 7.2 does not depend on the given derivation, thus it can be ignored when searching for the most probable derivation for a given sentence.

Definition 44 (optimal derivation)**.** *Let $\langle G, \omega_G \rangle$ be a weighted TAG and s be a sentence. Then, the most probable,* optimal derivation *of the sentence s can be determined using the following equation:*

$$\hat{\delta}(s) = \arg\min\nolimits_{\delta \in \Delta_G^{comp}(s)} \omega_G(\delta) \tag{7.3}$$

Finally, the MWE-promoting strategy can be implemented in such a weighted framework by assuming that the same, non-negative weight w_0 (e.g., 1) is assigned to each ET in the given grammar.

Proposition 4. *Let $\langle G, \omega_G \rangle$ be a weighted TAG such that $\omega_G(t) = 1$ for each $t \in I_G \cup A_G$, s be a sentence, and $\delta \in \Delta_G^{comp}(s)$. Then, $\omega_G(\delta) = |\delta|$ and, therefore, ω_G straightforwardly specifies the MWE-promoting strategy:*

$$\delta_1 \leq \delta_2 \iff \omega_G(\delta_1) \leq \omega_G(\delta_2) \tag{7.4}$$

Note that ω_G can be further adjusted with respect to a given input sentence using supertagging methods, given an appropriate probabilistic supertagging model together with an accompanying training data set.

[3] $[x] = 1$ if $x =$ TRUE and $[x] = 0$ otherwise.

7.3 ParTAGe: TAG parsing architecture

ParTAGe assumes a particular representation of TAG grammars which consists of two parts. Each TAG is first transformed into an equivalent, directed acyclic graph (DAG) representation with ordering of outgoing edges for each node. Afterwards, traversals of the individual ET trees and their subtrees are represented as paths in finite-state automata (FSAs).

7.3.1 Grammar DAG

Definition 45 (out-ordered directed graph)**.** *We define an* out-ordered directed graph *(*out-ordered graph *for short) as a tuple* $D = \langle V_D, E_D \rangle$ *where:*

- V_D *is the set of vertices.*
- $E_D \colon V_D \to V_D^*$ *represents the parent-child relations between the vertices and is defined for each* $v \in V_D$*. Thus, each element of* E_D *encodes a set of edges connecting a parent with its ordered children.*

Definition 46 (child)**.** *Let* D *be an out-ordered graph and* $v, w \in V_D$*. We say that* v *is a* child *of* w*, denoted* $child_D(v, w)$*, iff* $\exists_{1 \leq i \leq |E_D(w)|}(E_D(w)_i = v)$*.*

Henceforth, whenever the out-ordered graph D is unambiguously identifiable, we will refer to $V_D, E_D, child_D$, etc., as V, E, *child*, etc., for short.

Definition 47 (path)**.** *Let* D *be an out-ordered graph and* $v_1 \dots v_k \in V^+$*. Then, we say that* $v_1 \dots v_k$ *is a path in* D *connecting* v_1 *with* v_k *iff*

$$\forall_{1 \leq i \leq k-1}(child(v_{i+1}, v_i))$$

Definition 48 (out-ordered DAG)**.** *Let* D *be an out-ordered graph. Then, we say that* D *is acyclic (and hence* D *is an* out-ordered DAG*) iff, for any* $v \in V$*, there is no path in* D *longer than* 1 *which would connect* v *with itself.*

Henceforth, we will be only concerned with out-ordered DAGs, even if some of the following definitions apply to out-ordered graphs in general.

Definition 49 (root)**.** *Let* D *be an out-ordered DAG. We define* $root_D \colon V \to \mathbb{B}$ *as a function which tells whether a given vertex is a root in* D *or not.*

$$root_D(v) = \begin{cases} \text{FALSE}, & \text{if } \exists_{v' \in V}\, child(v, v') \\ \text{TRUE}, & \text{otherwise.} \end{cases}$$

Definition 50 (single-rooted DAG)**.** *Let D be an out-ordered DAG. Then, we say that D is* single-rooted *iff* $\exists!_{v \in V} root(v)$.

Definition 51 (leaf)**.** *Let D be an out-ordered DAG. We define $leaf_D \colon V_D \to \mathbb{B}$ as a function which tells whether a given vertex v is a leaf in D (i.e., $E(v) = \epsilon$) or not.*

Definition 52 (sub-DAG)**.** *Let D and D' be two out-ordered DAGs. Then, we say that D' is a* sub-DAG *of D, denoted $D' \subseteq D$, iff $V_{D'} \subseteq V_D$ and $E_{D'} \subseteq E_D$.*

Note that if D' is a sub-DAG of D and if it contains a given vertex v from V_D, then it also contains all the vertices v' reachable from v in D. This is because if D' contains a vertex $v \in V_D$, then it must (by definition) contain an edge leading from v to its children vertices, and there is only one such candidate edge $\langle v, c \rangle \in E_D$. By definition, it must therefore contain all the children nodes $(c_i)_{i=1}^{|c|}$ as well and, by extension, all the nodes and edges reachable from v via the parent-child relation.

Definition 53 (rooted sub-DAG)**.** *Let D and D' be two out-ordered DAGs. Then, we say that D' is a* sub-DAG of D rooted in $v \in V_D$ *iff $D' \subseteq D$, D' is single-rooted and $root_{D'}(v)$.*

Proposition 5 (subdag function)**.** *Let D be an out-ordered DAG and $v \in V_D$. Then, there exists exactly one out-ordered DAG D' such that D' is a sub-DAG of D rooted in v. We denote such a sub-DAG as $subdag_D(v)$.*

Definition 54 (grammar DAG)**.** *We define a* grammar DAG (GDAG) *as a tuple $D = \langle V_D, E_D, \Sigma_D, N_D, S_D, \ell_D, foot_D \rangle$ such that:*

- *$D_0 = \langle V_D, E_D \rangle$ is an out-ordered DAG.*
- *Σ_D and N_D are disjoint sets of terminals and non-terminals, respectively, and $S_D \in N_D$ is the start symbol.*
- *$\ell_D \colon V_D \to \Sigma_D \cup N_D$ is a function which assigns the non-terminal and terminal values to the individual vertices in the grammar. Moreover, $\forall_{v \in V_D}(\ell_D(v) \in \Sigma_D \implies leaf_{D_0}(v))$.*
- *$foot_D \colon V_D \to \mathbb{B}$ tells whether a given vertex is a foot node or not. Moreover, it holds that $foot_D(v) \implies leaf_{D_0}(v) \wedge \ell_D(v) \in N_D$.*

- *For each vertex* $r \in V_D$: $root_{D_0}(r)$ *and the corresponding* $D' = subdag_{D_0}(r)$, *there exists at most one* $v \in V_{D'}$ *such that* $foot_D(v)$. *Moreover, if such a vertex* $v \in V_{D'}$ *exists, than* $\ell_D(r) = \ell_D(v)$.

Thus a GDAG can be seen as a fusion of the individual ETs of a given TAG grammar in a single data structure. We implicitly extend the definitions introduced in the previous subsection (child, root, leaf, subdag, etc.) to grammar DAGs in the natural way. As with out-ordered graphs, we will omit the subscript $_D$ whenever D is unambiguously identifiable and write V, E, *child*, etc., instead of $V_D, E_D, child_D$, etc.

Proposition 6 (GDAG's vertex ⇒ TAG tree correspondence)**.** *Let* D_0 *be a GDAG,* v *be a vertex in* D_0, *and* $D = subdag_{D_0}(v)$. *Then,* D *unambiguously specifies a TAG tree* t *over non-terminals* N_D *and terminals* Σ_D. *Moreover,* t *is auxiliary if* D *contains a foot node, and initial otherwise. In case* v *is a root in* D_0 *and* t *is auxiliary,* t *is also a proper auxiliary tree.*

The TAG tree corresponding to a given root vertex in D can be determined in a recursive manner. For a given vertex $v \in V$, the TAG subtrees corresponding to the individual children in $E(v)$ are identified first, while the (non-)terminal attached to the node corresponding to v itself is determined on the basis of $\ell(v)$. Finally, the status of the node as a foot is determined based on $foot(v)$.

Proposition 7 (GDAG ⇒ TAG correspondence)**.** *Let* D *be a GDAG. Then,* D *uniquely determines the corresponding TAG* $G = \langle \Sigma_D, N_D, I, A, S_D \rangle$, *where the elementary trees in* I *and* A *are obtained by the traversals of the sub-DAGs rooted in the individual root vertices in* D.

Definition 55 (DAG encoding)**.** *Let* D *be a GDAG and* G *be the corresponding TAG. Then, we call* D *a* DAG encoding *of* G.

Proposition 8. *Let* G *be a TAG and* D *be its DAG encoding. Then, each vertex* $v \in V$ *corresponds to an EST in* G. *More precisely,* v *entails the corresponding* $subdag(v)$ *whose traversal leads to an EST, denoted* $est_D(v)$. *If* $root(v)$, *then* $est(v)$ *is also an ET in* G.

Note that it is not true that each TAG uniquely identifies the corresponding GDAG. Many different GDAGs can be constructed which represent one and the same TAG. Fig. 7.2 shows two possible DAG encodings of a fragment of

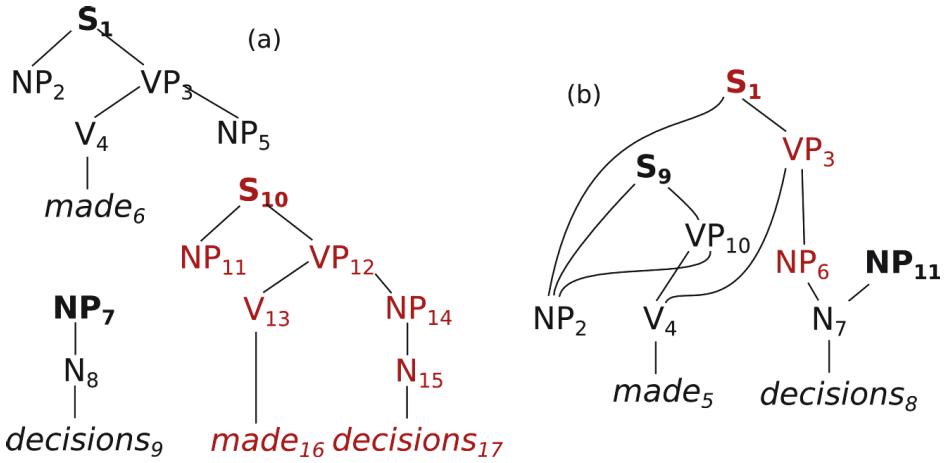


Figure 7.2: Two possible DAG encodings of the ETs *made*, *decisions*, and *made decisions* from Fig. 6.1. For the sake of clarity, the values of ℓ are put in place of the corresponding vertices, while the vertex identifiers are placed in subscript on the right. The roots of the individual ETs are marked in bold.

the TAG presented in Fig. 6.1. In particular, common ESTs in Fig. 7.2 (b) are shared among the individual ETs. This technique is useful for the sake of computation sharing, especially in bottom-up parsing. It was already used by Schabes and C. Waters (1995) within the context of *tree insertion grammars* (a TAG-related formalism where wrapping adjunction is not allowed, thus decreasing the parsing complexity to $\mathcal{O}(n^3)$) and can be easily extended to TAGs.

Def. 55 does not exclude the possibility of representing a given ET by several roots in a GDAG either. This can be useful if to the individual ETs additional computations, e.g. unification over feature structures assigned to the individual nodes, are assigned.

The following definition provides a generic encoding method which, for any given TAG, allows to construct the corresponding GDAG in which each source ET is represented by a distinct subdag.

Definition 56 (simple DAG encoding)**.** *Let G be a TAG and D be a grammar DAG. Then, we say that D is a* simple DAG encoding *of G iff:*

- $\Sigma_D = \Sigma_G$, $N_D = N_G$, *and* $S_D = S_G$.
- *Every node of every tree in $I_G \cup A_G$ is represented as a distinct $v \in V_D$.*

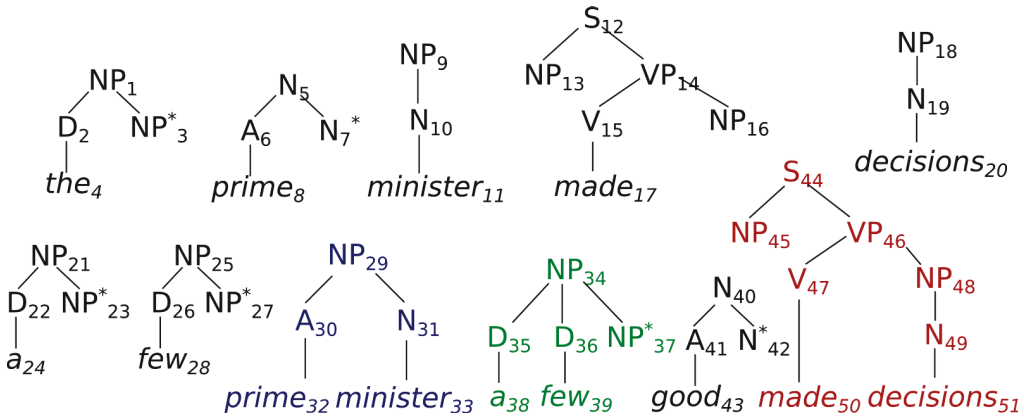


Figure 7.3: A simple DAG encoding (a copy of Fig. 6.1).

- *E_D directly encodes the parent-child relations present in $I_G \cup A_G$,*
- *ℓ_D and $foot_D$ are defined correspondingly.*

We also say that D is a simple DAG encoding *iff it is a simple DAG encoding of the TAG it entails.*

Fig. 7.3 illustrates an example of a simple DAG encoding. In fact, it constitues a copy of Fig. 6.1, but the meaning of the node IDs changes slightly. Formally, this encoding, restricted to the ETs *the* and *minister*, is: $V = \{1, 2, 3, 4, 9, 10, 11\}$, $S = \mathrm{S}$, $\ell = \{\langle 1, \mathrm{NP}\rangle, \langle 2, \mathrm{D}\rangle, \langle 3, \mathrm{NP}\rangle, \langle 4, \mathrm{the}\rangle, \langle 9, \mathrm{NP}\rangle, \langle 10, \mathrm{N}\rangle, \langle 11, \mathrm{minister}\rangle\}$, $foot = \{\langle 3, \textsc{true}\rangle\} \cup \{\langle v, \textsc{false}\rangle : v \in V \setminus \{3\}\}$, and $E = \{\langle 1, (2, 3)\rangle, \langle 2, (4)\rangle, \langle 3, \epsilon\rangle, \langle 4, \epsilon\rangle, \langle 9, (10)\rangle, \langle 10, (11)\rangle, \langle 11, \epsilon\rangle\}$.

Definition 57 (tree^1)**.** *Let G be a TAG and D be its simple DAG encoding. We define $\mathrm{tree}^1_D \colon V_D \to V_D$ as a function which tells, for a given vertex v, to which of D's root (and, by extension, ET in G) it corresponds.*

For instance, in Fig. 7.3, $tree^1(4) = 1$ and $est(tree^1(4))$ corresponds to the ET *the*, $tree^1(8) = 5$ and $est(tree^1(8))$ corresponds to the ET *prime*, $tree^1(45) = 44$ and $est(tree^1(45))$ corresponds to the ET *made decisions*, etc.

Note that we use a special index ^{1} to indicate that the result of this function is a single ET. In Sec. 7.5.2, we consider other TAG $\to$ DAG encodings, in which a given DAG vertex can belong to several ETs. In the light of this possibility, we define its generalized version.

Definition 58 (tree)**.** *Let D be a GDAG. Then, we define* $\text{tree}_D \colon V_D \to 2^{V_D}$ *as a function which tells, for a given vertex v, to which of D's roots it corresponds.*

For instance, in Fig. 7.2 (b), $tree(10) = \{9\}$, $tree(4) = \{1, 9\}$, $tree(7) = \{1, 11\}$, etc.

Proposition 9. *Let D be a GDAG and $r \in V_D \colon root_D(r)$. Then, it holds that $tree_D(r) = \{r\}$.*

Definition 59 (sub terminals)**.** *Let D be a GDAG and $v \in V$. Then, we define $sub_D(v) \in \mathcal{M}(\Sigma)$ as the multiset of terminals occurring in $est(v)$, i.e., in $sub(est(v))$.*

Note that *sub* is an extension of Def. 37 in that it applies to DAG nodes rather than to TAG trees.

Definition 60 (super^1 terminals)**.** *Let D be a simple DAG encoding and $v \in V$. Then, we define $super^1_D(v) \in \mathcal{M}(\Sigma)$ as the multiset of terminals occurring in $est(tree^1(v))$ outside of $est(v)$.*

Proposition 10. *Let D be a simple DAG encoding and $v \in V$. Then:*

$$super^1(v) = sub(tree^1(v)) \setminus sub(v).$$

Proposition 11. *Let D be a simple DAG encoding and $v \in V \colon root(v)$. Then, it holds that $super^1(v) = \emptyset_{ms}$.*

For instance, in Fig. 7.3, $sub(2) = \{\text{the}\}_{ms}$, $sub(3) = \emptyset_{ms}$, $sub(34) = \{\text{a}, \text{few}\}_{ms}$, $sub(47) = \{\text{made}\}_{ms}$, etc., and $super^1(1) = \emptyset_{ms}$, $super^1(16) = \{\text{made}\}_{ms}$, $super^1(45) = \{\text{made}, \text{decisions}\}_{ms}$, $super^1(47) = \{\text{decisions}\}_{ms}$, etc.

7.3.2 Grammar FSAs

The parser relies furthermore on an encoding of the possible traversals of ETs. Such traversals are represented in the form of paths in the corresponding finite-state automata (FSAs). The motivation behid such a representation is to reduce the space of possible chart items and, hence, to further increase computation sharing. A similar solution can be found in (Nederhof, 1998), where an LR automaton is used to represent the space of possible chart

items, grouped together into equivalence classes closed under prediction, and the possible transitions between such classes. The grammar representation described below can be seen as a specialization of the LR automaton to bottom-up parsing.

Definition 61 (FSA family)**.** *We define an* FSA family *as a tuple* $M = \langle Q_M, V_M, \delta_M, S_M, heads_M \rangle$ *such that:*

- Q_M *is the set of FSA-like states and* V_M *is the set of alphabet symbols,*
- $\delta_M \colon Q_M \times V_M \to Q_M$ *is the transition function,*
- $S_M \subseteq Q_M$ *is the set of start states, and*
- heads_M *is a function* $: Q_M \to 2^{V_M}$ *which returns the final symbols outgoing from a given state.*

An FSA family is similar to a regular finite-state automaton over alphabet V_M, but (i) it can contain several start states, and (ii) instead of final states, the $heads_M$ function specifies the states from which final quasi-transitions leave, as well as the corresponding alphabet symbols. We adopt a distinction between symbols emitted by the regular transitions and those emitted by the final quasi-transitions because we will use them for different purposes. An FSA family can be thus also seen as a transducer whose input alphabet and output alphabet are both subsets of V_D, and whose transition relation outputs symbols only on the transitions leading to final states.

Henceforth, whenever the FSA family M is unambiguously identifiable, we will refer to Q_M, V_M, δ_M, etc., as Q, V, δ, etc., for short.

Definition 62 (transition closure)**.** *Let* M *be an FSA family. We define the* transition closure *of* δ *as a partial function* $\hat{\delta} \colon Q \times V^* \to Q$ *such that:*[4]

$$\hat{\delta}(q, \mathbf{w}) = \begin{cases} \hat{\delta}(\delta(q, w), \mathbf{w}'), & \text{if } \mathbf{w} = (w) \cdot \mathbf{w}' \\ q, & \text{if } \mathbf{w} = \epsilon. \end{cases}$$

Definition 63 (traversals)**.** *Let* D *be a GDAG. Then, we define the set of* D*'s* traversals*, denoted* $\mathcal{T}(D) \subset E_D$*, as:*

$$\mathcal{T}(D) = \{\langle v, c \rangle \in E_D : c \neq \epsilon\}$$

For any given $\langle v, c \rangle \in \mathcal{T}(D)$*,* v *is called the* head *and* c *is called the* body*.*

[4]Note that, in accordance with Def. 6, we use (w) to denote a 1-element sequence.

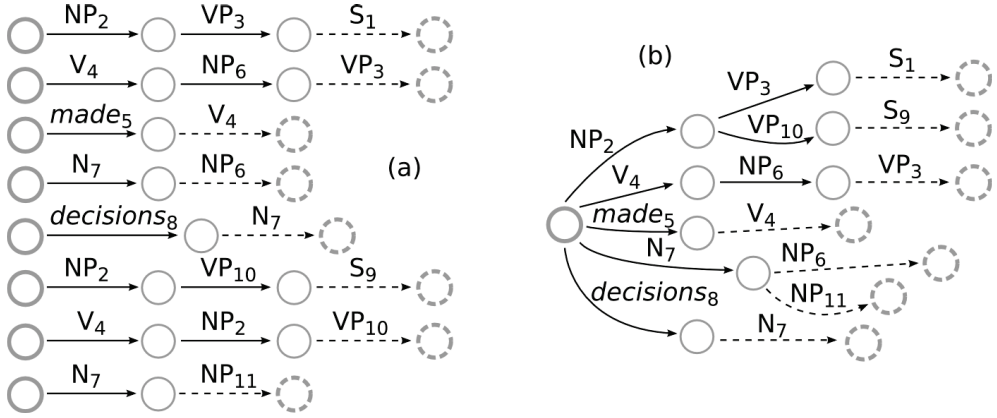


Figure 7.4: Two possible FSA encodings of the GDAG illustrated in Fig. 7.2 (b). Start states are highlighted in bold, while heads are marked with dashed arrows (leading to dummy, dashed states). (a) A simple FSA encoding in which each GDAG traversal is represented by a distinct quasi-FSA. (b) A prefix-tree-like FSA encoding.

Traversals are thus simply GDAG edges with non-empty sequences of target (children) vertices. From the parsing point of view, they represent traversals performed by the parser while matching the individual, non-trivial (i.e. of height greater than 0) grammar ESTs encoded in the GDAG against the input words.

Definition 64 (FSA encoding)**.** *Let D be a GDAG and M be an FSA family. Then, we say that M is an* FSA encoding *of D iff $V_M = V_D$ and M encodes all D's traversals with their heads put at the end of the corresponding FSA paths. Formally, for any $\langle v, c\rangle \in V_D \times V_D^*$ it must hold that:*

$$\langle v, c\rangle \in \mathcal{T}(D) \iff \exists!_{q_0 \in S_M}\Big(v \in heads(\hat{\delta}(q_0, c))\Big).$$

Note that, if M is an FSA encoding of D, then transitions in M carry the identifiers of the vertices (rather than of edges) in D. Fig. 7.4 illustrates two possible DAG encodings of the GDAG illustrated in Fig. 7.2 (b).

Definition 65 (traversal suffixes)**.** *Let M be an FSA family. We define $suff_M(q) \in 2^{V \times V^*}$ as the set of traversal suffixes – sequences of DAG vertices*

terminated with the head – encoded in M and starting from $q \in Q$.

$$\begin{aligned} \mathit{suff}(q) &= \mathit{suff}^b(q) \cup \mathit{suff}^h(q) \\ \mathit{suff}^b(q) &= \{\langle h, (v) \cdot b\rangle : \langle\langle q_1, v\rangle, q_2\rangle \in \delta, q_1 = q, \langle h, b\rangle \in \mathit{suff}(q_2)\} \\ \mathit{suff}^h(q) &= \{\langle h, \emptyset\rangle : h \in \mathit{heads}(q)\}. \end{aligned}$$

For instance, assuming the FSA family M illustrated in Fig. 7.4 (b) and that $S_M = \{q_0\}$ where q_0 is the left-most state, $\mathit{suff}(\hat{\delta}(q_0, (\mathrm{NP}_2, \mathrm{VP}_3))) = \{\langle \mathrm{S}_1, \epsilon\rangle\}$, $\mathit{suff}(\hat{\delta}(q_0, (\mathrm{N}_7))) = \{\langle \mathrm{NP}_6, \epsilon\rangle, \langle \mathrm{NP}_{11}, \epsilon\rangle\}$, $\mathit{suff}(\hat{\delta}(q_0, (\mathrm{NP}_2))) = \{\langle \mathrm{S}_1, (\mathrm{VP}_3)\rangle, \langle \mathrm{S}_9, (\mathrm{VP}_{10})\rangle\}$, $\mathit{suff}(\hat{\delta}(q_0, \epsilon)) = \{\langle \mathrm{S}_1, (\mathrm{NP}_2, \mathrm{VP}_3)\rangle, \langle \mathrm{S}_9, (\mathrm{NP}_2, \mathrm{VP}_{10})\rangle, \langle \mathrm{VP}_3, (\mathrm{V}_4, \mathrm{NP}_6)\rangle, \ldots\}$, etc.

Definition 66 (tree)**.** *Let M be an FSA encoding of a GDAG D. We extend the definition of a* tree *(cf. Def. 58) to FSA states as a function $\mathit{tree}_M \colon Q \to 2^{V_D}$ such that:*

$$\mathit{tree}_M(q) = \bigcup\{\mathit{tree}_D(v) : \langle v, c\rangle \in \mathit{suff}(q)\}$$

Namely, a given $q \in Q$ can be a part of several traversals of several different ETs and the function $\mathit{tree}(q)$ returns the set of such ETs, represented by D's roots. For instance, $\mathit{tree}(\hat{\delta}(q_0, (\mathrm{NP}_2, \mathrm{VP}_3))) = \{\mathrm{S}_1\}$, $\mathit{tree}(\hat{\delta}(q_0, (\mathrm{NP}_2))) = \{\mathrm{S}_1, \mathrm{S}_9\}$, $\mathit{tree}(\hat{\delta}(q_0, \epsilon)) = \{\mathrm{S}_1, \mathrm{S}_9, \mathrm{NP}_{11}\}$, etc.

Different GDAG $D \to$ FSAs encodings can be constructed. In the simplest possible characterization, each traversal $\langle h, c\rangle \in \mathcal{T}(D)$ is represented by a distinct, trivial path terminated with a single head, as illustrated in Fig. 7.4 (a). We will call such an encoding a *simple FSA encoding* of D.

For convenience, we define the versions of suff_M and tree_M working in the simplified context:

Definition 67 (suff^1_M)**.** *Let M be a simple FSA encoding and $q \in Q_M$. Then, we define* suff^1_M *as a function which returns the only traversal suffix represented by q. Formally, $\mathit{suff}^1_M(q) = \langle h, b\rangle$ such that $\mathit{suff}_M(q) = \{\langle h, b\rangle\}$.*

Definition 68 (tree^1_M)**.** *Let M be a simple FSA encoding and $q \in Q_M$. Then, we define $\mathit{tree}^1_M(q) = r$ such that $\mathit{tree}_M(q) = \{r\}$.*

7.3.3 Basic parser

We define the parsing algorithms implemented in ParTAGe in the deductive framework (Shieber et al., 1995), where parsing takes the form of an inference

Figure 7.5: A hypothetical parsing configuration considered by the parser. The *prime minister* is analysed as a MWE, while the possible ways of parsing the remaining part of the sentence are unknown yet.

process. We start by giving the specification of a simplified variant of the parsing algorithm used in ParTAGe. In Sec. 7.3.5 we describe the main version actually implemented in ParTAGe, which we extend to handle weights and feature structures in sections 7.3.7 and 7.5.3, respectively.

In what follows, we assume a fixed GDAG grammar representation D, a particular FSA encoding M of D (we will call them the *underlying* M and D), and a fixed input sentence $s \in \Sigma^*$. This corresponds to the fact that neither the sentence nor the grammar undergoes any modifications while parsing a particular input sentence. Nevertheless, it is worth keeping in mind that all the definitions introduced below are, in fact, parameterized by these three objects.

Chart items

Chart items in the deductive framework represent parsing configurations, while inference rules specify the elementary ways in which such parsing configurations can be combined. A sample parsing configuration is illustrated in Fig. 7.5. It represents a situation where the ET responsible for recognizing the MWE *prime minister* is matched with the corresponding words in the input sentence. Fig. 7.6, on the other hand, illustrates a configuration where (a part of) the *made decision* MWE ET is matched over the remaining part of the sentence. In the second configuration, the fact that *a few* and *good* are adjoined to the NP node and the N node, respectively, is not explicated. In fact, depending on the underlying grammar, other sub-derivations which lead to the same configuration may be possible (e.g. analyzing *a* as a modifier of the NP *few good decisions*, cf. the ET *a* in Fig. 7.3), but the possible sub-derivations leading to the individual configurations are abstracted over.

Fig. 7.7 illustrates an inference process which leads from the two configu-

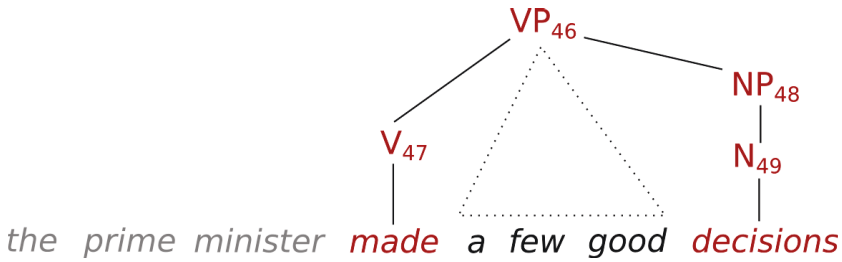


Figure 7.6: Another parsing configuration, where the VP subtree of the ET *made decisions* is matched against *made a few good decisions*.

rations shown above to the conclusions that:

- the left NP leaf of the ET *made decisions* (NP_{45}) can be substituted by the fully matched ET *prime minister* and,
- since both NP and VP subtrees of the entire ET *made decisions* are matched over adjacent parts of the input sentence, the ET itself can be matched over the entire input sentence (but the word *the*).

Definition 69 (positions)**.** *We define* $pos(s) = \{0, \ldots, n\}$ *as the set of positions between the words in the input sentence* s*, before* s_1 *and after* s_n*.*

Definition 70 (span)**.** *Let* s *be an input sentence and* $i, l \in pos(s)$*,* $j, k \in pos(s) \cup \{-\}$*,* $i \leq l$*,* $i \leq j \leq k \leq l$ *if* $(j, k) \neq (-, -)$*, and* $(j = -)$ *iff* $(k = -)$*. Then, we say that* $\langle i, j, k, l \rangle$ *is a* span *in* s*. For the sake of brevity, we will also write* $\langle i, l \rangle$ *to denote* $\langle i, -, -, l \rangle$*.*

Definition 71 (simple/gapped span)**.** *Let* $r = \langle i, j, k, l \rangle$ *be a span. Then, we say that* r *is* simple *iff* $\langle j, k \rangle = \langle -, - \rangle$*, and that it is* gapped *otherwise.*

Definition 72 (gap of a span)**.** *Let* $r = \langle i, j, k, l \rangle$ *be a gapped span. Then, we define its* gap*, denoted* $gap(r)$*, as a simple span* $\langle j, k \rangle$*.*

Definition 73 (coverage of a span)**.** *Let* $r = \langle i, j, k, l \rangle$ *be a span in the sentence* s*. Then, we define the* coverage *of* r*, denoted* $cover_s(r)$*, as:*

$$cover_s(r) = \begin{cases} \langle s_{i+1} \ldots s_j, s_{k+1} \ldots s_l \rangle, & \text{if } r \text{ is gapped} \\ s_{i+1} \ldots s_l, & \text{otherwise.} \end{cases}$$

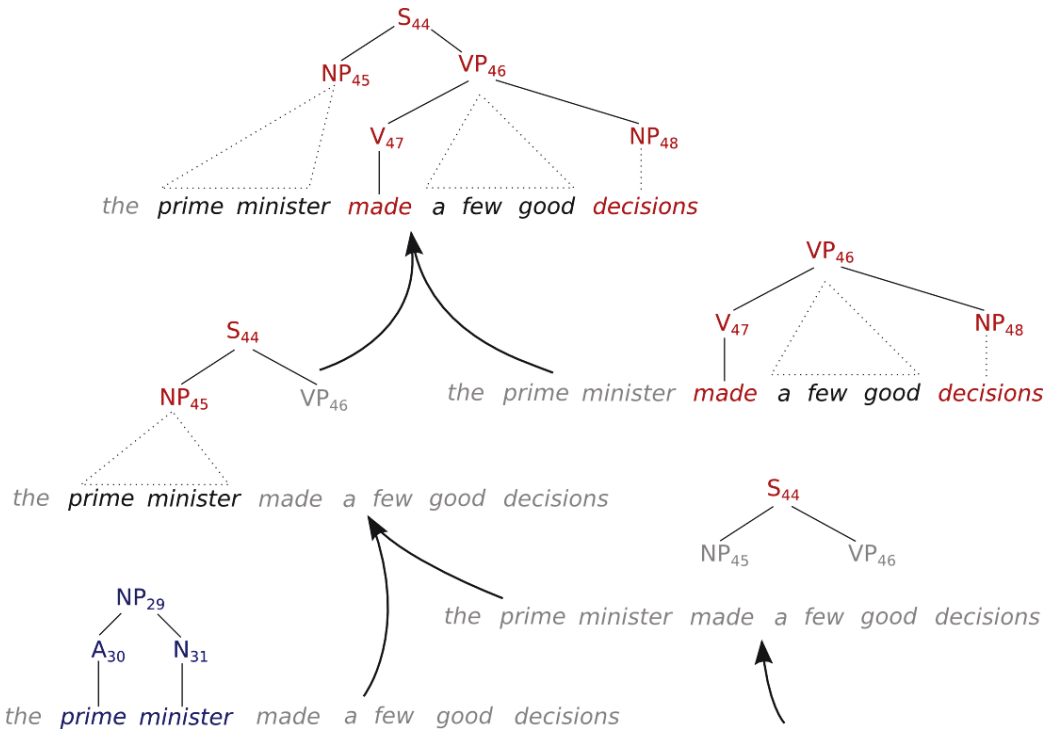


Figure 7.7: An inference leading from the two configurations shown in Fig. 7.5 and Fig. 7.6 to a configuration where the ET *made decisions* is matched over the entire sentence but the first word. It contains two active configurations, distinguished from the passive ones by marking some of the non-terminal nodes in gray (e.g. both the NP_{45} and VP_{46} nodes in the right-most bottom configuration).

Definition 74 (coverage of a derivation)**.** *Let δ be a derivation (tree or grove) and r be a span in s. Then, we say that δ* covers *r iff $\gamma(\delta) = cover_s(r)$.*

Definition 75 (coverage of a derivation chain)**.** *Let δ be a derivation chain and $r = \langle i, l \rangle$ be a simple span in s. Then, we say that δ* covers *r iff $\gamma(\delta) = cover_s(r)$.*

Formally, we define two types of chart items. **Passive chart items** represent configurations with fully recognized elementary subtrees (ESTs), as the ones in Fig. 7.5 and Fig. 7.6. **Active chart items**, on the other hand, represent configurations with partially recognized ESTs, e.g. the middle-left configuration in Fig. 7.7 where the subtree rooted in S_{44} is not fully recognized. More precisely, its left (trivial) subtree rooted in NP_{45} is matched against *prime minister*, but the right subtree rooted in VP_{46} is not aligned with any sentence words yet. Active items represent traversals of the individual grammar ESTs and are strongly related to the paths in the FSA representation of the grammar (cf. Sec. 7.3.2).

Definition 76 (passive item)**.** *Let $x \in V$ be a non-leaf node in the underlying GDAG, and r be a span in s. Then, $\langle x, r \rangle$ is a passive item.*

The exact interpretation of a passive item depends on the parsing algorithm. However, for all the parsers considered in this work, a passive item $\langle x, r \rangle$ exhibits the following invariant:

Proposition 12. *A passive chart item $\langle x, r \rangle$ asserts (provided that it is inferred by a parser) that a derivation tree δ exists such that the EST rooted in x is also the root of δ, and δ covers r.*

Definition 77 (active item)**.** *Let $q \in Q$ be a state of the underlying FSA family and r be a span in s. Then, $\langle q, r \rangle$ is an active item.*

Proposition 13. *An active item $\langle q, r \rangle$ asserts that a derivation grove $\delta = (\delta_i)_{i=1}^n$, a start state $q_0 \in S_M$, and a sequence of vertices $v = (v_i \in V)_{i=1}^n$ (prefix of some particular EST traversal) exist such that:*

- *$q = \hat{\delta}(q_0, v)$,*
- *the EST rooted in v_i is the root of δ_i for each $i = 1 \ldots n$, and*
- *δ covers r.*

Fig. 7.8 shows a more formal representation of the inference illustrated in Fig. 7.7, based on passive and active chart items which represent the corresponding parsing configurations, while Fig. 7.9 illustrates an inference which covers the words *good decisions*. For the sake of clarity, we distinguish active items from passive ones using the $_a$ and $_p$ subscript annotations, respectively. Besides, we assume a simple FSA encoding of the grammar depicted in Fig. 7.3 and we draw the individual traversals in the form of the dotted rules instead of specifying the states from the corresponding FSA representation. Elements on the left of a given dot represent the transition symbols on the path from a start state leading to a given state, while the elements on the right of an arrow represent the *heads*.

Definition 78 (chart item)**.** *Let $\langle x, r\rangle$ be either an active or a passive item, i.e., r is a span in s and $x \in Q_M \cup V_D$. Then, we say that $\langle x, r\rangle$ is a* chart item *(or* item *for short).*

Definition 79 (simple/gapped item)**.** *Let $v = \langle x, r\rangle$ be an item. Then, we say that v is* simple *iff r is simple and that v is* gapped *otherwise.*

Definition 80 (domain of items)**.** *We define a* domain of items*, denoted $\mathcal{I}$, as a set of active and passive chart items which can be possibly constructed by a parser over the sentence s.*

Inference rules

We now formally specify the basic, Earley-style, bottom-up TAG parser using the deductive framework (Shieber et al., 1995).

The inference rules of the basic parser, a simplified version of the parser implemented in ParTAGe, are given in Tab. 7.1. The basic parser also constitutes a version of the bottom-up Earley-like parser described in (Alonso et al., 1999), with three notable differences: (i) our parser allows multiple adjunctions at the same node, (ii) it explicitly handles substitution, and (iii) it relies on a particular, potentially compressed grammar representation.

Each of the inference rules takes zero, one or two chart items on input (*premises*, presented above the horizontal line) and yields a new item (*conclusion*, presented below the line) to be added to the chart if the conditions given on the right-hand side – which should refer only to the premise items and to the static properties of the parser, e.g., to the underlying grammar – are met. We can think of inference rules as functions whose domain is $2^{\mathcal{I}}$ (cf.

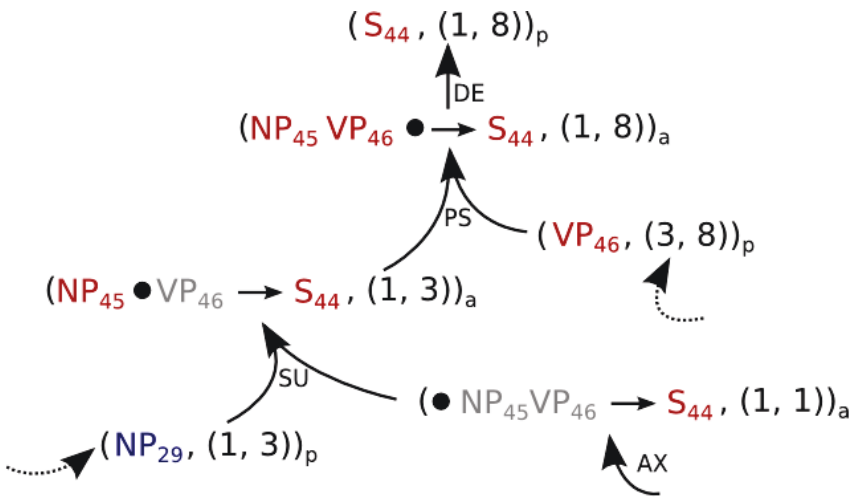


Figure 7.8: Fragment of the inference corresponding to the inference shown in Fig. 7.7, based on passive and active chart items. Applications of the inference rules are marked with their types (AX, DE, SU, ...). Note that, due to explicit application of the DE rule, this derivation has one additional arc in comparison with the one from Fig. 7.7.

Def. 81) because (i) the order in which the premise items are presented does not matter, and (ii) a single item from $\mathcal{I}$ cannot instantiate more than one premise of a given inference rule.[5]

Definition 81 (inference rule)**.** *Let $\mathcal{I}$ be a domain of items and $d\colon 2^{\mathcal{I}} \to \mathcal{I}$ be a partial function. Then, we say that d is an* inference rule*.*

Definition 82 (inference step)**.** *Let $\mathcal{I}$ be a domain of items, d be an inference rule, $V \subseteq \mathcal{I}$, and $v \in \mathcal{I}$. Then, we say that d* infers *v* from *V (or, alternatively, that v is* inferred from *V by d) iff $d(V)$ is defined and $d(V) = v$. Then we also say that d* applies to *V.*

The **axiom** rule (AX) introduces an active item for each position $(pos(s) \setminus \{n\})$ in the input sentence and each start state in the underlying FSA family.

[5]The property (ii) is not required to hold in general, but it applies to all the parsers described in this work.

The intuition behind AX is that any given grammar EST can potentially match any given span in the input sentence, thus its traversal has to be considered from each (non-final) position in s.

The **scan** rule (SC) matches the grammar terminal symbols with words from the input. More precisely, for a given $\langle q, \langle i, j, k, l\rangle\rangle_a$ active item, it checks whether a transition leading from the state $q \in Q$ to another state in Q exists such that the transition symbol v corresponds to the input word s_{l+1}, i.e., to the word immediately on the right of the item's span $\langle i, j, k, l\rangle$. It relies on the following function:

Definition 83 (leaves)**.** *Let D be a DAG encoding. We define* leaves_D *as a function from $N_D \cup \Sigma_D$ to 2^{V_D} which provides the set of vertices $v \in V_D$ such that $leaf_D(v) \wedge (\ell_D(v) = x)$ for a given $x \in N_D \cup \Sigma_D$.*

For instance, in Fig. 7.3, $leaves(NP) = \{3, 13, 16, 23, 27, 37\}$, $leaves(\text{few}) = \{28, 39\}$, etc. Let us note that the direct use of this function in the implementation of the parser might seem sub-optimal at first sight, because the sets of *leaves* corresponding to the individual terminals and non-terminals can be very large. However, this issue does not occur when the grammar is compressed first (see Sec. 7.5.2).

Deactivation (DE) transforms an active item into the corresponding passive item, provided that the active item represents a completed traversal of an EST. If all children subtrees of a given EST are matched against contiguous parts of a particular span, the entire EST (represented by the resulting passive item) can be also matched against the same span.

Pseudo substitution (PS) is similar to scan, but instead of matching grammar terminals against input words, grammar non-terminals are matched against already inferred non-terminals represented by passive items. It serves to match together two adjacent fragments of the same ET already recognized over adjacent spans.[6] **Substitution** (SU), on the other hand, handles regular substitution, i.e., it matches a fully recognized initial ET with a (non-foot) non-terminal leaf of another ET, the latter corresponding to the symbol of the next transition in the underlying FSAs.

The **foot adjoin** rule (FA) matches the foot – the next element on the traversal represented by the active premise item – against a simple span r placed directly on the right of the premise item's span. It puts up a hypothesis

[6]Note that the two premise items of the PS rule cannot be both gapped. Otherwise, an ET with two foots would have to exist.

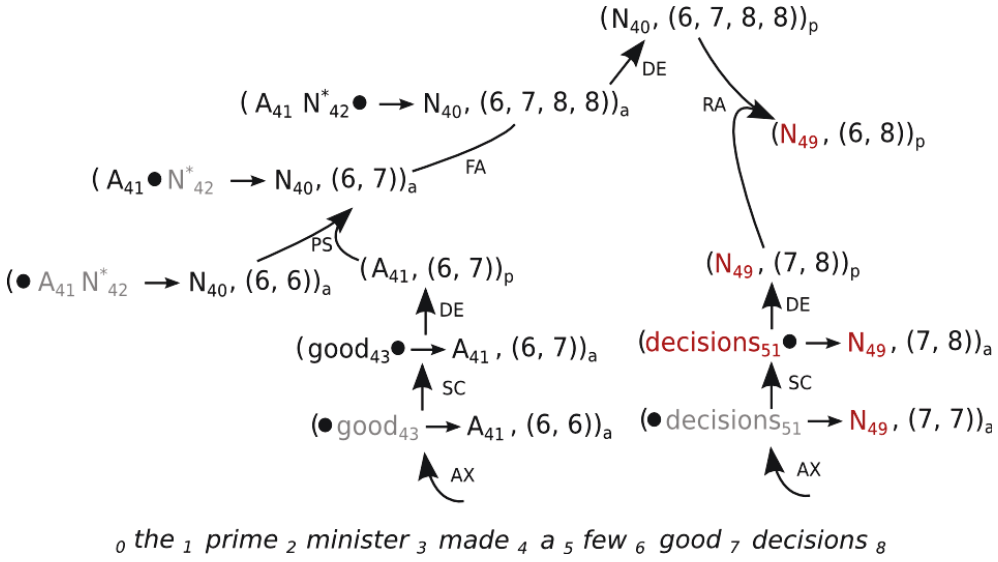


Figure 7.9: An inference leading to a derivation of N_{49} and the corresponding EST over the words *good decisions*.

that the EAT represented by the premise item can be at some point adjoined to the root of an EST recognized over r.

Finally, The **root adjoin** rule (RA) represents the actual adjoining of a fully recognized EAT t into the root of a recognized EST t'. Information that t' is recognized (with a modified span) is preserved in the conclusion and can be reused in order to recognize the full ET of which t' is a part.

The basic parser allows to modify a given ET node with several adjunctions, which is consistent with the extended model of TAG derivations (Schabes and M. Shieber, 1994), in opposition to the standard model of derivations where at most one adjunction per node is allowed (Vijay-Shanker, 1987). Adjunction can be also performed over the roots of EATs, which allows to distinguish independent (e.g., when both *prime* and *good* modify *decisions*) from independent (e.g., when the ET *prime* modifies the root of the ET *good* which, in turn, modifies *decisions*) derivations.

Definition 84 (parsing system)**.** *Let $\mathcal{I}$ be the domain of items and $\mathcal{D}$ be the set of inference rules which serve to deduce items in $\mathcal{I}$. Then, we say that $\langle \mathcal{I}, \mathcal{D} \rangle$ is a* parsing system*. We will also call the system specified by the rules in Tab. 7.1 the* basic parsing system*.*

Let us recall that a parsing system is implicitly parametrized by a TAG,

AX:	$\frac{}{\langle q_0,\langle i,i\rangle\rangle_a}$	$i\in pos(s)\backslash\{n\}$ $q_0\in S_M$
SC:	$\frac{\langle q,\langle i,j,k,l\rangle\rangle_a}{\langle\delta(q,v),\langle i,j,k,l+1\rangle\rangle_a}$	$v\in leaves(s_{l+1})$ $\delta(q,v)$ defined
DE:	$\frac{\langle q,\langle i,j,k,l\rangle\rangle_a}{\langle v,\langle i,j,k,l\rangle\rangle_p}$	$v\in heads(q)$
PS:	$\frac{\langle q,\langle i,j,k,l\rangle\rangle_a \quad \langle v,\langle l,j',k',l'\rangle\rangle_p}{(\delta(q,v),\langle i,j\cup j',k\cup k',l'\rangle\rangle_a}$	$\delta(q,v)$ defined
SU:	$\frac{\langle q,\langle i,j,k,l\rangle\rangle_a \quad \langle v,\langle l,l'\rangle\rangle_p}{\langle\delta(q,v'),\langle i,j,k,l'\rangle\rangle_a}$	$v'\in leaves(\ell(v))\wedge\neg foot(v')$ $\delta(q,v')$ defined $root(v)$
FA:	$\frac{\langle q,\langle i,l\rangle\rangle_a}{\langle\delta(q,v),\langle i,l,l',l'\rangle\rangle_a}$	$l'\in pos(s)\backslash\{n\}\wedge l<l'$ $v\in V\wedge\delta(q,v)$ defined $foot(v)$
RA:	$\frac{\langle w,\langle i,j,k,l\rangle\rangle_p \quad \langle v,\langle j,j',k',k\rangle\rangle_p}{\langle v,\langle i,j',k',l\rangle\rangle_p}$	$root(w)\wedge(j,k)\neq(-,-)$ $\ell(w)=\ell(v)$

Table 7.1: Inference rules underlying the *basic* parser, where $i \cup j$ is equal to i if $j = -$ and j otherwise.

its GDAG and FSA representations, and an input sentence.

Definition 85 (deterministic parsing system)**.** *Let $\langle \mathcal{I}, \mathcal{D} \rangle$ be a parsing system. Then, we say that $\langle \mathcal{I}, \mathcal{D} \rangle$ is* deterministic *iff, for any $V \subseteq \mathcal{I}$, there is at most one $d \in \mathcal{D}$ such that d applies to V.*

Proposition 14. *A basic parsing system is deterministic.*

All the parsing systems described in this work are deterministic, a property which we tacitly assume in the remaining of this document. While it is not crucial for ParTAGe, it simplifies the formal description of the parser.

7.3.4 Parsing and hypergraphs

Formally, a parsing inference can be expressed in terms of a hypergraph (Klein and Manning, 2001a; Gallo et al., 1993).[7] The idea behind the hypergraph representation is that each item deduced throughout the inference process is represented as a node, and each application of an inference rule – as a *hyperarc*. Namely, the conclusion node (called the *head* of the arc) is connected via an arc with the premise node or nodes (the *tail*) of a given inference rule's application.

Definition 86 (hypergraph)**.** *We define a* hypergraph *as a tuple $\mathcal{H} = \langle V_\mathcal{H}, E_\mathcal{H} \rangle$ such that:*

- *$V_\mathcal{H}$ is the set of nodes in $\mathcal{H}$,*
- *$E_\mathcal{H} \subseteq 2^{V_\mathcal{H}} \times V_\mathcal{H}$ is the set of hyperarcs which connect the individual nodes in $\mathcal{H}$. We also define two subsidiary functions* tail *and* head *which, for a given $\langle V, v \rangle \in E_\mathcal{H}$, return V and v, respectively.*
- *$\forall_{e \in E_\mathcal{H}}(head(e) \notin tail(e))$, and*
- *$\forall_{v \in V_\mathcal{H}} \exists_{e \in E_\mathcal{H}}(head(e) = v)$.*

The third property requires that the head of a given hyperarc is not in its tail, while the fourth property requires that each node in the hypergraph is the head of one or more hyperarcs.

[7] In the sense of (Gallo et al., 1993), we use *B-graphs*, a particular type of hypergraphs.

Definition 87 (parsing hypergraph)**.** *Let $\langle \mathcal{I}, \mathcal{D} \rangle$ be a parsing system. Then, we define the corresponding* parsing hypergraph *$\mathcal{H}$, such that $V_\mathcal{H} \subseteq \mathcal{I}$, as a fixed point of the application of the rules in $\mathcal{D}$, i.e., as the hypergraph satisfying the following equation:*

$$\langle V, v \rangle \in E_\mathcal{H} \iff \exists_{d \in \mathcal{D}}\ d \text{ infers } v \text{ from } V$$

From Def. 87 stems the basic construction of the hypergraph-based parsing algorithm, whose input is the parsing system $\langle \mathcal{I}, \mathcal{D} \rangle$ and the output is the resulting parsing hypergraph. Namely, the algorithm starts with an empty hypergraph $\langle V_{\mathcal{H}_0}, E_{\mathcal{H}_0} \rangle = \langle \emptyset, \emptyset \rangle$ and iteratively performs the following steps:

1. For each rule $d \in \mathcal{D}$, find the matching premise items V (if any) amongst the items already present in $V_{\mathcal{H}_i}$.
2. Determine the result v of the application of d on V.
3. Set $V_{\mathcal{H}_{i+1}} = V_{\mathcal{H}_i} \cup \{v\}$ and $E_{\mathcal{H}_{i+1}} = E_{\mathcal{H}_i} \cup \{\langle V, v \rangle\}$.

These steps are repeated as long as any new items or hyperarcs can be created. Once the process stops in the k-th iteration, the resulting hypergraph is defined as $\mathcal{H} = \langle V_{\mathcal{H}_k}, E_{\mathcal{H}_k} \rangle$.

Proposition 15. *Let $\langle \mathcal{I}, \mathcal{D} \rangle$ be a basic parsing system and $\mathcal{H}$ be the corresponding hypergraph. Then, for any $\langle V, v \rangle \in E_\mathcal{H}$, there is exactly one $d \in \mathcal{D}$ such that d infers v from V.*

The above property (which stems directly from Prop. 14) entails that, given a parsing system and the corresponding hypergraph, it is possible to unambiguously determine the inference rules used to infer the individual hyperarcs.

Definition 88 (simple path)**.** *Let $\mathcal{H}$ be a hypergraph, $v, w \in V_\mathcal{H}$ and $e_1 \dots e_k \in E^*_\mathcal{H}$ such that $k \geq 1$. Then, we say that $e_1 \dots e_k$ is a* simple path *connecting v with w iff:*

$$v \in \mathit{tail}(e_1) \wedge w = \mathit{head}(e_k) \wedge \forall_{i=1}^{k-1}(\mathit{head}(e_i) \in \mathit{tail}(e_{i+1}))$$

Proposition 16. *Let G be a lexicalized TAG, s be a sentence and $\langle \mathcal{I}, \mathcal{D} \rangle$ be a corresponding basic parsing system. Then, the corresponding parsing hypergraph is acyclic, i.e., for any $v \in V_\mathcal{H}$ there is no simple path connecting v with itself.*

It is possible to define a total order over items and to show that each conclusion item is greater than any of the premise items in this order, hence the above proposition.

Definition 89 (sub-hypergraph)**.** *Let $\mathcal{H}$ and $\mathcal{H}'$ be two hypergraphs. We say that $\mathcal{H}$ is a* sub-hypergraph *of $\mathcal{H}'$, denoted $\mathcal{H} \subseteq \mathcal{H}'$, iff $V_{\mathcal{H}} \subseteq V_{\mathcal{H}'} \wedge E_{\mathcal{H}} \subseteq E_{\mathcal{H}'}$.*

Definition 90 (hypergraph union)**.** *Let $\mathcal{H}$ and $\mathcal{H}'$ be two hypergraphs. We define their* hypergraph union*, denoted $\mathcal{H} \cup \mathcal{H}'$, as the elementwise union, i.e, as the hypergraph $\langle V_{\mathcal{H}} \cup V_{\mathcal{H}'}, E_{\mathcal{H}} \cup E_{\mathcal{H}'} \rangle$.*

Definition 91 (hyperpath)**.** *Let $\mathcal{P}$ be a hypergraph. Then, we say that $\mathcal{P}$ is a* hyperpath *iff:*

- $\forall_{v \in V_{\mathcal{P}}} \exists!_{e \in E_{\mathcal{P}}} (head(e) = v)$*, and*
- $\exists!_{v \in V_{\mathcal{P}}} \neg \exists_{e \in E_{\mathcal{P}}} (v \in tail(e))$.

The first point refers to the fact that, for each node $v \in V_{\mathcal{P}}$, there is exactly one hyperarc leading to v (i.e. whose head is v). The second point means that a hyperpath contains one distinguished node, the *root*, from which no other hyperarcs can leave. From these two points stems the conclusion that a hyperpath can be seen as a tree of items where the individual items are deduced from their children items via the inference rules.

Definition 92 (hyperpath root)**.** *Let $\mathcal{P}$ be a hyperpath. Then, we define its* root*, denoted $root(\mathcal{P})$, as the only $v \in V_{\mathcal{P}}$ such that $\neg \exists_{e \in E_{\mathcal{P}}} (v \in tail(e))$.*

Definition 93 (hyperpath in a hypergraph)**.** *Let $\mathcal{H}, \mathcal{P}$ be two hypergraphs. Then, we say that $\mathcal{P}$ is a* hyperpath in $\mathcal{H}$*, denoted $\mathcal{P} \in \mathcal{H}$, iff $\mathcal{P}$ is a hyperpath and $\mathcal{P} \subseteq \mathcal{H}$.*

The parsing hypergraph encodes all the possible inferences and, consequently, all the possible TAG derivation trees corresponding to a given sentence. Each item $q \in \mathcal{I}$ present in the parsing hypergraph asserts that a derivation which covers the item's span exists, while the different possible ways of constructing such derivations are represented by the different possible hyperpaths leading to q (i.e. all the hyperpaths in $\mathcal{H}$ whose root is q).

For instance, Fig. 7.10 (a) shows a fragment of the derivation corresponding to the fragment of the inference hyperpath illustrated in Fig. 7.8, while Fig. 7.10 (b) shows a fragment of the derivation corresponding to the fragment

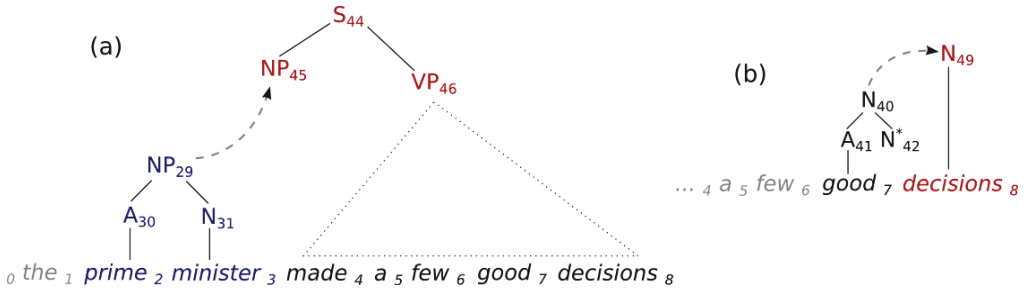


Figure 7.10: Two derivations, (a) and (b), corresponding to the inference fragments illustrated in Fig. 7.8 and Fig. 7.9, respectively. Alignment between VP_{46}, its corresponding EST, and the input words can be only hypothetized, since the inference fragment from Fig. 7.8 does not specify the inference leading to item $(\mathrm{VP}_{46}, (3, 8))_p$.

of the inference hyperpath illustrated in Fig. 7.9. The former includes an application of substitution which corresponds to the SU inference rule, while the latter contains a (partial) application of adjunction which corresponds to the RA inference rule. Applications of the PS rules are not explicitly present in the corresponding derivations, but they allow to join together adjacent ET fragments.

Definition 94 (GDAG derivation)**.** *Let G be a TAG and D be its DAG encoding. Then, we define a* GDAG derivation *(tree or grove) as a derivation (tree or grove) which contains D vertices in its nodes instead of ESTs from G (recall that each vertex $v \in V_D$ determines the corresponding EST in G, cf. Prop. 8).*

A single TAG derivation δ can be sometimes represented by several GDAG derivations – notably, if δ's root is an EST which is a subtree of several ETs in the grammar. For instance, Fig. 7.10 (b) shows a TAG derivation which would be identical if its root EST, rooted in N_{49}, were replaced by the EST rooted in N_{19} (see Fig. 7.3). Yet, such a replacement would entail a different GDAG derivation. In Sec. 7.5.2, we consider a simple subtree sharing DAG encoding technique which makes the two representations – TAG derivations and GDAG derivations – equivalent.[8]

[8]However, as mentioned before, it may be useful to distinguish two identical ETs if they come with two different FS-unification computations.

Proposition 17 (correspondence between hyperpaths and derivations)**.** *Let $\langle \mathcal{I}, \mathcal{D} \rangle$ be a basic parsing system and $\mathcal{H}$ be the corresponding hypergraph. Then, there is a one-to-one correspondence between hyperpaths in $\mathcal{H}$ and GDAG derivations which can be constructed over s. More precisely, each hyperpath in $\mathcal{H}$ uniquely identifies the corresponding GDAG derivation and, vice versa, each GDAG derivation covering some span in s uniquely identifies the corresponding hyperpath.*

The goal of the symbolic, hypergraph-based parsing algorithm can be seen as a two-step process: first the parser should construct the entire hypergraph $\mathcal{H}$, next it should retrieve from $\mathcal{H}$ the individual hyperpaths leading to final nodes. Now we sketch the algorithm which allows to perform the latter step.

First of all, we need to determine which items ($\mathcal{H}$ nodes) represent full derivations spanning the entire input sentence.

Definition 95 (final item)**.** *Let $\langle \mathcal{I}, \mathcal{D} \rangle$ be a parsing system and $\langle v, r \rangle \in \mathcal{I}$ be a passive item. We say that $\langle v, r \rangle \in \mathcal{I}$ is* final*, denoted final$(\langle v, r \rangle)$, iff $r = \langle 0, |s| \rangle \wedge \ell(v) = S$, where S is the start symbol of the underlying TAG.*

Definition 96 (final hyperpath)**.** *Let $\langle \mathcal{I}, \mathcal{D} \rangle$ be a parsing system, $\mathcal{H}$ be the corresponding hypergraph, and $\mathcal{P} \in \mathcal{H}$. Then, we say that $\mathcal{P}$ is* final*, denoted final$(\mathcal{P})$, iff final$(root(\mathcal{P}))$.*

The process of retrieving all the hyperpaths encoded in the given hypergraph $\mathcal{H}$ takes the following, recursive form:

$$ins_{\mathcal{H}}(x) = \begin{cases} \{\mathcal{P} \in ins_{\mathcal{H}}(e) : e \in E_{\mathcal{H}}, head(e) = x\}, \text{ if } x \in V_{\mathcal{H}} \\ edge(x) \cup \begin{cases} \mathcal{P}_{\emptyset}, \text{ if } x \in E_{\mathcal{H}} \wedge tail(x) = \emptyset \\ \bigcup_{v \in tail(x)} ins_{\mathcal{H}}(v), \text{ otherwise,} \end{cases} \end{cases} \tag{7.5}$$

where $\mathcal{P}_{\emptyset} = \langle \emptyset, \emptyset \rangle$ is an empty hyperpath and *edge* is a function which creates a trivial, single-edge hypergraph from a given arc:

$$edge(\langle V, v \rangle) = \langle V \cup \{v\}, \{\langle V, v \rangle\} \rangle. \tag{7.6}$$

For a given node $v \in V_{\mathcal{H}}$, the function $ins_{\mathcal{H}}(v)$ returns the set of all hyperpaths leading to v, while for a given hyperarc $e \in E_{\mathcal{H}}$, the function $ins_{\mathcal{H}}(e)$ returns the set of all hyperpaths leading to e.

Definition 97 (inside derivation)**.** *Let $\mathcal{H}$ be a hypergraph and $x \in V_{\mathcal{H}} \cup E_{\mathcal{H}}$. Then, we call each element of $ins_{\mathcal{H}}(x)$ an* inside derivation *of x.*

Thus, in order to retrieve all the final paths encoded in a given hypergraph $\mathcal{H}$, $ins_{\mathcal{H}}$ needs to be called over each final node in $\mathcal{H}$. This process ideally leads to a small set of syntactically plausible analyses, but from the theoretical standpoint there is no guarantee of that. Some hypergraph nodes may belong to several different final hyperpaths and the algorithm described above will visit them (and retrieve the corresponding sub-paths) several times as well. In fact, the resulting set of final hyperpaths (and, consequently, TAG derivations spanning the entire input sentence) can be even exponential.[9] Hence the need for syntactic disambiguation or for more elaborate parsing architectures in which subsequent processing steps (e.g. semantic analysis) take place directly over the hypergraph or a similar compressed representation of derivations.

7.3.5 Vanilla parser

The *vanilla* algorithm constitutes the main parser implemented in ParTAGe. Its inference rules are presented in Tab. 7.2. It brings two modifications in comparison with the basic parser: (i) the FA rule identifies ranges over which adjunction can possibly occur, instead of blindly skipping an arbitrary number of words to match the foot, (ii) adjunction is not allowed to take place over the roots of the auxiliary ETs.

The FA rule ensures that the resulting item is considered only if an elementary (sub)tree, recognized starting from l, and to which the corresponding ATs[10] could be adjoined, exists. In comparison with the FA rule of the basic parser, it requires (as a premise) a passive item which serves as a witness that adjunction over the given span can possibly take place. We will call such passive items *witness* items of the FA rule. The FA rule of the vanilla parser allows to significantly trim the search space and performs a similar function as the prediction rule in CFG parsing. It also entails that the invariants asserted by the individual chart items are more complex than in the case of the basic parser.

Definition 98 (vanilla derivation chain)**.** *Let $\vec{\delta}$ be a derivation chain. Then, we say that $\vec{\delta}$ is a* vanilla derivation chain *iff it complies with the vanilla*

[9] It could be even infinite if unary cycles were allowed in the hypergraph.

[10] Assuming the simple DAG and FSA encodings, there can be only one AT corresponding to a given traversal. In general, however, a particular traversal's prefix can correspond to several ETs.

parser, i.e., none of the auxiliary derivation trees $\vec{\delta}_i$ in $\vec{\delta}$ is an ET.[11]

Proposition 18. *Within the context of the vanilla parser, a passive item $\langle x, r \rangle$ asserts:*

- *an existence of a derivation tree δ such that δ covers r and the EST rooted in x is also the root of δ (cf. Prop. 12), and*
- *provided that r is gapped (which entails that δ is auxiliary), an existence of a vanilla derivation chain $\vec{\delta}$ such that $\vec{\delta}$ covers the r's gap and the $\vec{\delta}$'s tip (cf. Def. 35) contains the same non-terminal as the δ's foot.*

Proposition 19. *Within the context of the vanilla parser, an active item $\langle q, r \rangle$ asserts:*

- *an existence of a derivation grove δ satisfying the properties described in proposition 13, and*
- *provided that r is gapped, an existence of a vanilla derivation chain $\vec{\delta}$ such that $\vec{\delta}$ covers the r's gap and the $\vec{\delta}$'s tip contains the same non-terminal as the δ's foot.*

An example of a hyperpath which contains an application of the vanilla FA inference rule is shown in Fig. 7.11.

The modification of the parsing algorithm also entails that Proposition 17 no longer holds as it is. Rather, there exists a correspondence between hyperpaths, on the one hand, and (i) TAG derivations and (ii) TAG derivation chains, if the gap is defined, on the other hand.

The additional constraint in RA and in FA imposed on the modified node is that it must not be a root of an AT. This means that the vanilla parsing algorithm models the so-called independent derivations but does not allow the traditional, dependent ones, where one AT modifies the root node of another AT. A similar constraint is adopted in *tree insertion grammars*, where the roots of auxiliary trees cannot be modified, while simultaneous adjunction is allowed (Schabes and C. Waters, 1995). Gardent and Narayan (2015) argue that mixed derivations should be allowed in general, and in Ex. 7.7 in particular. While both *old* and *Syrian Orthodox* independently modify *church*, it is not true with respect to *Syrian* and *Orthodox* separately. Gardent

[11] This is related to the fact that the vanilla parser does not allow the EATs to adjoin to the roots of other EATs.

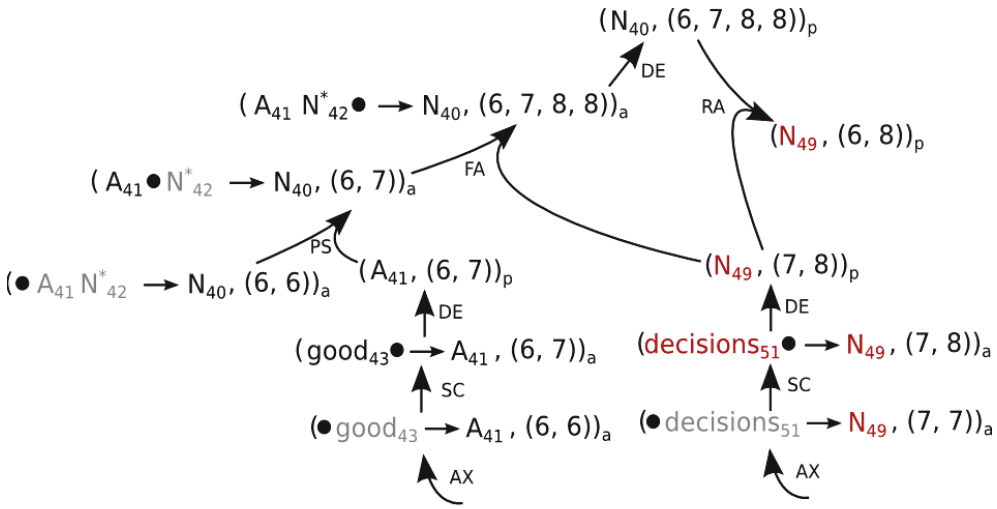


Figure 7.11: A hyperpath constructed with the vanilla parser on top of the words *good decisions* (see also Fig. 7.9). Note that the item $\langle \mathrm{N}_{49}, \langle 7, 8 \rangle \rangle$ first serves as a witness that the foot N^*_{42} can be matched against the span $\langle 7, 8 \rangle$, and afterwards it is effectively adjoined to once the entire auxiliary ET rooted in N_{40} is recognized over the span $\langle 6, 7, 8, 8 \rangle$.

AX:	$\frac{}{\langle q_0,\langle i,i\rangle\rangle_a}$	$i\in pos(s)\backslash\{n\}$ $q_0\in S_M$
SC:	$\frac{\langle q,\langle i,j,k,l\rangle\rangle_a}{\langle\delta(q,v),\langle i,j,k,l{+}1\rangle\rangle_a}$	$v\in leaves(s_{l+1})$ $\delta(q,v)$ defined
DE:	$\frac{\langle q,\langle i,j,k,l\rangle\rangle_a}{\langle v,\langle i,j,k,l\rangle\rangle_p}$	$v\in heads(q)$
PS:	$\frac{\langle q,\langle i,j,k,l\rangle\rangle_a \quad \langle v,\langle l,j',k',l'\rangle\rangle_p}{(\delta(q,v),\langle i,j\cup j',k\cup k',l'\rangle\rangle_a}$	$\delta(q,v)$ defined
SU:	$\frac{\langle q,\langle i,j,k,l\rangle\rangle_a \quad \langle v,\langle l,l'\rangle\rangle_p}{\langle\delta(q,v'),\langle i,j,k,l'\rangle\rangle_a}$	$v'\in leaves(\ell(v))\wedge\neg foot(v')$ $\delta(q,v')$ defined $root(v)$
FA:	$\frac{\langle q,\langle i,l\rangle\rangle_a \quad \langle v,\langle l,j',k',l'\rangle\rangle_p}{\langle\delta(q,v'),\langle i,l,l',l'\rangle\rangle_a}$	$v'\in leaves(\ell(v))\wedge foot(v')$ $\delta(q,v')$ defined $root(v)\Longrightarrow(j',k')=(-,-)$
RA:	$\frac{\langle w,\langle i,j,k,l\rangle\rangle_p \quad \langle v,\langle j,j',k',k\rangle\rangle_p}{\langle v,\langle i,j',k',l\rangle\rangle_p}$	$root(w)\wedge(j,k)\neq(-,-)$ $\ell(w)=\ell(v)$ $root(v)\Longrightarrow(j',k')=(-,-)$

Table 7.2: Inference rules underlying the *vanilla* parser. Recall that $i \cup j = i$ if $j = -$ and j otherwise.

and Narayan (2015) propose to analyse *Syrian* as a modifier of *Orthodox*. An alternative analysis would be that *Syrian Orthodox church*, as a multiword named entity, should be modeled as a single ET.

(7.7) The meerkat admired the old Syrian Orthodox church.

The vanilla algorithm could be modified to allow dependent derivations by removing the $root(v) \implies (j', k') = (-, -)$ constraint from the RA and the FA rules. The parser could be also left as is and the dependent modifications could be distinguished from the independent ones at the moment of the unfolding of the derivations encoded in the hypergraph. However, the two types of modifications can yield different FS values, thus it could be hard to reconcile the latter solution with FS-aware parsing.

In the actual implementation of the vanilla parser, two types of passive items are distinguished – *top* passive items, which represent fully recognized ETs, and non-top passive items which represent fully recognized ESTs which are not ETs. This modification slightly changes the form of the individual

inferences rules – e.g., the SU rule requires a top-passive premise item and, thus, does not have to refer to grammar leaves – and makes the size of the resulting inference computations smaller – different top-passive items over the same span and referring (via $\ell(v)$) to the same non-terminal are then conflated into a single item. We will introduce this change in the subsequent versions of the parsing algorithm.

7.3.6 Parsing algorithm

Here we give a more detailed description of the parsing algorithm sketched above. It applies to the vanilla parsing system, but can be easily extended to cover additional inference rules and handle weights.

Algorithm 1 Parsing algorithm

```
1: Q ← ∅                                   ▷ The priority queue with open items
2: C ← ∅                                   ▷ The set of closed items
3: E ← ∅                                   ▷ The set of hyperarcs
4: APPLY-AX                                ▷ Create all the instantiations of the AX rule
5: while Q ≠ ∅ do
6:     x ← DELETE-MIN(Q)
7:     C ← C ∪ {x}
8:     if passive x then                   ▷ Consider rules depending on x's type
9:         APPLY-PS(x); APPLY-SU(x);
10:        APPLY-FA(x); APPLY-RA(x)
11:    else
12:        APPLY-SC(x); APPLY-DE(x)
13:    end if
14: end while
```

Algorithm 1 shows the pseudocode of the body of the main procedure of the parser. It first creates three basic data structure: the priority queue of open items Q, the set of closed items C, and the set of hyperarcs E. At the end of this procedure, $Q = \emptyset$ and the resulting hypergraph is specified by $V_{\mathcal{H}} = C$ and $E_{\mathcal{H}} = E$.

The algorithm is based on a total order defined over items, which allows to store them in a priority queue. This total order (see Def. 99) guarantees that, if there are two items which can undergo an application of a particular rule $d \in \mathcal{D}$, then they will be processed by the algorithm (i.e. removed from

Q) always in the same order, regardless of their particular form. For instance, the passive premise item of the PS rule will always be removed after its active premise item, while the passive premise item of RA representing the full AT will always be removed after the passive premise item representing the EST node being adjoined to.

Definition 99 (total order on items)**.** *Let $\langle x, r_1\rangle$ and $\langle y, r_2\rangle$ be two items. Then we define the total order $\leq$ as*

$$\langle x, r_1\rangle \leq \langle y, r_2\rangle \iff \langle end(r_1), len(r_1)\rangle \leq_{lex} \langle end(r_2), len(r_2)\rangle,$$

where $beg(\langle i, j, k, l\rangle) = i$, $end(\langle i, j, k, l\rangle) = l$, $len(r) = end(r) - beg(r)$, and $\leq_{lex}$ is the lexicographic order on $\mathbb{N} \times \mathbb{N}$.

The fact that the above definition takes into account the length of the span is related to the RA rule, which can in general apply to two passive items with the same ending position.

Proposition 20. *Let $\langle \mathcal{I}, \mathcal{D}\rangle$ be a vanilla parsing system and $v, w \in \mathcal{I}$ be two items such that $v \leq w$ and $w \leq v$. Then, none of the rules $d \in \mathcal{D}$ can be applied to $\{v, w\}$.*

Algorithm 2 Initialization: determine instantiations of the AX rule

```
15: procedure APPLY-AX
16:     for i ∈ pos(s) \ {n} do          ▷ For each non-terminal position in s
17:         for q_0 ∈ S_M do                          ▷ For each FSA start state
18:             x ← ⟨q_0⟨i, i⟩⟩_a                    ▷ Create the resulting item
19:             Q ← Q ∪ {x}                                ▷ Put it in the queue
20:             E ← E ∪ {⟨∅, x⟩}                ▷ Store the incoming hyperarc
21:         end for
22:     end for
23: end procedure
```

Alg. 1 satisfies a property which is important for the analysis of its time complexity:

Proposition 21. *Let $\langle \mathcal{I}, \mathcal{D}\rangle$ be a vanilla parsing system and $\mathcal{A}$ be the algorithm specified in Alg. 1. Then, each deduced hyperarc is added (cf. e.g. line 2.20) to the set of hyperarcs E only once.*

Within the context of an inference rule with two premises, the above property stems from the fact that the rule's application over two particular items v and w is only considered when the greater of them (i.e. v if $w < v$[12] and w otherwise) is removed from the queue.

Algorithm 2 shows the procedure responsible for determining all the instantiations of the AX rule at the very beginning of the parsing process. It does not differ considerably from its formal, deductive description in Tab. 7.2.

Algorithm 3 Determine the instantiations of the PS inference rule with respect to the given passive item.

24: **procedure** APPLY-PS(x)
25: $\langle v, \langle i', j', k', l' \rangle \rangle_p \leftarrow x$
26: **for** $y \in \{\langle q, \langle i, j, k, l \rangle \rangle_a \in C : l = i' \wedge \delta(q, v)$ defined$\}$ **do**
27: $z \leftarrow \langle \delta(q, v), \langle i, j \cup j', k \cup k', l' \rangle \rangle_a$
28: $Q \leftarrow Q \cup \{z\}$
29: $E \leftarrow E \cup \{\langle \{x, y\}, z \rangle\}$
30: **end for**
31: **end procedure**

Algorithm 3 shows the procedure responsible for determining all the PS instantiations with respect to the given passive item. In particular, the procedure searches for all the active items in C (i) which end at the same position as the given passive item begins, and (ii) which correspond to ET traversals awaiting precisely the node provided by the passive item. Its time complexity relies on the computation cost of the line 3.26. The desired behavior is that the creation of each PS instantiation (as well as the instantiations of the other types) is constant, thus the computation cost of this line should be linear w.r.t. the number of the matching active items y.

To this end, the individual active items in C are indexed by (i) their ending positions and (ii) their FSA states. Then, when looking for the active items which can PS-match with the given passive item $\langle v, r \rangle_p$, the following procedure is used:

1. The set of active items in C ending at the position $beg(r)$ is retrieved. More precisely, a map M_1 from FSA states to such items is determined.

[12] Which should be understood as $w \leq v \wedge \neg(v \leq w)$.

2. The set of the FSA states Q_2, from which a transition labeled with the the DAG vertex v leaves, is determined. In practice, a map which allows to determine Q_2 for the individual vertices $v \in V$ is pre-computed on the basis of the underlying FSA family.

3. The intersection between the key-set in M_1 and the set Q_2 is calculated.

4. Finally, for each FSA state q in this intersection, all the corresponding active items (i.e., $M_1(q)$) are retrieved.

Note that, assuming the simple DAG and FSA encodings, the size of Q_2 is at most 1, and the above procedure can be implemented so as to achieve the desired, linear behavior w.r.t the number of the retrieved active items.

The implementation of the other, binary inference rules – SU, RA, and FA – follows the same pattern: when an item is removed from the queue, the parser looks for the corresponding premise items using appropriate indexing strategies to minimize the lookup cost. In contrast, the implementation of the remaining, unary rules, is rather straightforward.

7.3.7 Weighted inference rules

A weighted parsing system is an extension of a parsing system in which the individual inference rules are annotated with functions which allow to calculate the weight of the conclusion item on the basis of the weights of the premise items. Formally, such weighted systems can be defined in the *weighted deductive framework* (Nederhof, 2003).

Definition 100 (weighted GDAG)**.** *Let D be a GDAG and $\omega_D \colon V_D \to \mathbb{R}$ be a partial function, defined over D's roots, which represents the weights assigned to the individual ETs. Then, we say that $\langle D, \omega_D \rangle$ is a* weighted GDAG*.*

Recall that each $r \in V_D \colon root_D(r)$ unambiguously determines the corresponding ET $est(r)$.

Proposition 22 (weighted DAG encoding)**.** *Let $\langle D, \omega_D \rangle$ be a weighted GDAG. Then, it unambiguously determines the corresponding weighted TAG $\langle G, \omega_G \rangle$. We call $\langle D, \omega_D \rangle$ a* weighted DAG encoding *of $\langle G, \omega_G \rangle$.*

In what follows, we assume a fixed weighted TAG $\langle G, \omega_G \rangle$, a particular weighted DAG encoding $\langle D, \omega_D \rangle$ of $\langle G, \omega_G \rangle$, a particular FSA encoding M of D, and a fixed input sentence s.

Tab. 7.3 specifies the weighted version of the parser described in Tab. 7.2, which models Def. 43 in that the weights assigned to the resulting hyperpaths (this notion will be introduced in Sec. 7.3.8) should correspond to the weights of the corresponding TAG derivations. Such a weighted parser can be formalized in terms of weighted inference rules:

Definition 101 (weighted inference rule)**.** *Let $\mathcal{I}$ be a domain of items and $d\colon (\mathcal{I} \to \mathbb{R}) \to (\mathcal{I} \times \mathbb{R})$ be a partial function. Then, we say that d is a* weighted inference rule.

The intuition behind a weighted inference rule is that, in comparison with a regular inference rule (cf. Def. 81):

- it takes as argument a set of items together with their corresponding weights ($\mathcal{I} \to \mathbb{R} \subset 2^{\mathcal{I} \times \mathcal{R}}$) rather than just a set of items ($2^{\mathcal{I}}$), and
- it not only calculates the resulting item, but also computes the corresponding weight.

For instance, the PS inference rule (as specified in Tab. 7.3) is a function d_{PS} which maps any given $g = \{\langle\langle q, \langle i, j, k, l\rangle\rangle_a, x_1\rangle, \langle\langle v, \langle l, j', k', l'\rangle\rangle_p, x_2\rangle\}$ to $\langle\langle \delta(q, v), \langle i, j \cup j', k \cup k', l'\rangle\rangle_a, x_1 + x_2\rangle$, provided that $\langle q, \langle i, j, k, l\rangle\rangle_a$, $\langle v, \langle l, j', k', l'\rangle\rangle_p \in \mathcal{I}$ and that $\delta(q, v)$ is defined. Otherwise, $d_{PS}(g)$ is not defined. Thus, $d_{PS}(\{\langle\langle NP_{45} \bullet VP_{46} \to S_{44}, \langle 1, 3\rangle\rangle_a, 1\rangle, \langle\langle VP_{46}, \langle 3, 8\rangle\rangle_p, 2\rangle\}) = \langle\langle NP_{45}\, VP_{46} \bullet \to S_{44}, \langle 1, 8\rangle\rangle_a, 3\rangle$, while $d_{PS}(\{\langle\langle \bullet NP_{45}\, VP_{46} \to S_{44}, \langle 1, 1\rangle\rangle_a, 0\rangle, \langle\langle NP_{29}, \langle 1, 3\rangle\rangle_p, 1\rangle\})$ is not defined (contrary to the SU rule, which would apply to the latter arguments – see Fig. 7.12).

Each weighted inference rule unambiguously determines the corresponding regular inference rule, thus the properties and algorithms described in the preceding sections apply within the context of weighted inference rules and weighted parsing systems as well.

Definition 102 (weighted parsing system)**.** *Let $\mathcal{I}$ be the domain of items and $\mathcal{D}$ be the set of weighted inference rules which serve to deduce items in $\mathcal{I}$. Then, we say that $\langle \mathcal{I}, \mathcal{D}\rangle$ is a* weighted parsing system*. We also call the system specified by the rules in Tab. 7.3 the* vanilla weighted parsing system.

In the majority of the weighted rules presented in Tab. 7.3, the weight computed for the conclusion item is simply a sum of the weights of the premise items. This corresponds to the fact that most of the rules bring

AX:	$\dfrac{}{0{:}\langle q_0,\langle i,i\rangle\rangle_a}$	$\begin{array}{c} i{\in}pos(s)\backslash\{n\} \\ q_0{\in}S_M \end{array}$
SC:	$\dfrac{x{:}\langle q,\langle i,j,k,l\rangle\rangle_a}{x{:}\langle\delta(q,v),\langle i,j,k,l{+}1\rangle\rangle_a}$	$\begin{array}{c} v{\in}leaves(s_{l+1}) \\ \delta(q,v) \text{ defined} \end{array}$
DE:	$\dfrac{x{:}\langle q,\langle i,j,k,l\rangle\rangle_a}{x{+}c{:}\langle v,\langle i,j,k,l\rangle\rangle_p}$	$\begin{array}{c} v{\in}heads(q) \\ c{=}[root(v)]\omega_D(tree^1(v)) \end{array}$
PS:	$\dfrac{x_1{:}\langle q,\langle i,j,k,l\rangle\rangle_a \quad x_2{:}\langle v,\langle l,j',k',l'\rangle\rangle_p}{x_1{+}x_2{:}\langle\delta(q,v),\langle i,j{\cup}j',k{\cup}k',l'\rangle\rangle_a}$	$\delta(q,v)$ defined
SU:	$\dfrac{x_1{:}\langle q,\langle i,j,k,l\rangle\rangle_a \quad x_2{:}\langle v,\langle l,l'\rangle\rangle_p}{x_1{+}x_2{:}\langle\delta(q,v'),\langle i,j,k,l'\rangle\rangle_a}$	$\begin{array}{c} v'{\in}leaves(\ell(v)){\wedge}\neg foot(v') \\ \delta(q,v') \text{ defined} \\ root(v) \end{array}$
FA:	$\dfrac{x_1{:}\langle q,\langle i,l\rangle\rangle_a \quad x_2{:}\langle v,\langle l,j',k',l'\rangle\rangle_p}{x_1{:}\langle\delta(q,v'),\langle i,l,l',l'\rangle\rangle_a}$	$\begin{array}{c} v'{\in}leaves(\ell(v)){\wedge}foot(v') \\ \delta(q,v') \text{ defined} \\ root(v) \Longrightarrow (j',k'){=}(-,-) \end{array}$
RA:	$\dfrac{x_1{:}\langle w,\langle i,j,k,l\rangle\rangle_p \quad x_2{:}\langle v,\langle j,j',k',k\rangle\rangle_p}{x_1{+}x_2{:}\langle v,\langle i,j',k',l\rangle\rangle_p}$	$\begin{array}{c} root(w){\wedge}(j,k){\neq}(-,-) \\ \ell(w){=}\ell(v) \\ root(v) \Longrightarrow (j',k'){=}(-,-) \end{array}$

Table 7.3: The weighted version of the vanilla parser from Tab. 7.2. To each item the corresponding weight, given before the colon, is assigned. Recall that $\omega_D(tree^1(v))$ refers to the grammar-specified weight of the $tree^1(v)$. $[x]$ is equal to 1 if x is TRUE and to 0 otherwise.

no additional weight. The two exceptions are the deactivation rule, which adds the weight of the corresponding ET, and the foot-adjoin rule, which transfers only the weight of the active item, in accordance with the fact that the internal derivation of the witness item is irrelevant.

7.3.8 Weighted hypergraphs

Definition 103 (root arc)**.** *Let $\mathcal{P}$ be a hyperpath. We define its* root arc*, denoted $rootarc(\mathcal{P})$, as the arc $e \in E_{\mathcal{P}}$ such that $head(e) = root(\mathcal{P})$ (see also Def. 86 and Def. 92).*

Definition 104 (weighted hypergraph)**.** *Let $\langle \mathcal{I}, \mathcal{D} \rangle$ be a weighted parsing system and $\mathcal{H} = \langle V_{\mathcal{H}}, E_{\mathcal{H}} \rangle$ be the corresponding hypergraph.*[13] *We define the corresponding* weighted hypergraph *as a pair $\langle \mathcal{H}, \omega_{\mathcal{H}} \rangle$ such that $\omega_{\mathcal{H}}$ is a function which assigns the weights to the individual hyperpaths $\mathcal{P} \in \mathcal{H}$. Formally, let $\mathcal{P} \in \mathcal{H}$ be a hyperpath, $\langle R, r \rangle = rootarc(\mathcal{P})$, and $d \in \mathcal{D}$ be such that d infers r from R. Then:*

$$\omega_{\mathcal{H}}(\mathcal{P}) = d(\{\langle w, \omega_{\mathcal{H}}(\mathcal{P}') \rangle : w \in R, \mathcal{P}' \in ins_{\mathcal{P}}(w)\})_2.$$

Recall that $ins_{\mathcal{P}}(w)$ is the set of hyperpaths leading to $w \in V_{\mathcal{P}}$ in the hypergraph $\mathcal{P}$ (cf. Eq. 7.5). Since $\mathcal{P}$ is a hyperpath itself, it contains only one hyperpath $\mathcal{P}'$ leading to w and, therefore, the argument of d is a function (as required) which assigns the weights to the individual tail items $w \in V_{\mathcal{P}}$.

Fig. 7.12 illustrates a version of the hyperpath shown in Fig. 7.8 in which the weights $\omega_{\mathcal{H}}(\mathcal{P})$ computed for the individual hyperpaths $\mathcal{P}$ (on the basis of the rules presented in Tab. 7.3) are shown next to their roots. The weights are based on the assumption that all ETs have the weight of 1, hence the weight of the hyperpath leading to $\langle VP_{46}, \langle 3, 8 \rangle \rangle$ is 2 (*decisions* is modified by *a few* and *good*), and the weight of the hyperpath leading to $\langle NP_{29}, \langle 1, 3 \rangle \rangle$ is 1 (*prime minister* is analysed as a MWE).

Proposition 23 (correspondence between hyperpath and derivation weights)**.** *Let $\langle \mathcal{I}, \mathcal{D} \rangle$ be a vanilla weighted parsing system and $\mathcal{W} = \langle \mathcal{H}, \omega_{\mathcal{H}} \rangle$ be the corresponding weighted parsing hypergraph. Then, for each $\mathcal{P} \in \mathcal{H}$ and the corresponding TAG derivation δ (cf. Prop. 17, Prop. 18, and Prop. 19) it holds that $\omega_{\mathcal{H}}(\mathcal{P}) = \omega_G(\delta)$ (see also Def. 43).*

[13] Note that a weighted parsing system unambiguously determines the corresponding non-weighted parsing system.

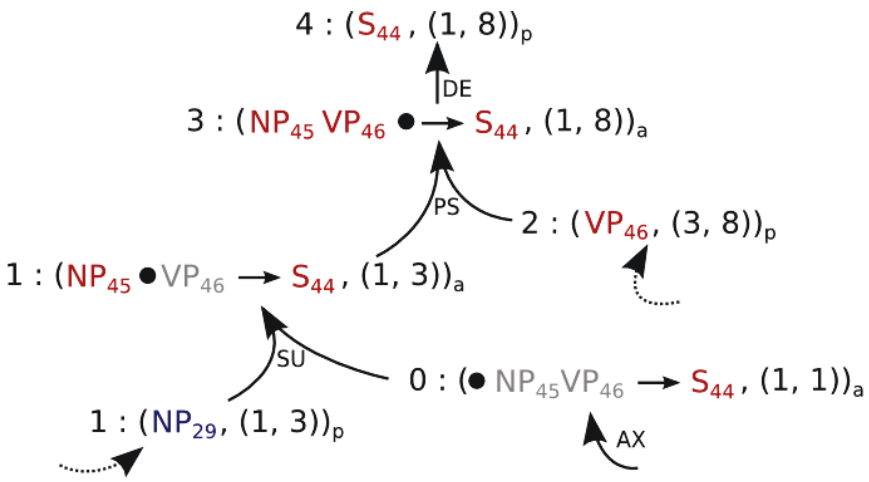


Figure 7.12: A weighted version of the hyperpath presented in Fig. 7.8 in which the weights computed for the individual hyperpaths are shown on the left of their roots.

Definition 105 (monotonicity)**.** *Let $\mathcal{W} = \langle \mathcal{H}, \omega_{\mathcal{H}} \rangle$ be a weighted hypergraph. Then, we say that $\omega_{\mathcal{H}}$ is* monotonic *iff:*

$$\forall_{\mathcal{P}' \in \mathcal{H}} \forall_{\mathcal{P} \subseteq \mathcal{P}'} \Big(\omega_{\mathcal{H}}(\mathcal{P}) \leq \omega_{\mathcal{H}}(\mathcal{P}') \Big). \tag{7.8}$$

We also say that a weighted parsing system is monotonic *if the corresponding weighted hypergraph is monotonic.*

The above definition states that the weight $\omega_{\mathcal{H}}(\mathcal{P}')$ of any hyperpath $\mathcal{P}' \in \mathcal{H}$ is equal or greater than the weight $\omega_{\mathcal{H}}(\mathcal{P})$ of any hyperpath $\mathcal{P}$ contained in $\mathcal{P}'$. Note that an equivalent definition is obtained by requiring that only the weights $\omega_{\mathcal{H}}(\mathcal{P})$ of the paths $\mathcal{P} \subseteq \mathcal{P}'$ such that $root(\mathcal{P}) \in tail(rootarc(\mathcal{P}'))$ are equal or smaller than $\omega_{\mathcal{H}}(\mathcal{P}')$.

Proposition 24. *Let $\mathcal{W} = \langle \mathcal{H}, \omega_{\mathcal{H}} \rangle$ be a weighted hypergraph. Then, $\omega_{\mathcal{H}}$ is monotonic iff:*

$$\forall_{\mathcal{P}' \in \mathcal{H}} \forall_{\mathcal{P} \subseteq \mathcal{P}'} \Big(root(\mathcal{P}) \in tail(rootarc(\mathcal{P}')) \implies \omega_{\mathcal{H}}(\mathcal{P}) \leq \omega_{\mathcal{H}}(\mathcal{P}') \Big). \tag{7.9}$$

Proposition 25. *Let $\langle \mathcal{I}, \mathcal{D} \rangle$ be a vanilla weighted parsing system. Then, $\langle \mathcal{I}, \mathcal{D} \rangle$ is* not *monotonic.*

First of all, a vanilla weighted parsing system is not guaranteed to be monotonic because the weights assigned to the individual ETs in the grammar can be, in general, negative. In such a case, the weight the DE rule assigns to the conclusion item can be lower than the weight of the premise item. Moreover, even when all the ET weights are non-negative, the FA inference rule violates the required property: the resulting weight x_1 can be smaller than the weight x_2 assigned to the witness item (cf. Tab. 7.3). In both cases, Eq. 7.8 does not hold.

Another definition of monotonicity, which we will call *HC-monotonicity*, can be found in (Huang and Chiang, 2005). A HC-monotonic parsing system exhibits the property that, if one of the weights assigned to the tail items increases, then the weight resulting from applying a weighted inference rule d should not decrease either. This property holds for all the weighted parsing systems described in this work and, henceforth, we will implicitly assume that it is not violated, unless stated otherwise.

Just as it is possible to retrieve all the final hyperpaths from a given hypergraph, it is also possible to retrieve only the least-weight (shortest) final hyperpath[14]. It can be done with a version of Eq. 7.5 (p. 134) which, for a given node $v \in V_{\mathcal{H}}$, searches recursively for the shortest hyperpath amongst the shortest paths determined for the individual hyperarcs $e \in E_{\mathcal{H}}$ such that $head(e) = v$.

The problem of finding the shortest path in a hypergraph has been already studied in (Gallo et al., 1993). Its extended version, where the goal is to find the k-shortest paths embedded in a hypergraph, has been investigated in (Nielsen et al., 2005) and (Huang and Chiang, 2005). In particular, the solution described in the latter work allows negative weights (i.e., non-monotonic parsing systems) and, therefore, it could be used within the context of the weighted vanilla parser even if some of the weights assigned to the individual ETs were negative (cf. the rule DE in Tab. 7.3).

7.4 A* parsing with MWE-driven heuristic

As mentioned in Sec. 7.3.4, the basic hypergraph-based parsing algorithm creates the entire parsing hypergraph for a given parsing system and, then, selects the optimal (i.e. the shortest) hyperpath amongst all the final paths encoded in the graph. Assuming the vanilla parser, the memory complexity

[14]Provided that the underlying weighted parsing system is HC-monotonic

of this solution is $\mathcal{O}(n^6)$, where n is the length of the input sentence, since $\mathcal{O}(n^6)$ is the number of hyperarcs which can be potentially created in the vanilla parsing system.[15] The memory complexity can be reduced to $\mathcal{O}(n^4)$ if only the shortest hyperpath is searched for – only a single, optimal incoming hyperarc has to be stored per node then, and there are $\mathcal{O}(n^4)$ chart items which can be potentially created by the parser.

An alternative parsing architecture, based on the classical A* algorithm, is based on the principle that the shortest hyperpath can be found without creating the entire hypergraph corresponding to a given parsing system. Rather, the parser strives to visit only the nodes and arcs belonging to the shortest path. To this end, it exploits a heuristic which estimates the cost of parsing the remaining part of the input sentence for each constructed item. The idea of A* parsing has been introduced in the work of Klein and Manning (2003) and later extended to other formalisms (Lewis and Steedman, 2014; Angelov and Ljunglöf, 2014) as well as to k-best parsing (Pauls and Klein, 2009).

Definition 106 (inside weight)**.** *Let $\mathcal{W} = \langle \mathcal{H}, \omega_\mathcal{H} \rangle$ be a weighted parsing hypergraph and $x \in V_\mathcal{H} \cup E_\mathcal{H}$. Then, we define the* inside weight *of x as the weight of its optimal inside derivation (cf. Def. 97):*

$$\beta(x) = \min_{\mathcal{P} \in ins(x)} \omega_\mathcal{H}(\mathcal{P})$$

Definition 107 (crossing derivation)**.** *Let $\mathcal{H}$ be a hypergraph and $v \in V_\mathcal{H}$. Then, we define a* crossing derivation *of v as any of the final hyperpaths $\mathcal{P} \in \mathcal{H}$ such that v belongs to $\mathcal{P}$. Formally, the set of v's crossing derivations is:*

$$cross(v) = \{\mathcal{P} : \mathcal{P} \in \mathcal{H}, final(\mathcal{P}), v \in V_\mathcal{P}\}$$

Similarly, we define the set of the crossing derivation of a given $e \in E_\mathcal{H}$ as:

$$cross(e) = \{\mathcal{P} : \mathcal{P} \in \mathcal{H}, final(\mathcal{P}), e \in E_\mathcal{P}\}$$

and a crossing derivation *of e as a member of this set.*

Definition 108 (crossing weight)**.** *Let $\mathcal{W} = \langle \mathcal{H}, \omega_\mathcal{H} \rangle$ be a weighted parsing hypergraph and $x \in V_\mathcal{H} \cup E_\mathcal{H}$. Then, we define the* crossing weight *of x as the weight of its optimal crossing derivation:*

$$\gamma(x) = \min_{\mathcal{P} \in cross(x)} \omega_\mathcal{H}(\mathcal{P})$$

[15] There are 6 free positions – i, j, k, l, j', l' – in the premise items of the rule RA in Tab. 7.2. The upper bound on the size of the set of the possible instantiations of this rule is thus $\mathcal{O}(n^6)$. Instantiations of the other rules are more constrained.

Definition 109 (outside derivation)**.** *Let $\mathcal{H}$ be a hypergraph and $v \in V_{\mathcal{H}}$. Then, we define an* outside derivation *of v as any of the final hyperpaths $\mathcal{P}' \in \mathcal{H}$ such that v belongs to $\mathcal{P}'$, accompanied with the corresponding inside derivation of v. Formally, the set of outside derivations of v is:*

$$out(v) = \{(\mathcal{P}', \mathcal{P}) : \mathcal{P}' \in \mathcal{H}, final(\mathcal{P}'), \mathcal{P} \subseteq \mathcal{P}', root(\mathcal{P}) = v\}$$

Similarly, we define the set of the outside derivations of a given $e \in E_{\mathcal{H}}$ as:

$$out(e) = \{(\mathcal{P}', \mathcal{P}) : \mathcal{P}' \in \mathcal{H}, final(\mathcal{P}'), \mathcal{P} \subseteq \mathcal{P}', rootarc(\mathcal{P}) = e\}$$

and an outside derivation *of e as a member of this set.*[16]

Definition 110 (outside weight)**.** *Let $\mathcal{W} = \langle \mathcal{H}, \omega_{\mathcal{H}} \rangle$ be a weighted parsing hypergraph and $x \in V_{\mathcal{H}} \cup E_{\mathcal{H}}$. Then, we define the* outside weight *of x as:*

$$\alpha(x) = \min_{(\mathcal{P}, \mathcal{P}') \in out(x)} \Big(\omega_{\mathcal{H}}(\mathcal{P}) - \omega_{\mathcal{H}}(\mathcal{P}') \Big)$$

Proposition 26. *Let $\mathcal{W} = \langle \mathcal{H}, \omega_{\mathcal{H}} \rangle$ be a weighted hypergraph corresponding to a weighted parsing system and $x \in V_{\mathcal{H}} \cup E_{\mathcal{H}}$. Then, it holds that $\gamma(x) = \beta(x) + \alpha(x)$.*[17]

According to the above proposition, the weight of any optimal, final hyperpath can be divided into two parts for any of its component hypernodes v (hyperarcs e, respectively). Namely, $\beta(v)$ represents the minimal weight of reaching the node v (hyperarc e), while $\alpha(v)$ represents the minimal weight of reaching a final node from v (e).

Proposition 27. *Let $\mathcal{W} = \langle \mathcal{H}, \omega_{\mathcal{H}} \rangle$ be a weighted parsing hypergraph and $x \in V_{\mathcal{H}}$ be a final node. Then, $\alpha(x) = 0$ and $\beta(x) = \gamma(x)$ is equal to the weight of the optimal full derivation ending in x.*

7.4.1 Dijkstra-style parsing

A Dijkstra-style parsing algorithm, presented in Alg. 4, is similar to Alg. 1. The notable differences are that:

[16] It is tempting to say that an outside derivation is a difference between $\mathcal{P}'$ and $\mathcal{P}$, which could be defined as a set difference between the corresponding sets of hyperarcs. However, the result of such an operation does not necessarily satisfy all the properties of a hyperpath.

[17] This property does not necessarily hold for non-HC-monotonic parsing systems.

- To each item v in the priority queue Q the estimation $\hat{\beta}(v)$ of its inside weight $\beta(v)$ is assigned.
- The total order over items (used to arrange them in Q) is based on the estimations of their inside weights rather than on their spans.
- The algorithm stops when the first final item is found.
- C is a map from closed items to their β values.
- The order in which the items are removed from Q cannot be controlled statically anymore. Recall that, in Alg. 1, this order depends on the span values of the individual items. When the items are removed from Q according to their β weights, it is no longer possible to enforce that, e.g., the passive premise item of the SU rule is always removed from Q after the corresponding active premise item. Thus, the reversed versions of the binary procedures APPLY-SU, APPLY-PS, etc., have to be implemented.

The estimations are computed using the weighted inference rules and when an item is inferred which has been already deduced before, the minimum of the computed weights is preserved. This parsing algorithm can be seen as an adaptation of the algorithm specified in (Nederhof, 2003, p. 140) to a particular formalism and to particular inference rules.

Alg. 6 shows a reversed variant of the procedure shown in Alg. 5, which looks for the corresponding passive items given an active item. Its efficiency relies on the computation cost of the line 32, which could be implemented as follows.[18] Given an active item $\langle q, r \rangle_a$:

1. The set S_1 of passive items in C beginning at the position $end(r)$ is retrieved. More precisely, a map $M_1 \colon V_D \to 2^{S_1}$ which indexes the items in S_1 by the DAG vertices they contain is determined.
2. The set of transition symbols $V_2 \subseteq V_D$ outgoing from q is determined.
3. The intersection between the key-set in M_1 and V_2 is calculated.

[18] The actual implementation in ParTAGe does not exactly follow this procedure. It first picks one of the transition symbols $v \in V_D$ outgoing from the state q and then looks for the passive items with v and placed on the right. Within the context of the simple FSA encoding this method is sufficiently efficient, but it does not compose nicely with grammar compression techniques, where a large number of transitions can leave a given state q.

Algorithm 4 Dijkstra-style parsing algorithm

1: $Q \leftarrow \emptyset$ ▷ The priority queue: function from open items to $\hat{\beta}$ values
2: $C \leftarrow \emptyset$ ▷ The function from closed items to β values
3: $E \leftarrow \emptyset$ ▷ The set of hyperarcs
4: stop $\leftarrow false$ ▷ When to stop the algorithm
5: APPLY-AX
6: **while** ¬stop **do**
7: $(x, w) \leftarrow$ DELETE-MIN(Q)
8: $C \leftarrow C \cup \{(x, w)\}$
9: **if** FINAL(x) **then**
10: stop $\leftarrow true$
11: **else**
12: APPLY-PS(x,w); APPLY-SU(x,w); APPLY-FA(x,w)
13: APPLY-RA(x,w); APPLY-SC(x,w); APPLY-DE(x,w)
14: ▷ Below, the parser considers the reversed rule applications for all the inference rules with two premise items, whose processing order cannot be controled anymore.
15: APPLY-PS-REV(x,w); APPLY-SU-REV(x,w)
16: APPLY-FA-REV(x,w); APPLY-RA-REV(x,w)
17: **end if**
18: **end while**

4. Finally, for each vertex v in this intersection, all the corresponding active items ($M_1(v)$) are retrieved.

Algorithm 5 Dijkstra-style parsing: determine the instantiations of the PS inference rule with respect to the given passive item.

procedure APPLY-PS(x, w_x)
  guard (passive x) ▷ Process the item only if passive
  $\langle v, \langle i', j', k', l' \rangle\rangle_p \leftarrow x$
  for $y \in \{(\langle q, \langle i, j, k, l \rangle\rangle_a, w_y) \in C : l = i' \wedge \delta(q, v)$ defined$\}$ **do**
    $z \leftarrow \langle \delta(q, v), \langle i, j \cup j', k \cup k', l' \rangle\rangle_a$
    $w_z \leftarrow w_x + w_y$
    $Q \leftarrow Q \cup_{min} \{(z, w_z)\}$ ▷ If item z is already present in Q, the minimum of $Q(z)$ and w_z is preserved.
    $E \leftarrow E \cup \{(\{x, y\}, z)\}$
  end for
end procedure

Algorithm 6 Dijkstra-style parsing: the reversed variant of APPLY-PS.

procedure APPLY-PS-REV(y, w_y)
  guard (active y)
  $\langle q, \langle i, j, k, l \rangle\rangle_a \leftarrow y$
  for $x \in \{(\langle v, \langle i', j', k', l' \rangle\rangle_p, w_x) \in C : l = i' \wedge \delta(q, v)$ defined$\}$ **do**
    $z \leftarrow \langle \delta(q, v), \langle i, j \cup j', k \cup k', l' \rangle\rangle_a$
    $w_z \leftarrow w_x + w_y$
    $Q \leftarrow Q \cup_{min} \{(z, w_z)\}$
    $E \leftarrow E \cup \{(\{x, y\}, z)\}$
  end for
end procedure

Definition 111 (correctness)**.** *Let $\langle \mathcal{I}, \mathcal{D} \rangle$ be a weighted parsing system and $\mathcal{A}$ be a parsing algorithm (e.g., the algorithm specified in Alg. 4). Then, $\mathcal{A}$ is* correct *iff it holds that when a final item is removed from the queue, the underlying hypergraph*[19] *contains an optimal hyperpath.*

[19] I.e., the hypergraph induced by the set of hyperarcs E maintained by the algorithm.

Consequently, when a parsing algorithm is correct, the resulting hypergraph is guaranteed to encode an optimal TAG derivation tree covering the input sentence.

Proposition 28. *Let $\langle \mathcal{I}, \mathcal{D} \rangle$ be a monotonic parsing system (cf. Def. 105) and $\mathcal{A}$ be the Dijkstra-style parsing algorithm specified in Alg. 4. Then, $\mathcal{A}$ exhibits the following properties:*

- *When an item v is removed from the priority queue Q (cf. line 7 of Alg. 4), its estimated $\hat{\beta}(v)$ (stored as the priority in Q) equals $\beta(v)$.*
- *Items are removed from Q in an ascending order of their inside weights.*

As a result, when a final item is removed from the queue, the parsing shortest-path algorithm can be stopped since we know that no other final items v with lower $\beta(v) + \alpha(v)$ can be found anymore (recall that $\alpha(v) = 0$ since v is final). Then, based on the items stored in the closed set C and the hyperarcs stored in E, it is possible to restore an optimal inside derivation of v using the methods mentioned in Sec. 7.3.8.

Proposition 29. *Let $\langle \mathcal{I}, \mathcal{D} \rangle$ be a weighted parsing system and $\mathcal{A}$ be the Dijkstra-style parsing algorithm specified in Alg. 4. Then,*

$$\langle \mathcal{I}, \mathcal{D} \rangle \text{ is monotonic } \implies \mathcal{A} \text{ is correct.}$$

Finally, let us recall that the vanilla weighted parsing system is *not* monotonic (cf. Prop. 25). Alg. 4 can be used with this system but does not guarantee that the first final item removed from the queue is optimal.[20]

7.4.2 A* parsing

The Dijkstra-style parsing algorithm provides a speed-up with respect to the standard Alg. 1 when only the least-weight derivation is searched for. The A* parsing algorithm provides a further improvement of this idea, which relies on an estimation of the α function.

Let us imagine that we know in advance the $\beta(v)$ and $\alpha(v)$ values for the individual nodes v in the weighted parsing hypergraph. Then, we could

[20] Recall that the vanilla weighted parser is HC-monotonic, thus it is still possible to first create the entire hypergraph and then to retrieve the 1 or k shortest hyperpaths.

simply use a version of Alg. 4 where the total order defined over the items corresponds to their $\beta(v) + \alpha(v)$ values. Under such conditions, the parser would actually never visit (i.e. remove from the queue) any node whose total weight exceeds the optimal $\beta(v) + \alpha(v)$, which underlies the idea of A* parsing.

The values of β, as mentioned above, can be computed directly using the weighted inference rules. However, the values of α are unknown, since they refer to the parts of the hypergraph which are yet to be found and connected. Thus the A* parsing algorithm relies on their estimation, called A* heuristic.

Definition 112 (A* heuristic)**.** *Let $\mathcal{W} = \langle \mathcal{H}, \omega_\mathcal{H} \rangle$ be a weighted parsing hypergraph and $x \in V_\mathcal{H} \cup E_\mathcal{H}$. Let $h(x)$ be an estimation of $\alpha(x)$. Then, we call h an* A* heuristic *(or* heuristic *for short).*

Definition 113 (admissible heuristic)**.** *Let $\mathcal{W} = \langle \mathcal{H}, \omega_\mathcal{H} \rangle$ be a weighted parsing hypergraph and h be a heuristic. Then, we say that h is* admissible *iff it never over-estimates the value of α. Formally, h is admissible iff:*

$$\forall_{x \in V_\mathcal{H} \cup E_\mathcal{H}} (h(x) \leq \alpha(x))$$

Definition 114 (monotonic heuristic)**.** *Let $\mathcal{W} = \langle \mathcal{H}, \omega_\mathcal{H} \rangle$ be a weighted parsing hypergraph and h be a heuristic. Then, we say that h is* monotonic *iff:*

$$\forall_{e \in E_\mathcal{H}} \forall_{v \in tail(e)} \Big(\beta(v) + h(v) \leq \beta(e) + h(e) \Big). \tag{7.10}$$

The definition of a monotonic heuristic is strongly related to the definition of a monotonic weighted hypergraph (cf. Def. 105 and Prop. 24). The main differences are that: (i) Eq. 7.9 refers to hyperpath weights in general, while Eq. 7.10 focuses on the weights of the optimal hyperpaths exclusively, and (ii) Eq. 7.9 does not take the values of the heuristic into account.

Definition 115 (A* parsing algorithm)**.** *Let $\mathcal{A}$ be a version of Alg. 4 in which the total order over items $x \in \mathcal{I}$ (used to arrange them in Q) is based on their $\hat{\beta}(x) + h(x)$ weights. Then, we call $\mathcal{A}$ the* A* parsing algorithm*.*

Proposition 30. *Let $\langle \mathcal{I}, \mathcal{D} \rangle$ be a weighted parsing system, $\mathcal{W} = \langle \mathcal{H}, \omega_\mathcal{H} \rangle$ be the corresponding weighted hypergraph, h be a monotonic, admissible heuristic, and $\mathcal{A}$ be the A* parsing algorithm. Then, $\mathcal{A}$ can be used to dynamically construct $\mathcal{W}$ and to find a shortest path therein. Moreover, $\mathcal{A}$ satisfies the following properties, analogous to those described in Prop. 28:*

Figure 7.13: A hypothetical parsing configuration considered by the parser: *prime minister* analysed as a MWE, the cost of parsing the remaining part of the sentence to be determined.

- *When an item v is removed from Q, its estimated $\hat{\beta}(v)$ equals $\beta(v)$.*
- *Items are removed from Q in an ascending order of their $\beta + h$ weights.*

Proposition 31. *Let $\langle \mathcal{I}, \mathcal{D} \rangle$ be a weighted parsing system, h be a heuristic, and $\mathcal{A}$ be the A⋆ parsing algorithm. Then,*

$$h \text{ is monotonic} \wedge h \text{ is admissible} \implies \mathcal{A} \text{ is correct.}$$

7.4.3 MWE-driven A⋆ heuristic

Our goal is therefore to estimate $\alpha(v)$, the weight remaining to parse the entire input sentence, for any $v \in V_{\mathcal{H}}$. Given the close relation between hyperpaths and TAG derivation, this task can be cast back in the realm of TAG derivation trees. Fig. 7.13 illustrates a hypothetical parsing configuration considered by the parser, represented by a passive item $\langle \text{NP}_{29}, \langle 1, 3 \rangle \rangle_p$. The inside weight is already known and it is equal to 1, since we assume here that all the ETs are assigned weight 1 and since the EST rooted in NP_{29} matches the entire span $\langle 1, 3 \rangle$.

As mentioned before, our point of departure is the work of Lewis and Steedman (2014), where the weight of parsing the remaining part of the sentence is the sum of the (minimal) weights of the elementary grammar units (EUs) which can analyse the individual words in this part. In this formalization, the structure of each EU is abstracted over and the only pieces of information which matter are (i) to which word in the input sentence this EU is attached, and (ii) what its weight is.

In our probabilistic formalization we also assume that the weight of the final derivation is the sum of the weights of the participating ETs. If there

were no multi-anchored ETs in the grammar, it would be possible to use the same heuristic as the one described in Lewis and Steedman (2014).

Proposition 32. *Let $\langle G, \omega_G \rangle$ be a weighted lexicalized TAG such that $\omega_G(t) \geq 0$ for each ET t in G and there are no multi-anchored ETs in G. Let also $\langle \mathcal{I}, \mathcal{D} \rangle$ be a vanilla weighted parsing system parameterized by $\langle G, \omega_G \rangle$, $\mathcal{W} = \langle \mathcal{H}, \omega_{\mathcal{H}} \rangle$ be the corresponding weighted hypergraph, and $x \in V_{\mathcal{H}}$. Then, the heuristic defined as:*

$$h(x) = \sum_{(w,k) \in rest(x)} minw(w) \times k,$$

where:

- *$rest(x)$ is the bag of words in the input sentence outside of x's span (see Def. 121 for a more precise definition),*
- *$(w, k) \in rest(x)$ is a word and its multiplicity in a bag,*
- *$minw(w) = min\{\omega_G(t) : t \in I_G \cup A_G, anchor(t) = w\}$ denotes the minimal weight of scanning w by an ET,*
- *$anchor(t)$ is a terminal of the given ET t,*

is an admissible (never overestimating) heuristic.

For instance, assuming the grammar from Fig. 7.3 restricted to mono-anchored ETs and that each ET has weight 1, $minw(w) = 1$ for each terminal w in the grammar. Furthermore, let $x = \langle \mathrm{NP}_{29}, \langle 1, 3 \rangle \rangle_p$ be the item representing the parsing configuration shown in Fig. 7.13. Then, $rest(x) = \{\text{the}, \text{made}, \text{a}, \text{few}, \text{good}, \text{decisions}\}_{ms}$.

This idea can be extended to a MWE-aware context in a way which does not entail significant computational overhead. Fig. 7.14 shows how the weights of the individual ETs in a given, MWE-driven TAG derivation can be projected over the words of the input sentence. Fig. 7.15 shows another, more compositional derivation and the corresponding weights. In both cases, the weight of the derivation is equal to the sum of the weights projected on the individual words in the sentence.

When we face the task to estimate α for an item ranging over a span $\langle i, j, k, l \rangle$ in sentence s, the weights projected over the individual words are, of course, unknown, since they depend on the particular derivation covering the entire sentence. However, based on the underlying grammar and the weights

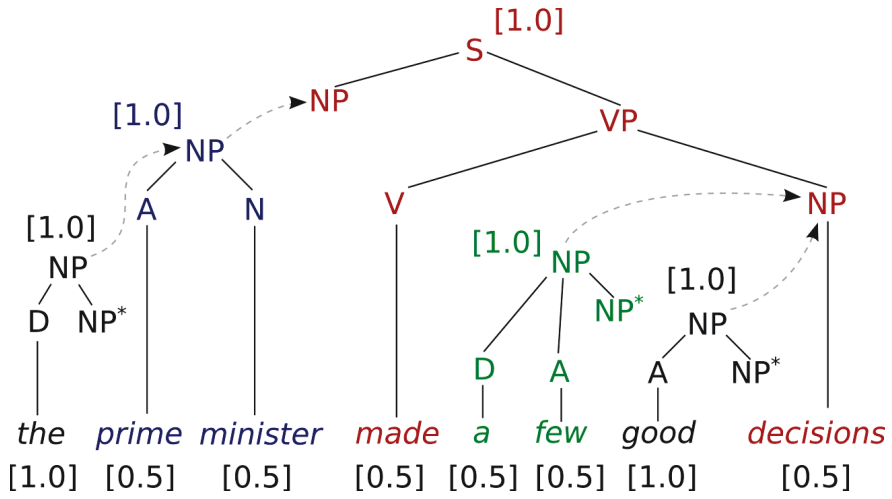


Figure 7.14: Example of projecting the weights of the ETs on the corresponding terminals.

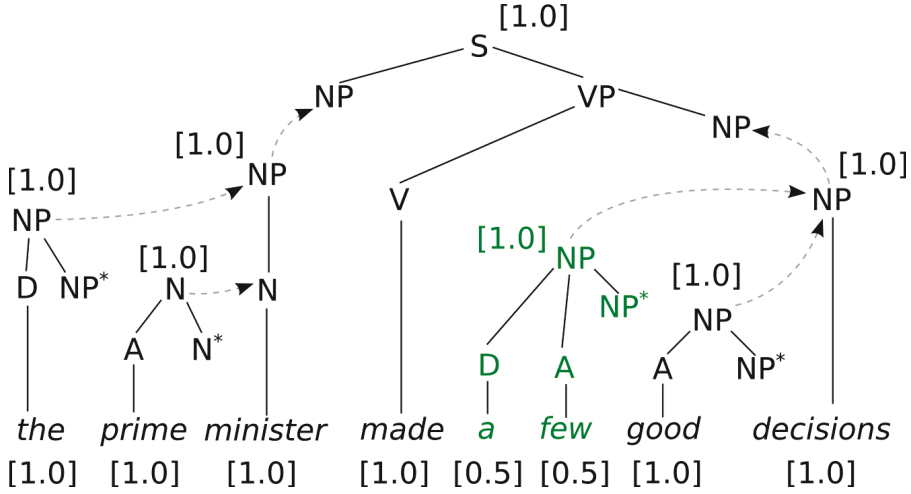


Figure 7.15: Example of projecting the weights of the ETs on the corresponding terminals.

assigned to the individual ETs, it is possible to estimate the minimal weight which can be projected over each of the words outside the span $\langle i, j, k, l\rangle$. This, in turn, allows to compute a lower-bound estimate of the weight of parsing the remaining part of the input sentence.

Definition 116 (scanning weight)**.** *Let $\langle G, \omega\rangle$ be a weighted TAG and $w \in \Sigma_G$. Then, we define the* minimal scanning weight *of w by an ET (*scanning weight *henceforth) as the minimum proportion of w among all terminals of a single ET. More precisely,*

$$minw(w) = \min_{t \in I_G \cup A_G : (w,i) \in sub_G(t)} \frac{\omega_G(t)}{|sub_G(t)|}. \tag{7.11}$$

For instance, assuming the grammar from Fig. 7.3 and that each ET has the weight 1, $minw(\text{the}) = 1$ (terminal *the* is present only in a single, mono-anchored ET), $minw(\text{prime}) = 0.5$ (*prime* is present in the MWE ET *prime minister*), $minw(\text{made}) = 0.5$, etc. If there were an ET corresponding to the MWE *take into account* in this grammar, then $minw(\text{take}) = minw(\text{into}) = minw(\text{acount}) = \frac{1}{3}$.

Variants of the $minw(w)$ definition include distributing the weights of individual terminals in an ET proportionally to their frequencies in the corpus. Our experiments (cf. Ch. 8) did not show any advantage of such a distribution over the uniform one.

Definition 117 (scanning cost)**.** *Let $m \in \mathcal{M}(\Sigma)$ be a multiset of words and minw be the scanning weight function. Then, $cost(m)$ represents the globally minimal cost of scanning all the words in m:*

$$cost(m) = \sum_{(x,k) \in m} minw(x) \times k \tag{7.12}$$

For instance, assuming the grammar from Fig. 7.3 and that each ET has the weight 1, $cost(\{\text{the}, \text{made}, \text{a}, \text{few}, \text{good}, \text{decisions}\}_{ms}) = 1+0.5+0.5+0.5+0.5+0.5 = 3.5$ (note that the argument corresponds to the words remaining to be parsed in the parsing configuration shown in Fig. 7.13).

Definition 118 (span words)**.** *Let $r = \langle i, j, k, l\rangle$ be a span in the sentence s. Then, $mid(r)$ represents the multiset of words covered by r (cf. Def. 73):*

$$mid(\langle i, j, k, l\rangle) = \begin{cases} \{s_{i+1}, \ldots, s_j, s_{k+1}, \ldots, s_l\}_{ms} & \text{if } (j,k) \neq (-,-) \\ \{s_{i+1}, \ldots, s_l\}_{ms} & \text{otherwise} \end{cases} \tag{7.13}$$

For instance, assuming the sentence from Fig. 7.14, $mid(\langle 1,3\rangle) = \{\text{prime, minister}\}_{ms}$, $mid(\langle 0,8\rangle) = \{\text{the, prime, minister, made, a, few, good, decisions}\}_{ms}$, $mid(\langle 0,1,3,8\rangle) = \{\text{the, made, a, few, good, decisions}\}_{ms}$, etc.

Definition 119 (gap words)**.** *Let s be an input sentence and $r = \langle i,j,k,l\rangle$ be a span in s. Then, $in(r)$ represents the multiset of words covered by r's gap:*

$$in(\langle i,j,k,l\rangle) = \begin{cases} \{s_{j+1},\ldots,s_k\}_{ms} & \text{if } (j,k) \neq (-,-) \\ \emptyset & \text{otherwise} \end{cases} \tag{7.14}$$

For instance, assuming the sentence from Fig. 7.14, $in(\langle 0,8\rangle) = in(\langle 1,3\rangle) = \emptyset_{ms}$, $in(\langle 0,1,3,8\rangle) = \{\text{prime, minister}\}_{ms}$, etc.

Definition 120 (outer words)**.** *Let $\langle i,j,k,l\rangle$ be a span in s. Then, $out(r)$ represents the multiset of words outside (on the left and right of) r:*

$$out(\langle i,j,k,l\rangle)) = \{s_1,\ldots,s_i,s_{l+1},\ldots,s_{|s|}\}_{ms}. \tag{7.15}$$

For instance, assuming the sentence from Fig. 7.14, $out(\langle 0,8\rangle) = \emptyset_{ms}$, $out(\langle 1,3\rangle) = \{\text{the, made, a, few, good, decisions}\}_{ms}$, $out(\langle 0,1,3,8\rangle) = \emptyset_{ms}$, etc.

Proposition 33 (sentence coverage)**.** *Let r be a span in s. Then,*

$$in(r) \cup mid(r) \cup out(r) = \{s_i : i \in \{1\ldots|s|\}\}_{ms} \tag{7.16}$$

Definition 121 (remaining words)**.** *We define $rest(r)$ as a multiset of words in the input sentence s outside of the given span r, i.e.,*

$$rest(r) = out(r) \cup in(r) \tag{7.17}$$

For instance, assuming the sentence from Fig. 7.14, $rest(\langle 0,8\rangle) = \emptyset_{ms}$, $rest(\langle 1,3\rangle) = \{\text{the, made, a, few, good, decisions}\}_{ms}$, $rest(\langle 0,1,3,8\rangle) = \{\text{prime, minister}\}$, $rest(\langle 0,1,3,5\rangle) = \{\text{prime, minister, few, good, decisions}\}_{ms}$, etc.

Definition 122 (required words)**.** *We define $req^1(x)$ as the multiset of words required by the yet unparsed part of the ET corresponding to a given $x \in V_D \cup Q_M$, i.e.,*

$$req^1(x) = \begin{cases} super^1(x), & \text{if } x \in V_D, \\ super^1(v) \cup \bigcup_{i=1}^{|c|} sub(c_i), & \text{if } x \in Q_M, \end{cases}$$

where $(v,c) = suff^1(x)$ (for the case of $x \in Q_M$).

For instance, assuming the grammar from Fig. 7.3, $req^1(\mathrm{D}_2) = \emptyset_{ms}$, $req^1(\mathrm{NP}_{45}) = \{\mathrm{made}, \mathrm{decisions}\}$, $req^1(\mathrm{A}_{30}) = \{\mathrm{minister}\}$, etc. Moreover, assuming a simple FSA encoding of this grammar and using dotted rules to specify its states, $req^1(\bullet \mathrm{NP}_{45}\mathrm{VP}_{46} \rightarrow \mathrm{S}_{44}) = \{\mathrm{made}, \mathrm{decisions}\}_{ms}$, $req^1(\mathrm{NP}_{45} \bullet \mathrm{VP}_{46} \rightarrow \mathrm{S}_{44}) = \{\mathrm{made}, \mathrm{decisions}\}_{ms}$, $req^1(\mathrm{NP}_{45}\mathrm{VP}_{46} \bullet \rightarrow \mathrm{S}_{44}) = \emptyset_{ms}$, $req^1(\bullet \mathrm{A}_{41}\mathrm{N}^*_{42} \rightarrow \mathrm{N}_{40}) = \{\mathrm{good}\}_{ms}$, $req^1(\mathrm{A}_{41} \bullet \mathrm{N}^*_{42} \rightarrow \mathrm{N}_{40}) = \emptyset_{ms}$, $req^1(\mathrm{made}_{50} \bullet \rightarrow \mathrm{V}_{47}) = \{\mathrm{decisions}\}_{ms}$, $req^1(\mathrm{V}_{47} \bullet \mathrm{NP}_{48} \rightarrow \mathrm{NP}_{46}) = \{\mathrm{decisions}\}_{ms}$, etc.

Definition 123 (primary heuristic)**.** *For any given item we define a primary heuristic h_0 as in equation (7.18).*

$$h_0(\langle x, r\rangle) = \begin{cases} \infty, \ \textit{if } req^1(x) \not\subseteq \textit{rest}(r) \\ \textit{cost}(\textit{rest}(r) \setminus req^1(x)), \ \textit{otherwise.} \end{cases} \tag{7.18}$$

Definition 124 (top)**.** *Let $x \in V_D \cup Q_M$. Then, we define*

$$\textit{top}(x) = [x \in V_D \wedge \textit{root}_D(x)],$$

where [] is the Iverson bracket.

Definition 125 (heuristic)**.** *Let $\mathcal{W} = \langle \mathcal{H}, \omega_{\mathcal{H}} \rangle$ be a vanilla weighted parsing hypergraph and $\langle x, r\rangle \in V_{\mathcal{H}}$ be a chart item. Then, the estimation for the weight of the item $\langle x, r\rangle$'s best outside derivation, i.e. $\alpha(\langle x, r\rangle)$, is given by equation (7.19).*

$$h(\langle x, r\rangle) = \omega_D(\textit{tree}^1(x))[\neg \textit{top}(x)] + h_0(\langle x, r\rangle) \tag{7.19}$$

Given a hyperarc $e \in E_{\mathcal{H}}$, we define $h(e)$ as $h(\textit{head}(e))$.

For instance, $h(\langle \mathrm{NP}_{29}, \langle 0, 3\rangle\rangle_p) = 0 + 3$, $h(\langle \mathrm{NP}_{45}\mathrm{VP}_{46} \bullet \rightarrow S_{44}, \langle 0, 8\rangle\rangle_a) = 1 + 0$, $h(\langle \mathrm{S}_{44}, \langle 0, 8\rangle\rangle_p) = 0 + 0$, $h(\langle \mathrm{NP}_{13} \bullet \mathrm{VP}_{14} \rightarrow S_{12}, \langle 0, 3\rangle\rangle_a) = 1 + 2.5$, $h(\langle \bullet \mathrm{decisions}_{20} \rightarrow N_{19}, \langle 7, 7\rangle\rangle_a) = 1 + 4.5$, etc.

Fig. 7.16 and Fig. 7.17 show fragments of hyperpaths, decorated with β and h values, corresponding to the derivations shown in Fig. 7.14 (a MWE-driven derivation) and Fig. 7.15 (a compositional derivation), respectively. Both hyperpaths are part of a larger parsing hypergraph constructed over the given input sentence and based on the grammar shown in Fig. 7.3. However, the total $\beta + h$ weights assigned to the individual chart items in the MWE-oriented hyperpath are lower than the total weight assigned to any of the items present in the compositional hyperpath. This means that, in principle,

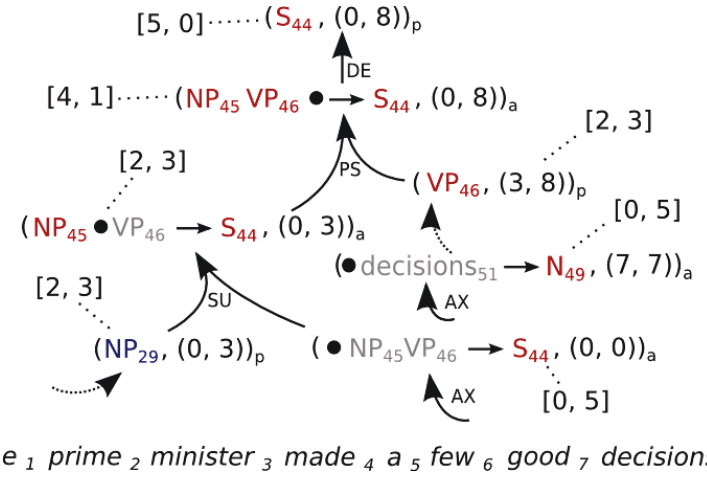


Figure 7.16: A fragment of a hyperpath corresponding to the MWE-oriented derivation shown in Fig. 7.14. The individual items are decorated with the pairs of the corresponding $[\beta, h]$ values, linked to them via dotted edges.

the A* parser should only explore the MWE-driven hyperpath before reaching the final item corresponding to the MWE-oriented derivation, while the items belonging to the compositional hyperpath should never be removed from the priority queue of the parser.

Fig. 7.14 and Fig. 7.15 illustrate a MWE-related ambiguity. Both derivations entail roughly equivalent syntactic structures (i.e., derived trees). In a real-sized TAG extending the grammar shown in Fig. 7.3, there could be many more ETs anchored in *the*, *prime*, *minister*, etc. For instance, the grammar would certainly contain ETs representing the past participle uses of the word *made*, as in *a house made of wood*. ETs which do not lead to a syntactically valid interpretation of the input sentences would be nevertheless considered by a symbolic TAG parser. In particular, the parser would consider the interpretations where *made* modifies (i) *minister*, (ii) *prime minister*, or (iii) *the prime minister*, before realizing that none of these interpretations leads to a valid parse. The MWE-driven A* parser, however, would benefit from the fact that *made* is a part of the MWE *made decisions*, potentially present in the sentence, and it would not remove the chart items corresponding to the three invalid, past participle-based interpretations before reaching the MWE-oriented derivation.

Proposition 34. *Let $\mathcal{W} = \langle \mathcal{H}, \omega_{\mathcal{H}} \rangle$ be a vanilla weighted parsing hypergraph.*

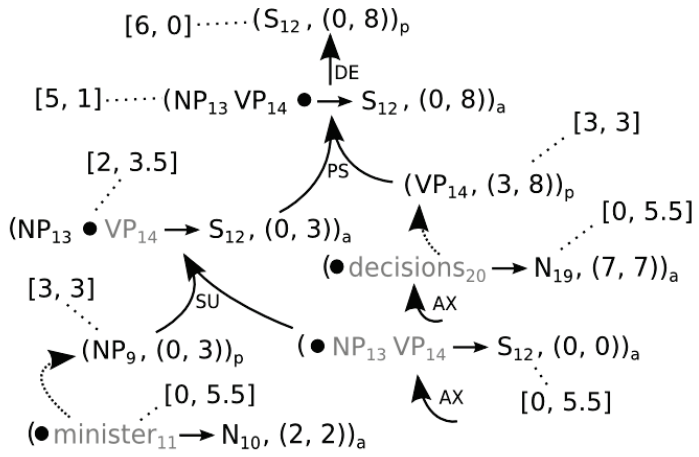


Figure 7.17: A fragment of a hyperpath, decorated with $[\beta, h]$ values, corresponding to the compositional derivation shown in Fig. 7.15. Note that the β value attached to the item $\langle \mathrm{NP}_{13} \bullet \mathrm{VP}_{14} \rightarrow \mathrm{S}_{12}, \langle 0, 3 \rangle \rangle_a$ is smaller than the sum of the β values assigned to the tail items of the corresponding, incoming hyperarc. This is because another, lighter inside derivation of this item exists in the hypergraph, even though it is not shown here.

Then, the heuristic defined by Eq. 7.19 is admissible.

The heuristic is based on the premise that when a given ET is being parsed (i.e. when an item $\langle x, r\rangle$ referring to one of its nodes or one of its traversals is considered), its weight $\omega_D(\mathit{tree}^1(x))$ will have to be eventually incorporated into the weight of the full TAG derivation. However, when a top-level passive item is considered (which entails that $\omega_D(\mathit{tree}^1(x))[\neg \mathit{top}(x)] = 0$), the corresponding tree weight has already been added to β. I.e., it is a part of the inside derivation, in accordance with Def. 43.

Furthermore, the remaining part of the ET being matched can contain some terminals. Since the weight of the ET is already accounted for, the scanning cost of such terminals should not be computed as a part of the cost of scanning the words in the remaining part of the input sentence. Thus, in the second line of Eq. 7.18, the required terminals are subtracted from the remaining terminals before the resulting scanning cost is computed. In the first line of this equation, on the other hand, the situation where the remaining part of the sentence does not contain all the required terminals is handled.

Proposition 35. *Let $\mathcal{W} = \langle \mathcal{H}, \omega_{\mathcal{H}}\rangle$ be a vanilla weighted parsing hypergraph. Then, the heuristic defined by Eq. 7.19 is* not *monotonic.*

The heuristic is not monotonic because the underlying vanilla weighted parsing system is *not* monotonic (cf. Def. 105). The problem lies in the weighted FA inference rule (see Tab. 7.3) which does not transfer the inside weight of the adjunction witness. As a result, the heuristic might underestimate the cost of scanning the items covered by the gap which, in fact, is already known.

Proposition 36. *Let $\langle G, \omega_G\rangle$ be a TAG, D be its simple DAG encoding and M be a D's simple FSA encoding. Let also s be a sentence, $\langle \mathcal{I}, \mathcal{D}\rangle$ be the corresponding vanilla parsing system, and $\langle x, r\rangle \in \mathcal{I}$ be an item. Then, the time complexity of computing the value $h(\langle x, r\rangle)$ is $\mathcal{O}(|\mathit{req}^1(x)|)$.*

The computation of the weight of $\mathit{tree}^1(x)$, as well as verifying $\mathit{top}(x)$, is constant time. The computation of $h_0(\langle x, r\rangle)$ is potentially more costly. The values of req^1 can be pre-computed for the individual $x \in V_D \cup Q_M$, which adds the $\mathcal{O}(|V_D| + |Q_M|)$ pre-processing overhead. Similarly, the values of *rest* can be pre-computed for the individual spans r in s, with at most

$\mathcal{O}(|s|^4)$ pre-processing overhead.[21] Both tasks are thus less expensive than TAG parsing itself from the time complexity point of view.

Proposition 37. *Let $\langle x, r\rangle$ be an item such that $req^1(x) \subseteq rest(r)$. Then,*

$$cost(rest(r) \setminus req^1(x)) = cost(rest(r)) - cost(req^1(x)). \tag{7.20}$$

From the above proposition stems that, if we pre-compute not only the values of req^1 and $rest$, but also the corresponding values of $cost \circ req^1$ and $cost \circ rest$,[22] respectively, then the value $cost(rest(r) \setminus req^1(x))$ can be computed in constant time provided that $req^1(x) \subseteq rest(r)$. However, the time complexity of verifying that $req^1(x) \subseteq rest(r)$ is $\mathcal{O}(|req^1(x)|)$.[23]

7.5 Extensions

We now propose three extensions of the parsing architecture described in Sec. 7.3 and the MWE-promoting strategy specified in Sec. 7.4, with the goal of showing their practicality in a real-world setting. Firstly, we introduce a variant of the MWE-promoting A* heuristic which not only has better theoretical properties, but also is easier to compute. Its monotonicity guarantees the correctness of the parser (cf. Def. 111) and makes it safe to combine it with other A* heuristics. Secondly, we show how to combine the A* parsing strategy with grammar compression techniques. Clearly, in a real-world setting, different parsing efficiency enhancements should ideally combine to provide an optimal solution. Finally, we show how to extend the parser so as to account for feature structures, which are commonly used in practical TAGs and allow to model the idiosyncratic properties of MWEs.

7.5.1 Adjunction-aware A* heuristic

The A* heuristic defined in Sec. 7.4.3 is admissible but it is not monotonic. Below we define a variant of this heuristic which is monotonic provided that the underlying TAG is non-negative.

[21] We assume some kind of data sharing between the individual values of *rest*. Otherwise, the memory complexity of such a solution could be prohibitive.

[22] The symbol $\circ$ represents the function composition in this case.

[23] Assuming constant lookup time in $rest(r)$.

Definition 126 (non-negative weighted TAG)**.** *Let $\langle G, \omega_G \rangle$ be a weighted TAG. Then, we say that $\langle G, \omega_G \rangle$ is non-negative iff $\forall_{t \in I_G \cup A_G}(\omega_G(t) \geq 0)$.*

First of all, let us note that the heuristic defined in Eq. 7.19 can be transformed into a version (based on Prop. 37) which, while slightly worse in terms of its estimations (when $req^1(x) \not\subseteq rest(r)$), is easier to compute:

$$h_{sim}(\langle x, r \rangle) = \omega_D(tree^1(x))[\neg top(x)] + cost(rest(r)) - cost(req^1(x)) \quad (7.21)$$

It can be shown that $h_{sim}(v)$ is smaller than $h(v)$ for any chart item v in a vanilla weighted parsing hypergraph and, thus, it is admissible.

Definition 127 (amortized weight)**.** *Let $x \in V_D \cup Q_M$. Then, we define its* amortized weight *as:*

$$A(x) = \omega_D(tree^1(x))[\neg top(x)] - cost(req^1(x)) \quad (7.22)$$

For a given $x \in V_D \cup Q_M$, $A(x)$ accounts for the weight of the ET t containing x, and for the fact that t may still contain some terminals which need to be consumed (w.r.t. to the position of x in t). Thus, $A(x)$ can be intuitively understood as the weight of the already parsed part of t (with the exception of the case where x is a root).

Proposition 38. *Let $x \in V_D : root_D(x)$. Then, $A(x) = 0$.*

Proposition 39. *Let $x \in V_D : \neg root_D(x)$. Then, the total weight that $tree^1_D(x)$ projects*[24] *over the words in $est_D(x)$ is equal or smaller than $A(x)$.*

The total weight that $tree^1_D(x)$ projects over the words in $est_D(x)$ equals $\omega_D(tree^1(x)) - w$, where w is the total weight that $tree^1_D(x)$ projects over the words in $req^1(x)$. The weight w, in turn, is equal or greater than $cost(req^1(x))$, hence Prop. 39.

Definition 128 (witness derivation chains)**.** *Recall that, in accordance with the propositions 18 and 19, each gapped item v (within the context of the vanilla parser) asserts an existence of at least one derivation chain satisfying certain constraints (in particular, the chain must cover the gap of v's span). We define the* witness derivation chains *of v, denoted $\vec{\Delta}_v$, as the set of such derivation chains.*

[24] See Fig. 7.15 for an example of projecting ET weights over the corresponding words.

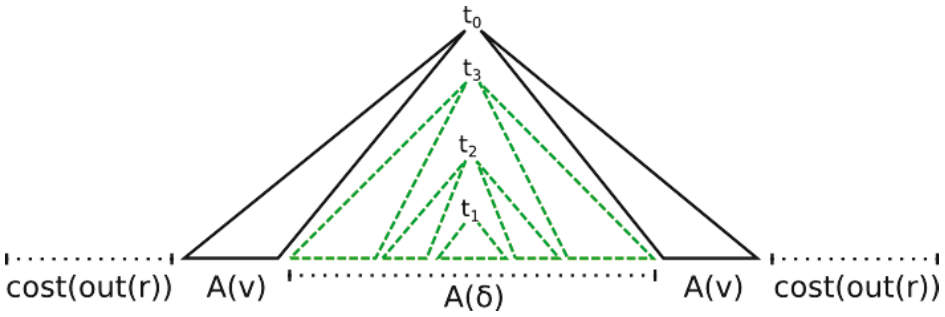


Figure 7.18: A configuration corresponding to a gapped passive item $v = \langle t_0, r\rangle$ with a particular witness derivation chain $\delta \in \vec{\Delta}_v$ (highlighted in green), where $|\delta| = 3$ and $\forall_{1\leq i\leq 3}(t_i = root_{drv}(\delta_i))$.

Definition 129 (amortized weight of a derivation chain)**.** *Let $\vec{\delta}$ be a GDAG derivation chain.*[25] *Then, we define its* amortized weight *as:*

$$A(\vec{\delta}) = \sum_{i=1}^{|\vec{\delta}|}\Big(\omega_G(\vec{\delta}_i) + A(root_{drv}(\vec{\delta}_i))\Big) \tag{7.23}$$

where $root_{drv}(\delta)$ is the root EST of the derivation tree δ.

Thus, $A(\vec{\delta})$ is the sum of the weights of the derivation trees participating in a given chain $\vec{\delta}$, adjusted with the amortized weights of their respective root ESTs.

Proposition 40. *Let $\langle x, r\rangle \in V_{\mathcal{H}}$ be a gapped chart item of a vanilla weighted parsing hypergraph $\mathcal{H}$ and $\vec{\delta} \in \vec{\Delta}_{\langle x,r\rangle}$. Then, the total weight that $\vec{\delta}$ projects over the words in $gap(r)$ is equal or smaller than $A(\vec{\delta})$.*

The weights projected by the individual ETs in a given chain $\vec{\delta} \in \vec{\Delta}_v$ over $gap(r)$ are accurately accounted for in Eq. 7.23. As to the remaining, EST trees in $\vec{\delta}$, by Prop. 39 the total weights they project over the words in $gap(r)$ are smaller than their amortized weights.

Fig. 7.18 shows a configuration of the parser which corresponds to (i) a gapped passive item $v = \langle t_0, r\rangle$, and (ii) one of its witness derivation chains

[25]We extend the notion of a derivation chain into a GDAG derivation chain in a similar way as we do it for derivations (cf. Def. 94). The functions applying to derivation chains can be easily extended to GDAG derivation chains as well.

$\delta \in \vec{\Delta}_v$. Item v refers to a particular EST, represented by a GDAG vertex t_0, and each of the derivation trees δ_i in δ contains in its root an EST represented by a GDAG vertex t_i (recall that each GDAG vertex unambiguously specifies the corresponding EST, cf. Prop. 8). The individual ESTs t_i are not necessarily ETs and, thus, can come with non-empty multisets of required words $req^1(t_i)$.

The idea behind the adjunction-aware heuristic is that the cost of such required words can be subtracted from the cost of the outer words, $cost(out(r))$, even before the value of the latter is known. This behavior is modeled in $A(\delta)$ (cf. Eq. 7.23), where the amortized weight of the individual ESTs – t_1, t_2, and t_3 – is accounted for. This amortized weight, in turn, subtracts $cost(req^1(t_i))$ from the weight of the corresponding tree (cf. Eq. 7.22).

The heuristic must provide a lower-bound estimate of the outside weight $\alpha(v)$ and, hence, it must choose the optimal derivation chain amongst all the chains in $\vec{\Delta}_v$. Put differently, when looking at the item v, the parser has no way of knowing which of the witness derivation chains in $\vec{\Delta}_v$ is a part of an optimal crossing derivation of v and, therefore, it must assume that it is the lowest-weight one.

Definition 130 (foot weight)**.** *Let $v \in V_{\mathcal{H}}$ be a chart item of a vanilla parsing hypergraph $\mathcal{H}$. Then, we define its* foot weight *as:*

$$\phi(v) = \begin{cases} 0, & \text{if } v \text{ is not gapped,} \\ \min\{A(\vec{\delta}) : \vec{\delta} \in \vec{\Delta}_v\}, & \text{otherwise.} \end{cases} \tag{7.24}$$

Definition 131 (adjunction-aware heuristic)**.** *Let $\mathcal{W} = \langle \mathcal{H}, \omega_{\mathcal{H}} \rangle$ be a vanilla weighted parsing hypergraph and $\langle x, r \rangle \in V_{\mathcal{H}}$ be a chart item. Then, we define the* adjunction-aware heuristic *as:*

$$h_{adj}(\langle x, r \rangle) = A(x) + cost(out(r)) + \phi(\langle x, r \rangle). \tag{7.25}$$

Given a hyperarc $e \in E_{\mathcal{H}}$, we define $h_{adj}(e)$ as the estimation of $h_{adj}(head(e))$ based on the incoming hyperarc e.[26]

The main differences, in comparison with the heuristic specified in Def. 125, are that:

- the cost of $out(r)$ rather than $rest(r)$ is taken into account, and

[26] As will be later explained, the values of ϕ are computed via inference rules, which makes the value of $h_{adj}(e)$ dependent on e.

- the cost of the gap is accounted for in $\phi(\langle x, r\rangle)$, which potentially provides a better estimation of the cost of scanning the terminals in $in(r)$ than $cost(in(r))$ does.

Proposition 41. *Let $\langle x, r\rangle \in V_{\mathcal{H}}$ be a gapped chart item of a vanilla weighted parsing hypergraph $\mathcal{H}$. Then,*

$$cost(in(r)) \leq \phi(\langle x, r\rangle). \tag{7.26}$$

Let us look at a particular $\vec{\delta} \in \vec{\Delta}_{\langle x,r\rangle}$. The individual derivation trees $\vec{\delta}_i$ in $\vec{\delta}$ together project some particular weights over all the words in $gap(r)$. Those weights are at least as high as the global minimum projection weights represented by $minw$ and, thus, the total weight $\vec{\delta}$ projects over $gap(r)$ is greater or equal to $cost(in(r))$. At the same time, this total weight is equal or smaller than $A(\vec{\delta})$ (cf. Prop. 40). This reasoning applies to any $\vec{\delta} \in \vec{\Delta}_{\langle x,r\rangle}$, hence the above proposition.

Proposition 42. *Let $\mathcal{W} = \langle \mathcal{H}, \omega_{\mathcal{H}}\rangle$ be a hypergraph corresponding to a vanilla weighted parsing system. Then, the heuristic defined by Eq. 7.25 is admissible.*

The admissibility of the heuristic h_{adj} stems from the observation that, given a gapped item $v = \langle x, r\rangle$ and a derivation tree δ corresponding to a final hyperpath crossing v,[27] one of the derivation chains $\vec{\delta} \in \vec{\Delta}_v$ must also be a part of δ. Let us first assume that $\vec{\delta}$ is a least-weight derivation chain, i.e., that it minimizes $\phi(v)$. Then, the globally minimal weights of all the words required both by x and $\vec{\delta}$ are subtracted from $cost(out(r))$, so that they are not accounted for twice. As to the words remaining in $out(r)$, the heuristic assumes that they will be scanned with the globally smallest costs possible, which entails admissibility of the estimation. Otherwise, if $\vec{\delta}$ is not a least-weight derivation, then including the (optimal) foot weight $\phi(v)$ in h_{adj}'s calculation is all the more an underestimation.

Proposition 43. *Let $\mathcal{W} = \langle \mathcal{H}, \omega_{\mathcal{H}}\rangle$ be a hypergraph corresponding to a vanilla weighted parsing system. Then, the heuristic defined by Eq. 7.25 is monotonic, provided that the underlying grammar is non-negative.*

The values of ϕ are computed via inference rules specified in Tab. 7.4. In comparison with Tab. 7.3, a pair of numbers $(x, y) \in \mathbb{R} \times (\mathbb{R} \cup \{-\})$ is

[27] I.e., a final hyperpath which is a crossing derivation of v

attached to each item, specified before the colon in the inference rules.[28] The meaning of x is as before: it represents the inside weight. The value y, on the other hand, represents $\phi(v)$, where v is the corresponding item (given after the colon). The value $y = -$ is used within the context of the ungapped items, whose foot weight is known to be equal to 0.

Algorithm 4 can be used to implement the rules specified in Tab. 7.4. The priority of an item v is based on the estimation $\hat{\beta}(v)$ of its inside weight $\beta(v)$ (as before) and the estimation of its h_{adj} value, which relies on the estimation $\hat{\phi}(v)$ of $\phi(v)$ computed via the rules. When an item is inferred which has been already deduced before, (i) the minimum of the computed inside weight estimations, and (ii) the minimum of the computed ϕ estimations are preserved.

Proposition 44. *Let $\langle \mathcal{I}, \mathcal{D} \rangle$ be a weighted parsing system following the specification given in Tab. 7.4 and $\mathcal{A}$ be the Dijkstra-style parsing algorithm specified in Alg. 4, modified in order to trace the estimations of ϕ. Then, provided that the grammar is non-negative, $\mathcal{A}$ exhibits the following properties:*

- *When an item v is removed from the priority queue Q, its estimated $\hat{\beta}(v)$ equals $\beta(v)$ and its estimated $\hat{\phi}(v)$ equals $\phi(v)$.*
- *Items are removed from Q in an ascending order of their $\beta + h_{adj}$ weights.*

Proposition 45. *Let $\langle G, \omega_G \rangle$ be a TAG, D be its simple DAG encoding, and M be a D's simple FSA encoding. Let also s be a sentence, $\langle \mathcal{I}, \mathcal{D} \rangle$ be the corresponding adjunction-aware weighted parsing system, and $\langle x, r \rangle \in \mathcal{I}$. Then, the time complexity of computing the value $h_{adj}(\langle x, r \rangle)$ is $\mathcal{O}(1)$.*

Recall that to compute $h_{adj}(\langle x, r \rangle)$, the values $A(x)$, $cost(out(r))$, and $\phi(\langle x, r \rangle)$ have to be calculated. The values of A are grammar-dependent only and can be pre-computed for the individual $x \in V_D \cup Q_M$, with the $\mathcal{O}(|V_D \cup Q_M|)$ time complexity overhead. The values of $cost \circ out$ can be pre-computed with the $\mathcal{O}(|s|)$ time complexity overhead. The linear behavior can be obtained in this case by calculating, separately, the costs of (i) the span $\langle 0, k \rangle$ for each $k \in 0 \ldots |s|$, and (ii) the span $\langle k, |s| \rangle$ for each $k \in 0 \ldots |s|$, and then combining them to obtain $cost \circ out$ for any given span in s. Finally, the values of ϕ are computed via inference rules, of which the most complex

[28] A similar solution, used for trace the optimal weights of the prediction-related derivations, can be found in (Nederhof, 2003, p. 137).

AX:	$\frac{}{(0,-):\langle q_0,\langle i,i\rangle\rangle_a}$	$i\in pos(s)\setminus\{n\}$ $q_0\in S_M$
SC:	$\frac{(x,y):\langle q,\langle i,j,k,l\rangle\rangle_a}{(x,y):\langle \delta(q,v),\langle i,j,k,l+1\rangle\rangle_a}$	$v\in leaves(s_{l+1})$ $\delta(q,v)$ defined
DE:	$\frac{(x,y):\langle q,\langle i,j,k,l\rangle\rangle_a}{(x+c,y):\langle v,\langle i,j,k,l\rangle\rangle_p}$	$v\in heads(q)$ $c=[root(v)]\omega(tree(v))$
PS:	$\frac{(x_1,y_1):\langle q,\langle i,j,k,l\rangle\rangle_a \quad (x_2,y_2):\langle v,\langle l,j',k',l'\rangle\rangle_p}{(x_1+x_2,y_1\cup y_2):\langle \delta(q,v),\langle i,j\cup j',k\cup k',l'\rangle\rangle_a}$	$\delta(q,v)$ defined
SU:	$\frac{(x_1,y_1):\langle q,\langle i,j,k,l\rangle\rangle_a \quad (x_2,-):\langle v,\langle l,l'\rangle\rangle_p}{(x_1+x_2,y_1):\langle \delta(q,v'),\langle i,j,k,l'\rangle\rangle_a}$	$v'\in leaves(\ell(v))\wedge\neg foot(v')$ $\delta(q,v')$ defined $root(v)$
FA:	$\frac{(x_1,-):\langle q,\langle i,l\rangle\rangle_a \quad (x_2,y_2):\langle v,\langle l,j',k',l'\rangle\rangle_p}{(x_1,x_2+y_2+A(v)):\langle \delta(q,v'),\langle i,l,l',l'\rangle\rangle_a}$	$v'\in leaves(\ell(v))\wedge foot(v')$ $\delta(q,v')$ defined $root(v)\Longrightarrow (j',k')=(-,-)$
RA:	$\frac{(x_1,y_1):\langle w,\langle i,j,k,l\rangle\rangle_p \quad (x_2,y_2):\langle v,\langle j,j',k',k\rangle\rangle_p}{(x_1+x_2,y_2):\langle v,\langle i,j',k',l\rangle\rangle_p}$	$root(w)\wedge(j,k)\neq(-,-)$ $\ell(w)=\ell(v)$ $root(v)\Longrightarrow (j',k')=(-,-)$

Table 7.4: A variant of the weighted vanilla parser (cf. Tab. 7.3) adapted to trace the estimations of foot weights.

computation occurs in the FA rule which, nevertheless, can be performed in constant time (it is, again, based on the pre-computed values of A and relies on a constant lookup time).

7.5.2 Grammar compression

As shown before in Fig. 7.2, different DAG encodings of a given TAG are possible. In particular, Fig. 7.2 (b) presents a variant of a DAG encoding where each grammar EST is represented by exactly one DAG vertex.

Definition 132 (subtree sharing)**.** *Let G be a TAG and D be its DAG encoding. Then, we say that D is a* subtree sharing *DAG encoding of G iff each grammar EST is represented by exactly one vertex in V_D.*

Proposition 46. *Let D be a simple GDAG. Let also $\langle \mathcal{I}, \mathcal{D}\rangle$ be a vanilla parsing system based on D and $\mathcal{H}$ be the corresponding hypergraph. Then,*

for any two passive items $\langle v, r\rangle, \langle v', r'\rangle \in \mathcal{I}$ *such that* $r = r'$ *and* $est_D(v)$ *is identical to* $est_D(v')$*, it holds that* $\langle v, r\rangle \in V_{\mathcal{H}} \iff \langle v', r'\rangle \in V_{\mathcal{H}}$.

The above proposition suggests that, from the bottom-up parsing point of view, there is no point in distinguishing two identical grammar ESTs, even when they belong to two different ETs. This motivates the use of the subtree sharing encoding method.

Even though quite simple, the subtree sharing encoding has an important influence on the complexity of the applications of the individual parsing inference rules (for instance, cf. the rules SC and SU in Tab. 7.2).

Proposition 47. *Let* D *be a subtree sharing GDAG and* $x \in \Sigma_D \cup N_D$*. Then,* $|leaves_D(x)| \leq 1$ *(cf. Def. 83).*

Note that a leaf is a also a subtree, it must be thus shared among the trees containing it.

On the other hand, the subtree sharing encoding renders more difficult the computation of the values of the heuristic h. Recall that, in the definition of h, we rely on the functions $tree^1_D$, req^1_D, etc., and, consequently, on a simple DAG encoding. Otherwise, the calculation of $h(\langle x, r\rangle)$ for a given item $\langle x, r\rangle$ must consider all the different ETs to which $x \in V_D \cup Q_M$ may correspond. Thus, while a symbolic TAG parser can become significantly faster when the subtree encoding is used (Waszczuk et al., 2016a), this encoding may make the computations of the values of the A* heuristic significantly slower at the same time.

Given a particular DAG encoding, different FSA encoding methods can be used to store its various traversals (cf. Def. 63). In the simplest case, each traversal is represented by a distinct, single path quasi-FSA (see Fig. 7.4 (a)). Another possibility is to use a prefix tree in which common prefixes of the individual traversals are shared (see Fig. 7.4 (b)).

Definition 133 (prefix tree encoding)**.** *Let* D *be a GDAG. Then, we say that* M *is a* prefix tree *FSA encoding of* D *iff* M *encodes the individual traversals of* D *in the form of a prefix tree.*

Definition 134 (traversal prefix)**.** *Let* M *be an FSA family – either simple or a prefix tree – and* $q \in Q_M$*. We define* $pref_M(q) \in V_D^*$ *as the traversal prefix of* q *– a sequence of DAG vertices starting from some* $q_0 \in S_M$ *and terminating in* q*.*

An even more compressed representation could be based on a minimal deterministic finite-state automaton.[29] However, we did not identify a significant speed-up when using this compression technique in comparison with a prefix tree (Waszczuk et al., 2016a), thus we will focus on the prefix tree representation in this section.

Proposition 48. *Let D be a GDAG and M be its simple FSA encoding. Let also $\langle \mathcal{I}, \mathcal{D} \rangle$ be a vanilla parsing system based on D and $\mathcal{H}$ be the corresponding hypergraph. Then, for any two active items $\langle q, r \rangle, \langle q', r' \rangle \in \mathcal{I}$ such that $r = r'$ and $pref_M(q) = pref_M(q')$, it holds that $\langle q, r \rangle \in V_{\mathcal{H}} \iff \langle q', r' \rangle \in V_{\mathcal{H}}$.*

The above proposition shows that the prefix tree encoding allows to group active chart items into classes equivalent from the left-to-right, bottom-up parsing point of view, in a similar way as subtree sharing groups together passive items corresponding to identical grammar ESTs.

Both the simplified heuristic $h_{sim}(\langle x, r \rangle) = A(x) - cost(rest(r))$ (see Eq. 7.21 and Def. 127) and the adjunction-aware heuristic h_{adj} are more robust than h with respect to both the subtree sharing and the prefix tree optimizations. Even though the amortized weight A has to be re-defined in order to account for such optimizations, its values can be pre-computed before parsing takes place.

Definition 135 (extended amortized weight)**.** *Let D be a GDAG, M be its FSA encoding, and $x \in V_D \cup Q_M$. Then, we define the* extended amortized weight *of x as:*

$$A_{ext}(x) = \begin{cases} 0, & \text{if } top(x) \\ A_D(x), & \text{if } x \in V_D \\ A_M(x), & \text{otherwise,} \end{cases} \tag{7.27}$$

where:

$$A_D(x) = \min\{\omega_D(t) - cost(sub_D(t) \setminus sub_D(x)) \colon t \in tree_D(x)\} \tag{7.28}$$

and:

$$A_M(x) = \min\{\omega_D(t) - cost((sub_D(t) \setminus sub_D(v)) \cup \bigcup_{i=1}^{|c|} sub_D(c_i)) \colon \langle v, c \rangle \in suff_M(x), t \in tree_D(v)\} \tag{7.29}$$

[29] One way to obtain suffix sharing is to replace the heads of the top-level traversals by the corresponding non-terminal symbols, as proposed in (Waszczuk et al., 2016a).

Proposition 49. *Let D be a simple GDAG, M be its simple FSA encoding, and $x \in V_D \cup Q_M$. Then, it holds that $A(x) = A_{ext}(x)$.*

A_{ext} can be alternatively expressed in a recursive manner and its individual values calculated using dynamic programming techniques, hence the following proposition:

Proposition 50. *Let D be a GDAG and M be its FSA encoding. Then, the values of A_D, A_M, and A_{ext} can be calculated in time $\mathcal{O}(|D| + |\delta_M|)$, where $|D| = \sum_{\langle p,c\rangle \in E_D} |c|$.*

Proposition 51. *Let G be a TAG, D be its subtree sharing DAG encoding, and M be a D's prefix tree FSA encoding. Let also s be a sentence, $\langle \mathcal{I}, \mathcal{D}\rangle$ be the corresponding adjunction-aware weighted parsing system, and $\langle x, r\rangle \in \mathcal{I}$. Then, the time complexity of computing the value $h_{adj}(\langle x, r\rangle)$ (based on the pre-computed values of A_{ext}) is $\mathcal{O}(1)$.*

Related work

FSA-based grammar encoding considerably speeds up CFG parsing (Klein and Manning, 2001b) but it is not straightforwardly applicable to TAGs (which consist of trees rather than flat rules). It is, however, enabled by the flattening transformation proposed in this paper. Previous proposals of applying FSA-based compression to TAGs are manifold. Kallmeyer (2010) and Prolo (2002) describe LR parsers for TAGs, in which predictions are pre-compiled off-line into an FSA. Each state of this FSA is a set of dotted production rules closed under prediction.

Another automata-based solution for LTAGs and related lexicalized formalisms has been proposed by Evans and Weir (1997); Carroll et al. (1998). The traversal of an ET, starting from its anchor (lexical unit), is represented there as an automaton. Sets of trees attached to common anchors are then converted to automata, merged and minimized using standard techniques. As a result, structure sharing occurs only within tree families, while in our solution all ETs are represented with a single automaton which provides sharing between rules assigned to different lexical units. Another potential advantage of our solution lies in the subtree-sharing it enables, which allows different rules – even when represented by completely different paths in the automaton – to share common middle elements if these middle elements represent common subtrees. Finally, our method can be used for TAGs in general, not only for lexicalized TAGs.

Villemonte de La Clergerie (2010) proposes a method of grammar compression directly at the stage of its definition. A linguist uses a formal language including factoring operators (e.g., disjunctions over tree fragments, Kleene-star-alike repetitions, optional or shuffled fragments, etc.) and the resulting grammar is then converted into a Logic Push-Down Automaton for parsing. The price to pay for this highly compact resource is its high potential overgeneration. Moreover, grammar description and parsing are not separated, hence large non-factorized TAGs can be hardly coped with. Our solution abstracts away from how the TAG is represented, compression is automatic and the FSA representation is strongly equivalent to the original TAG.

Similar techniques can be also used in dependency parsing, as shown in (Nasr and Rambow, 2010), where a grammar takes the form of a family of FSAs, and each FSA is used to encode the possible dependency tree fragments (specifying subcategorization and modification requirements) corresponding to its lexical anchor (each FSA is required to contain a lexical anchor). The parsing algorithm is then an extension of the chart parsing algorithm for CFGs, and chart items refer to grammar FSAs rather than to the unfolded dependency fragments.

7.5.3 Parsing with feature structures

Practical TAG grammars are often decorated with feature structures (FSs), which specify unification-like computations within the scope of the individual ETs. Such computations turn out useful within the context of MWEs as well. For instance, they allow to enforce additional agreement constraints, as shown before in Fig. 5.4.

One possible way to add support for FSs would be to handle them in the post-processing phase of syntactic parsing, a solution implemented in TuLiPa, a parsing environment which employs Range Concatenation Grammar as a pivot formalism and which can be used to parse with several mildly context-sensitive formalisms, in particular with TAGs (Parmentier et al., 2008). On the other end of the spectrum would be a solution where FSs are not only handled during parsing, but also common parts of different FSs are shared between the individual trees of the grammar. Such a solution would extend the mechanism of sharing common parts of ETs implemented in ParTAGe.

The solution implemented in ParTAGe is placed between the two solutions mentioned above. Namely, FSs are processed during parsing, but no sharing

of common FS parts is performed. Structurally, FSs can be seen as graphs (Francez and Wintner, 2012), and thus implementing the sharing functionality for them would be more difficult than for elementary trees. Moreover, this decision allows to abstract away from the particularities and low-level details of FSs and to focus on what they represent from the parsing point of view – that is, unification-like computations over derivation trees, computations which can possibly fail.

As already mentioned, TuLiPa handles FSs in post-processing, but there are also advantages of handling them during parsing. Notably, if we consider probabilistic A* parsing, then handling FSs after parsing may lead to the rejection of the most-probable parse(s) found by the A* algorithm due to potential unification failures over the corresponding FSs. This undesirable situation is avoided when unification of FSs is resolved on the fly.

Definition 136 (rose trees)**.** *Let X be a set. Then, we define $R(X)$ as the set of* rose trees *with nodes labeled by the values from the set X. Formally, $R(X)$ can be defined as the smallest set satisfying the following properties:*

$$\forall_{x \in X} \langle x, \epsilon \rangle \in R(X) \tag{7.30}$$

$$\vec{t} \in R(X)^* \implies \forall_{x \in X} \langle x, \vec{t} \rangle \in R(X) \tag{7.31}$$

Definition 137 (set with bottom)**.** *Let X be a set. Then, we define $X^\perp$ as $X \cup \{\perp\}$.*

Fig. 7.19 provides an example of two rose trees. The tree on the right belongs to $R(F^\perp)$, where F is a set of feature structures, and takes the form $\langle \perp, (\langle x, \epsilon \rangle, \langle e, (l, r) \rangle) \rangle$, where $l = \langle \perp, (\langle \perp, \epsilon \rangle) \rangle$, $r = \langle \perp, (\langle x, \epsilon \rangle, l) \rangle$, $x \in F$ is the FS corresponding to $\{\langle \text{Pers}, 3 \rangle, \langle \text{Num}, \text{sg} \rangle\}$, and $e \in F$ is an empty FS.

Definition 138 (unification system)**.** *We define a* unification system *as a tuple $\mathcal{U} = \langle F_\mathcal{U}, \leq_\mathcal{U}, \uplus_\mathcal{U}, T_\mathcal{U}, \theta_\mathcal{U} \rangle$ such that:*

- *$F_\mathcal{U}$ is a set of FS-like*[30] *values such that $\perp \notin F_\mathcal{U}$,*
- *$\leq_\mathcal{U}$ is a total order on $F_\mathcal{U}$,*
- *$\uplus_\mathcal{U} \colon F_\mathcal{U} \to F_\mathcal{U} \to F_\mathcal{U}^\perp$ is a function which unifies two values and returns $\perp$ in case of unification failure,*

[30] By FS-like values we mean the values handled by the unification system, typically feature structure values.

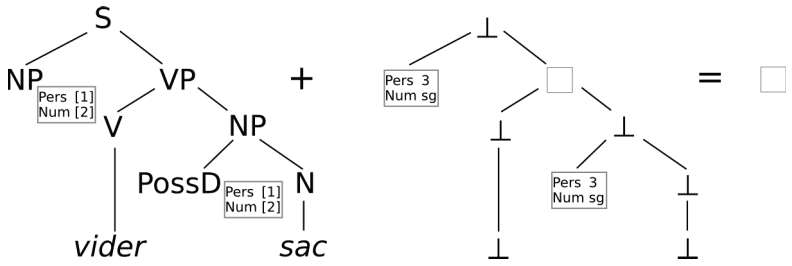


Figure 7.19: A graphical representation of a unification computation given an ET decorated with FSs and unification variables (on the left) and a tree of FSs originating from adjunctions and substitutions (on the right). Among the non-leaf nodes, an adjunction is performed on the VP node exclusively. The unification of the corresponding person and number features succeeds and the result is an empty FS.

- *$T_\mathcal{U}$ is the set of objects (in practice, grammar ETs) for which unification computations are defined,*
- *$\theta_\mathcal{U} \colon T_\mathcal{U} \to R(F_\mathcal{U}^\perp) \to F_\mathcal{U}^\perp$ is a family of unification computations defined for the individual objects in $T_\mathcal{U}$.*

A particular unification computation $\theta_\mathcal{U}(t)$, for a given $t \in T_\mathcal{U}$, takes a tree of FS-like values $\nu \in R(F_\mathcal{U}^\perp)$ and returns a single FS-like value $v \in F_\mathcal{U}$, or $\perp$ if the unification computation fails. Within the context of a real FS unification, the role of $\theta_\mathcal{U}(t)$ is to verify that the FS values attached to the individual nodes in t unify with the corresponding FS values in ν, the latter originating from the adjunction and substitution operations performed on the individual nodes in t. In ν, the value $\perp$ is used to indicate the non-leaf nodes which are not modified via adjunction.[31] Fig. 7.19 shows an example of a unification computation based on FS values.

A unification system abstracts over the particularities of feature structures and, thus, allows to express different types of unification-like computations. A simple example is a system where $F_\mathcal{U} = \{\top\}$[32] and $\theta_\mathcal{U}(t)$ verifies that all

[31] An alternative solution would be to assume that a neutral element $e \in F_\mathcal{U}$ exists such that $e \uplus v = v \uplus e = v$ for each $v \in F_\mathcal{U}$, and assign it to all non-modified internal nodes. Then the type of ω would be $R(F_\mathcal{U}) \to F_\mathcal{U}^\perp$. An empty FS provides such a neutral element in case of FSs. However, such an alternative solution would make it more difficult to account for adjunction constraints, which can be handled by unification computations.

[32] $\top$ stands for top.

the adjunction constraints assigned to the individual nodes in t are satisfied. Examples of adjunction constraints are *null adjunction* and *obligatory adjunction*. The former forbids adjunction at a given node, while the latter makes adjunction at the node required. In order to account for such constraints, $\theta_{\mathcal{U}}(t)(\nu)$ can look at each node in t and verify that (i) if it is marked with null adjunction, then the corresponding value in ν is $\bot$, or (ii) if it is marked with obligatory adjunction, then the corresponding value in ν is $\top$. Otherwise, $\theta_{\mathcal{U}}(t)(\nu)$ fails.

ParTAGe requires a total order to be defined over chart items. This allows to speed up the search of the corresponding chart items when the individual inference rules of the parser are considered. This is why, in order to satisfy this requirement, we assume a total order $\leq_{\mathcal{U}}$ over the set $F_{\mathcal{U}}$ of FS-like values.

Tab. 7.5 shows a variant of the vanilla parser adapted to handle FS-like computations. The main modifications introduced in the FS-aware version of the parser are:

- Three types of chart items are distinguished – active, passive and (new) top items. Top items represent fully recognized ETs over which the unification computation was performed (and did not fail). Passive items can also represent fully recognized ETs, but their root nodes can still undergo adjunction and their unification computation is not yet performed.

- A new inference rule (FI, standing for *finalize*), related to the distinction between passive and top chart items, is introduced. It models the transition from a passive to the corresponding top item and applies the unification computation θ assigned to the corresponding ET. If the computation fails, the corresponding top chart item is not inferred.

- A *trace* is added to the individual chart items. It keeps track of the FS-like values computed for the ETs inserted (substituted, adjoined) in place of the already processed non-terminals of the ET represented by a given chart item. This enables to perform the unification computation once the full ET is matched.

In the definitions below and in Tab. 7.5, we assume a fixed GDAG D, FSA family M, input sentence s, and unification system $\mathcal{U}$. The individual computations are defined over the roots in D, i.e., $T_{\mathcal{U}} = \{v \in V_D : root_D(v)\}$. Recall that each root in D represents a grammar ET (cf. Prop. 8).

AX:	$\dfrac{}{\langle q_0,\epsilon,\langle i,i\rangle\rangle_a}$	$i\in pos(s)\backslash\{n\}$ $q_0\in S_M$
SC:	$\dfrac{\langle q,\vec{t},\langle i,j,k,l\rangle\rangle_a}{\langle\delta(q,v),\vec{t}\cdot(\langle\bot,\epsilon\rangle),\langle i,j,k,l{+}1\rangle\rangle_a}$	$v\in leaves(s_{l+1})$ $\delta(q,v)$ defined
DE:	$\dfrac{\langle q,\vec{t},\langle i,j,k,l\rangle\rangle_a}{\langle v,\langle\bot,\vec{t}\rangle,\langle i,j,k,l\rangle\rangle_p}$	$v\in heads(q)$
FI:	$\dfrac{\langle r,t,r\rangle_p}{\langle\ell_D(r),x,r\rangle_t}$	$root_D(r)$ $x{=}\theta_{\mathcal{U}}(tree^1_D(r))(t)$ $x\neq\bot$
PS:	$\dfrac{\langle q,\vec{t},\langle i,j,k,l\rangle\rangle_a \quad \langle v,t,\langle l,j',k',l'\rangle\rangle_p}{(\delta(q,v),\vec{t}\cdot(t),\langle i,j{\cup}j',k{\cup}k',l'\rangle\rangle_a}$	$\delta(q,v)$ defined
SU:	$\dfrac{\langle q,\vec{t},\langle i,j,k,l\rangle\rangle_a \quad \langle v,x,\langle l,l'\rangle\rangle_t}{\langle\delta(q,v'),\vec{t}\cdot(\langle x,\epsilon\rangle),\langle i,j,k,l'\rangle\rangle_a}$	$v'\in leaves(v)\wedge\neg foot(v')$ $\delta(q,v')$ defined
FA:	$\dfrac{\langle q,\vec{t},\langle i,l\rangle\rangle_a \quad \langle v,t,\langle l,j',k',l'\rangle\rangle_p}{\langle\delta(q,v'),\vec{t}\cdot\langle\bot,\epsilon\rangle,\langle i,l,l',l'\rangle\rangle_a}$	$v'\in leaves(\ell(v))\wedge foot(v')$ $\delta(q,v')$ defined $root(v)\Longrightarrow(j',k')=(-,-)$
RA:	$\dfrac{\langle w,x,\langle i,j,k,l\rangle\rangle_t \quad \langle v,\langle y,\vec{t}\rangle,\langle j,j',k',k\rangle\rangle_p}{\langle v,\langle z,\vec{t}\rangle,\langle i,j',k',l\rangle\rangle_p}$	$(j,k)\neq(-,-)\wedge w{=}\ell(v)$ $root(v)\Longrightarrow(j',k')=(-,-)$ $z{=}$ if $y\neq\bot$ then $x{\uplus}y$ else x $z\neq\bot$

Table 7.5: A variant of the vanilla parser adapted to handle FS-like values.

Definition 139 (top item)**.** *Let $v \in N_D$ be a non-terminal, $x \in F_{\mathcal{U}}$ be a FS-like value, and r be a span in s. Then, $\langle v, x, r \rangle_t$ is a* top item*.*

Definition 140 (passive item)**.** *Let $v \in V_D$ be a DAG vertex, $t \in R(F_{\mathcal{U}})$ be a tree of FS-like values, and r be a span in s. Then, $\langle v, t, r \rangle_p$ is a* passive item*.*

Definition 141 (active item)**.** *Let $q \in Q_M$ be an automaton state, $\vec{t} \in R(F_{\mathcal{U}})^*$ be a sequence of trees of FS-like values, and r be a span in s. Then, $\langle q, \vec{t}, r \rangle_a$ is an* active item*.*

Support for FSs is independent from the grammar compression techniques described in Sec. 7.5.2 and can be safely composed with both subtree sharing and prefix tree FSA encoding. It is also orthogonal to the idea of A* parsing and could[33] be used in combination with the heuristics described in Sec. 7.4.3 and Sec. 7.5.1.

7.6 Conclusions

Tab. 7.6 shows a comparison of the individual parsing architectures we have seen in this section based on their: (i) parsing time and space complexity, (ii) heuristic-related and statistical-parsing-related properties, (iii) ability to compose with grammar compression techniques, and (iv) support for FSs.

Let us recall that the heuristic-related properties are crucial for us since our goal is to use an A* heuristic to implement the MWE-promoting strategy (cf. Sec. 7.1) in order to improve accuracy and efficiency. Such a heuristic should ideally be admissible (cf. Eq. 113) and monotonic (cf. Eq. 7.10), since both properties entail correctness (cf. Def. 111). Besides, calculating its values should not cause an important computational overhead. It should also finely combine with grammar compression techniques which are often used to increase efficiency of symbolic parsing in general. Finally, it should be possible to extend an A*, heuristic-based parser to account for feature structures, since the latter play an important role in practical TAGs and allow to handle the idiosyncratic properties of MWEs.

The basic parser, specified in Tab. 7.1, is a symbolic parser which always generates the entire hypergraph, hence its space complexity is $\mathcal{O}(n^6)$, where n is the length of the input sentence. It combines with all three grammar

[33]We consider the support for FSs in ParTAGe as experimental and we did not verify empirically that it can be integrated with A* parsing.

		basic	vanilla	Dijkstra style	A* with h	A* with h_{adj}	FS-aware
complexity	time	$\mathcal{O}(n^6)$	$\mathcal{O}(n^6)$	$\mathcal{O}(n^6)$	$\mathcal{O}(n^6)$	$\mathcal{O}(n^6)$	$\mathcal{O}(n^6)$?
	space	$\mathcal{O}(n^6)$	$\mathcal{O}(n^6)$	$\mathcal{O}(n^4)$	$\mathcal{O}(n^4)$	$\mathcal{O}(n^4)$	$\mathcal{O}(n^6)$?
heuristic (statistical parsing)	complexity	n/a	n/a	n/a	$\mathcal{O}(req^1(x))$	$\mathcal{O}(1)$	n/a
	admissible	n/a	n/a	n/a	+	+	n/a
	monotonic	n/a	n/a	-	-	+	n/a
	correctness	n/a	n/a	?	?	+	n/a
compression	subtree sh.	+	+	+	~	+	+
	prefix tree	+	+	+	~	+	+
	min. DFA	+	+	+	~	+	-
	FS-support	-	-	-	-	-	+

Table 7.6: A comparison of the different parsing architectures.

compression techniques we mentioned in Sec. 7.5.2: subtree sharing, the prefix-tree representation of tree traversals, and compressing the traversals in the form of a minimal deterministic finite-state automaton (minimal DFA).

The vanilla parser (cf. Tab. 7.2), implemented in ParTAGe (cf. Alg. 4), is equivalent to the basic parser in terms of the properties we look at in Tab. 7.6. However, due to its modified FA inference rule, which performs a function similar to prediction in CFG parsing, in practice it should be faster and leave a smaller memory footprint.

When the Dijkstra-style algorithm (cf. Alg. 4) is used with the weighted parsing system (cf. Tab. 7.3) to find the least-weight derivation for a given sentence, its memory footprint drops to $\mathcal{O}(n^4)$. This is because only one, optimal incoming hyperarc per hypernode (chart item) has to be stored. However, even if the underlying grammar is non-negative, the weighted parsing system is not monotonic and, therefore, correctness of the parsing algorithm is not guaranteed (hence the question mark in Tab. 7.6).

The A* parsing algorithm (cf. Def. 115) based on the weighted parsing system (cf. Tab. 7.3) and the heuristic h (cf. Def. 125) is an extension of the Dijkstra-style parser in which h is used to guide the parser to reach the optimal solution more quickly. To recall, the heuristic h estimates the cost of parsing the remaining (w.r.t. the given chart item) part of the input sentence based on the potential MWE occurrences therein. This heuristic is admissible (cf. Prop. 34) but not monotonic (cf. Prop. 35), thus correctness of the parsing algorithm is not guaranteed. However, a runtime verification test we used in our experiments (see Ch. 8) showed that *non-gapped* items are

removed from the queue in an ascending order of the $\beta + h$ weights, which would suggest that the A* parser based on h, as well as the Dijkstra-style parser, are nevertheless both correct.[34]

The A* parsing algorithm used with the adjunction-aware heuristic h_{adj} (cf. Def. 131) is correct, since h_{adj} is both admissible (cf. Prop. 42) and monotonic (cf. Prop. 43), provided that the underlying grammar is non-negative. In terms of the quality of its estimations, h_{adj} is often more accurate (i.e., closer to α) than h due to taking account of the weights related to the witness items. On the other hand, h is more accurate than h_{adj} when the words required by the underlying item are not present in the remaining part of the sentence. This advantage of h, however, comes with a significant computational cost when grammar compression techniques are used. Having to consider many different elementary trees which can correspond to a given chart item $\langle x, r \rangle$ and, consequently, many different sets of required words $req^1(x)$ can be inefficient.[35] h_{adj} deals with this issue much more aptly and the complexity of calculating its values remains constant regardless of the grammar compression techniques used.

Finally, the FS-aware parser, which extends the vanilla parser with the support for feature structures, is a proof of concept which shows a possible way of dealing with feature structures on the fly. Such approach has a potentially detrimental effect on parsing time and space complexity, since we propose to enrich chart items with FS values. ParTAGe provides support for flat FSs (with top/bottom distinction) only, hence this is not necessarily an issue – the space of the possible FS values is bounded in this case. Anyhow, handling FSs on the fly has important advantages within the context of A* parsing – notably, it guarantees that the first final item found by the algorithm corresponds to a solution which is not only optimal but also satisfies the unification-related requirements.

[34](Nederhof, 2003, p. 140) shows that an algorithm similar to our Dijkstra-style parser, applied to a non-monotonic weighted CFG parsing system with prediction, still correctly computes the derivations with the lowest weights.

[35]It is, of course, possible, and we relied on such an implementation in our experiments. However, it works against the benefits of grammar compression.

Chapter 8

Experimental evaluation

In this chapter, we describe an experimental evaluation of the MWE-promoting strategy (cf. Sec. 7.1) and its A* implementation (cf. Sec. 7.4.3) based on the heuristic h (cf. Def. 125). In all the experiments the grammar was compressed using the subtree sharing (cf. Def. 132) and the prefix tree (cf. Def. 133) encoding techniques. The main goals of this experiment are: (i) to examine the influence of the MWE-promoting strategy on the accuracy of a symbolic TAG parser, and (ii) to investigate the possible speed-up gains stemming from its A*-based implementation.

An evaluation of a parser which jointly performs MWE recognition and syntactic parsing relies on an appropriate syntactic treebank annotated with MWEs. In a statistical setting, such a treebank is first (i) divided into two parts – called *training* and *evaluation* – then, (ii) the parameters of the parser are estimated over the training part, next (iii) the resulting parser is used to process the evaluation part – i.e., identify the MWE occurrences therein, as well as determine the syntactic structures of the individual sentences – and, finally, (iv) the syntactic parsing accuracy and the MWE recognition accuracy are calculated on the basis of the discrepancies between the annotations present in the evaluation part and the parser's output.

In a symbolic setting, a similar procedure could be followed. After dividing the treebank into two parts, an MWE-aware grammar could be extracted from the training part and the ability of the parser to correctly identify MWEs and determine syntactic structures in the evaluation part could be measured. However, such a process would mainly ascertain the quality of the extraction method used to obtained the grammar from the training part, which is not our goal. Whether an underlying symbolic grammar is hand-

crafted or extracted from a treebank (or both), the MWE-promoting strategy is not able to correct the mistakes made during the syntactic analysis stage. Rather, it is a special instance of a disambiguation strategy, whose quality can be measured independently from that of the syntactic analysis (and, consequently, independently from the underlying grammar). We follow this principle in our experiments and evaluate the MWE-promoting strategy with respect to sentences which the parser is able to correctly analyse.

8.1 Data preparation

The experiment relies on two main resources: a treebank annotated with MWEs, and a MWE-aware TAG grammar compatible with the treebank. Sec. 8.1.1 contains a description of the procedure of mapping MWE resources on a Polish constituency treebank which we used to obtain a MWE-aware treebank, while Sec. 8.1.2 describes the grammar extraction method we used to get a TAG grammar compatible with this treebank.

8.1.1 Mapping MWE resources on a treebank

Let us note that, while we projected MWEs on a treebank for the specific purpose of the evaluation of the MWE promoting strategy, its utility can be seen in a broader context. Treebanks in which MWEs have been explicitly annotated are highly precious resources enabling us to study their more or less unpredictable properties. They also constitute basic prerequisites for training and evaluating MWE-aware syntactic parsers (as in our case). However, few treebanks contain full-fledged MWE annotations, even for English (Rosén et al., 2015), and lexical resources of MWEs develop more rapidly than MWE-annotated treebanks (Losnegaard et al., 2016). It is, thus, interesting to examine how far MWE lexicons can help in completing the existing treebanks with annotation layers dedicated to MWEs (Savary and Waszczuk, 2017).

Our case study in this respect deals with four Polish resources: (i) the named-entity annotation layer of a Polish reference corpus, (ii) an e-lexicon of nominal, adjectival and adverbial continuous MWEs, (iii) a valence dictionary containing a phraseological component, and (iv) a treebank with no initial MWE annotations. We show how the 3 former resources can be automatically projected on the latter, by identifying syntactic nodes satisfying (totally or partly) the appropriate lexical and syntactic constraints.

Resources

The National Corpus of Polish (NCP) (Przepiórkowski et al., 2012) contains a manually double-annotated and adjudicated subcorpus of over 1 million words. Its **named entity layer** (**NCP-NE**), which builds on the morphosyntactic layer (relying in its turn on the segmentation layer), contains over 80,000 annotated NEs, 20% of which are MWNEs. Only the latter were used in the experiments described below. The annotation schema assumes notably the markup of nested, overlapping and discontinuous NEs, i.e., the annotation structures form trees (Savary et al., 2010).

SEJF (Czerepowicka and Savary, 2015) is a grammatical lexicon of Polish continuous MWEs containing over 4,700 compound nouns, adjectives and adverbs, where inflectional and word-order variation is described via fine-grained graph-based rules. It is provided in two forms – intensional (multiword lemmas and inflection rules) and extensional (list of morphologically annotated variants). The latter, generated automatically from the former, was used in our projecting experiments. Tab. 8.1 shows a sample extensional entry containing a MWE inflected form, its lemma and morphological tag: noun (`subst`) in singular (`sg`) genitive (`gen`) and masculine inanimate gender (`m3`).

Inflected form	Lemma	Tag
sposobu bycia	*sposób bycia*	`subst:sg:gen:m3`

Table 8.1: Sample inflected form of *sposób bycia* (lit. *way of being*) 'manner' in SEJF.

Walenty is a Polish large-scale valence dictionary with over 8,000 frames which describe VMWEs (in its 2015 version). For instance the idiom highlighted in Ex. 8.1 is described in Walenty as shown in Tab. 8.2 (cf. Ex. 3.1 and Tab. 3.2, which we repeat here for the readability reasons). Refer to Sec. 3.4 for a more detailed description of this resource.

(8.1) Nie umiem w tych sprawach **trzymać języka za zębami**.
Not know$_{\text{SG.1}}$ in these affairs hold$_{\text{INF}}$ tongue$_{\text{SG.GEN}}$ behind teeth.

(lit. *I cannot hold my tongue behind my teeth in such cases*) 'I cannot hold my tongue in such cases'

```
trzymać: subj{np(str)}+
   obj{lex(np(str),sg,'język',natr)}+
   {lex(prepnp(za,inst),pl,'ząb',natr)}
```

Table 8.2: Walenty description of *trzymać język za zębami* 'hold one's tongue'

Składnica is a Polish constituency treebank comprising about 9,000 sentences with manually disambiguated syntactic trees (Świdziński and Woliński, 2010). It was created by automatically generating all possible parses with a large-coverage DCG grammar, and then manually selecting the correct parse. It does not contain MWE annotations. Its morphosyntactic tagset is mostly equivalent to the one used in Walenty, although it uses Polish terms: `mian`=*mianownik* 'nominative', `dk`=*dokonany* 'perfective aspect', etc.

Fig. 8.1 shows the correct syntax tree from Składnica for Ex. 8.1. Each non-terminal node includes a feature structure (FS). Here, the FS of the node `fno` (nominal phrase), above the terminal *język* 'tongue', is highlighted. It includes the feature `neg=nie` meaning that this node occurs within the scope of a negated verb. This enables an easy validation of some constraints from Walenty entries, such as the structural genitive of direct objects.

A notable feature of Składnica is that dependents of the verbs are explicitly marked as either arguments (`fw`) or adjuncts (`fl`), i.e., valency is accounted for. Note, however, that the valency of head verbs in VMWEs can differ from the one of the same verbs occurring as simple predicates.

Projection

Since Składnica contains no explicit MWE annotations, we produced them automatically by projecting NCP-NE, SEJF and Walenty on the syntax trees. The projection for NCP-NE was straightforward and did not require manual validation, since Składnica is a subcorpus of the NCP, whose NE annotation and adjudication were performed manually. The projection for SEJF and Walenty, followed by a manual validation, consisted in searching for syntactic nodes satisfying all lexical constraints and part of syntactic constraints of a MWE entry. The required lexical nodes were to be contiguous for SEJF but not for Walenty.

Here, we give more details on the Walenty-to-Składnica projection, which was the most challenging one. It required defining correspondences at different

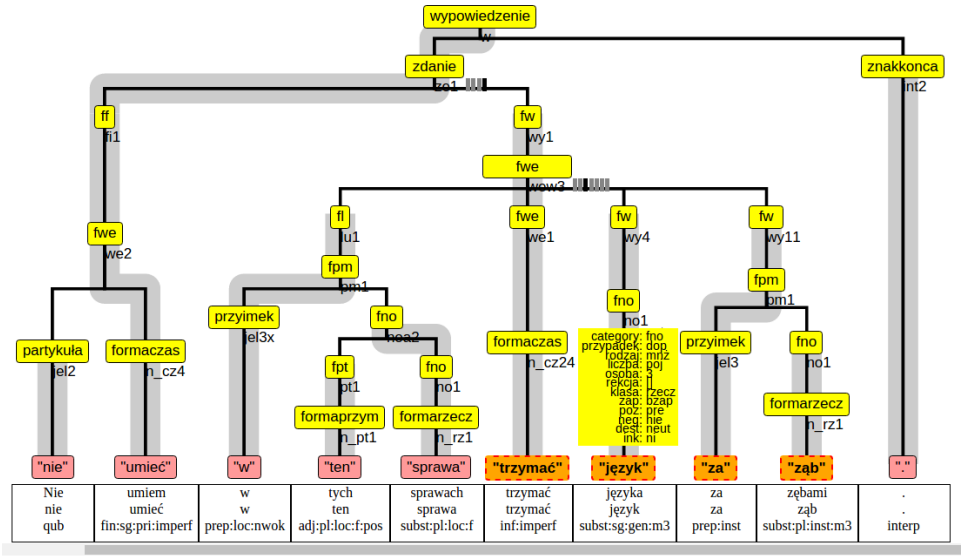


Figure 8.1: Syntax tree of Ex. 8.1 in Składnica. The categories denote: `ff` 'finite phrase', `fl` 'adjunct', `fno` 'nominal phrase', `formaczas` 'verbal phrase', `formaprzym` 'adjectival phrase', `formarzecz` 'nominal phrase', `fpm` 'prepositional phrase', `fpt` 'adjectival phrase', `fw` 'required phrase', `fwe` 'verbal phrase', `partykuła` 'particle', `przyimek` 'preposition', `wypowiedzenie` 'utterance', `zdanie` 'sentence', `znakkońca` 'ending punctuation'. The feature structure of the `fno` node dominating the terminal *język* 'tongue' is highlighted. The feature codes include: `przypadek` 'case', `rodzaj` 'gender', `liczba` 'number', `osoba` 'person', `rekcja` 'case government', and `neg` 'negation'. The values denote: `dop` 'genitive', `mnz` 'human inanimate', `poj` 'singular', and `nie` 'negated'.

levels. Explicit morphological values and phrase types could be translated rather straightforwardly due to largely compatible tagsets (`np`→`fno` 'nominal phrase', `mian`→`nom` 'nominative', etc.). Context-dependent values like `str` (structural case) were encoded in conditional statements taking combination of features into account. For instance, the argument specification `obj(np(str))` translated into a feature structure containing one of the following: $[category = fno, przypadek = bier, neg = tak]$, $[category = fno, przypadek = dop, neg = nie]$ (nominal object, either in the accusative in an affirmative sentence or in the genitive in a negative one).

Once these morphosyntactic correspondences were defined, the procedure

of identifying a Walenty MWE entry in Składnica consisted in checking if the current sentence contained a subtree in which: (i) the lexically constrained arguments and adjuncts (and their own, recursively embedded, lexically constrained dependents) were present, (ii) selected syntactic constraints (those concerning `np` and `prepnp` phrases) were fulfilled. For instance in Fig. 8.1, a head verb, a direct object with a lexicalized head and a lexicalized prepositional complement were searched for, but an ellipsis of the subject was allowed.

Query language

We split the task of MWE projection into three elements: (i) a query language, providing an interface between our MWE resources and the treebank, (ii) a set of dedicated procedures – one per each MWE resource – for compiling lexicon entries into the corresponding queries, and (iii) an interpreter which allows to run the queries over treebank nodes and, in particular, to answer the question whether the MWE entry – to which the executed query corresponds – occurs within the context of a given syntactic subtree.

Formally, we defined our core query language using the following abstract syntax:

$$
\begin{aligned}
&b\,(\text{Booleans}) ::= \texttt{true} \mid \texttt{false}\\
&n\,(\text{node queries}) ::= b \mid n_1 \wedge n_2 \mid n_1 \vee n_2 \mid \texttt{mark} \mid \texttt{satisfy}\,(\mathit{node} \rightarrow b)\\
&t\,(\text{tree queries}) ::= b \mid t_1 \wedge t_2 \mid t_1 \vee t_2 \mid \texttt{root}\,n \mid \texttt{child}\ t \mid \ldots
\end{aligned}
$$

Thus, the properties of a given syntactic node or tree can be verified via an appropriate node query (NQ) or tree query (TQ), respectively. Both kinds of queries are recursive and TQs can additionally build on NQs. For instance, from the query interpretation point of view, the TQ `root` n is satisfied for a given tree iff its root satisfies the NQ n. Also, the TQ `child` t is satisfied iff at least one of its root's children trees satisfies the TQ t. Finally, particular feature values (*category*, *przypadek*, etc.) can be verified using the NQ `satisfy`$(\mathit{node} \rightarrow b)$, which takes an arbitrary node-level predicate $(\mathit{node} \rightarrow b)$ and tells whether it is satisfied over the current syntactic *node*.

The particularity of this simple language of NQs and TQs is the `mark` construction, which allows to mark the currently considered syntactic node as a part of a MWE. When a TQ t, containing `mark`, is executed over a given tree, all the nodes matched with `mark` during the course of t's execution will

be returned as t's result, provided that the syntactic tree satisfies all the constraints encoded in t. `Mark` does not check any constraints by itself, but it can be easily combined with other NQs via query conjunction (i.e., $n \wedge$ `mark`).

Note that, based on our core language, more complex queries can be expressed, for instance:

$$\texttt{member}\ n \stackrel{\text{def}}{=} \texttt{root}\ n \vee \texttt{child}\ (\texttt{member}\ n) \tag{8.2}$$

which checks whether the current tree contains (either in its root or, recursively, in one of its subtrees) a node which satisfies the given node query n.

The query interpreter is defined over the core language only and handles MWE-related marking. For instance, given a query of type $t_1 \vee t_2$, while evaluating t_1, some subtree nodes may be marked as potential MWE components (by the `mark` operator). If, however, t_1 finally evaluates to `false`, then all these markings are wiped out. This behavior is guaranteed by an appropriate implementation of the core disjunction ($\vee$) operator.

Compiling MWE entries

Let us focus on the Walenty-to-query compilation and on the entry from Tab. 8.2 in particular. Its querified version checks that (i) the base form of the lexical head, reached via the head-annotated edges (grayed out in Fig. 8.1), corresponds to the main verb of the entry (i.e., *trzymać*), and (ii) each of the lexically-constrained elements of the frame (i.e., noun phrase *język* and prepositional phrase *za zębami*) is realized by one of the `child`-ren trees of the queried tree. Part (i) of the query is implemented by the version of the `member` query (see Eq. 8.2) restricted to head-annotated edges. Implementation of (ii) depends on the particular frame element. Tree queries corresponding to (i) and (ii) are then combined using the $\wedge$ operator.

The `obj{lex(np(str),sg,'język',natr)}` frame element is also translated to a $\wedge$-combined set of tree queries, which individually check that all the given restrictions are satisfied: the lexical head is *język*, the number is singular, etc. The node query which verifies that *język* is the lexical head is combined with `mark`, so that it is designated as a part of the resulting MWE annotation, provided that all the other entry-related constraints are also satisfied. Modifiers, if specified, are recursively compiled into tree queries which are then applied over `child`-ren trees. Here, `natr` specifies that no modifiers are allowed, constraint compiled into a query which checks that the

Source	True pos.	False pos.	Comp. readings	All	Comp. rate
NKJP	1,304	n/a	n/a	1,304	n/a
SEJF	368	18	23	409	0.94
Walenty	365	78	18	452	0.95
Total	2,037	96	41	2,165	0.95

Table 8.3: Projection results

corresponding tree is non-branching[1] (i.e., has no other children apart from its head, constraint satisfied in Fig. 8.1 by the subtree rooted with `fno` placed over the leaf *język*). The other element of the frame, which describes the prepositional argument *za zębami*, is compiled into a query in a similar way.

Projection results

Table 8.3 shows the projection results. Among the 2165 automatically identified candidate MWEs, those 1,304 stemming from NCP-NE were supposed correct (since resulting from manual double-annotation and adjudication). The 861 remaining candidates were manually validated. They contained 733 true positives, 96 false positives, and 41 candidates with a compositional reading, as in examples (8.3)-(8.4). Thus, the precision of the SEJF/Walenty projection was equal to 0.85. The idiomaticity rate (El Maarouf and Oakes, 2015), i.e., the ratio of occurrences with idiomatic reading to all correctly recognized occurrences, is about 0.95. We expect that if NEs were taken into account, this ratio would be even higher, since NEs seem to exhibit compositional readings relatively rarely. Note also that false positives are much more frequent for entries stemming from Walenty than for those from SEJF, which shows the higher complexity of verbal MWEs as compared to other, continuous, MWEs.

(8.3) ... w **drugiej połowie** XIX wieku
'... in the **second half** of the 19th century'
MWE: (lit. *second half*) 'one's husband or wife'

[1]The `non-branching` predicate is a part of the core language. We did not define it above for the sake of brevity.

(8.4) Nie **podał** Klossowi **ręki**, wskazał mu tylko krzesło.
'He did not **give** Kloss his **hand**, he just pointed at a chair.'
MWE: (lit. *give someone a hand*) 'help'

(8.5) Odetchnęła głęboko i **przymknęła oczy**.
'(She) breathed profoundly and **closed** her **eyes**.'
MWE: *przymknać oczy na coś* (lit. *to close one's eyes on sth*) 'to pretend not to see sth'

Notable errors in the projection procedure stem from allowing for the ellipsis of compulsory but non-lexicalized arguments. If all such arguments marked in Walenty were required in Składnica during the projection, correct MWEs occurrences with elided arguments would be missed, as in the case of the subject required in Tab. 8.2 but omitted in Fig. 8.1. Conversely, allowing for the ellipsis of such arguments results in some false positives, as in example (8.5), where the absence of the prepositional argument (headed by the preposition *na* 'on') excludes the idiomatic reading.

The result of the automatic projection of MWE resources on Składnica is available[2] under the GPL v3 license. The results are represented in a simplified custom XML format, meant for an easy use, e.g., in automatic grammar extraction. This format refers to identifiers of sentences and tokens in the Składnica trees, which enables users to automatically project annotations on the original treebank.

8.1.2 Grammar extraction

Following Chen and Shanker (2004), the first step to extract a LTAG grammar from a MWE-ignorant constituent treebank is to determine the status of the individual nodes in the individual syntactic trees. Let η' be a node immediately dominated by a node η. Then, η' can be either:

(a) On the path from η to its lexical head – we call such a path a *trunk*.

(b) A complement of η – when the lexical head of η' is a complement of η' head.

(c) An adjunct of η – when the lexical head of η' is a modifier of η's head.

[2] http://zil.ipipan.waw.pl/Sk%C5%82adnicaMWE

In Fig. 8.1, all the trunks are marked with grayed out edges. For instance, the trunk of the right `fwe` node (the one which dominates four other nodes) connects it with its lexical head, *trzymać* 'to hold'. Recall also that dependents of the verbs are explicitly marked as either arguments (`fw`) or adjuncts (`fl`).

In the case (a), both η' and η should end up on the trunk of the corresponding ET. In the case (b), a copy of η' should become a substitution non-terminal leaf of the ET to which η belongs, while another copy of η' should be placed in the root of a separate ET which attaches to the first copy via substitution. Finally, in the case (c), when η' is an adjunct of η, it should be modeled as a root of a separate EAT (to be determined on the basis of the syntactic subtree rooted in η') which adjoins to η.[3] Fig. 8.2 (a) shows the ETs extracted from the tree rooted in the right `fwe` node in Fig. 8.1, as well as the substitution and the adjunction relations which the extraction procedure entails.

In a MWE-aware treebank, the prominent difference is that a node can have several lexical heads, which are then modeled as the anchors of the corresponding ET. Consider again the tree rooted in the right `fwe` node in Fig. 8.1. In Składnica, it has one lexical head, *trzymać* 'to hold'. In the MWE-annotated variant of Składnica, all four lexical components – *trzymać* 'to hold', *język* 'tongue', *za* 'behind', and *ząb* 'tooth' – of the MWE identified in this tree can be interpreted as its lexical heads. Consequently, the trunk of the `fwe` node includes the paths to all the above-mentioned lexical components. Fig. 8.2 (b) shows the MWE ET extracted from the syntactic tree rooted in the right `fwe` node when the MWE annotation is taken into account.

Concerning the rules of deciding whether a given node η', directly dominated by η, is a complement or an adjunct of η, we followed – inspired by Krasnowska (2013) – the procedure as specified below:

- In case of the verbs, the status is already specified in Składnica via the `fw` and `fl` pseudo-syntactic nodes.[4] The `fw` and `fl` nodes are subsequently removed, i.e., they do not appear in the extracted ETs.

- Let x' be the label of η' and x be the label of η. Then, η' is a complement

[3] However, it is not alway possible to obtain such a behavior. In particular, if both the left and the right immediate sisters of η' are complements or trunk-elements of η, η' can be at best modeled as a modifier of one of its sister nodes.

[4] It seems that the only function of such nodes is, precisely, to mark the status of the nodes they dominate. Such information could be, perhaps more appropriately, placed in FSs.

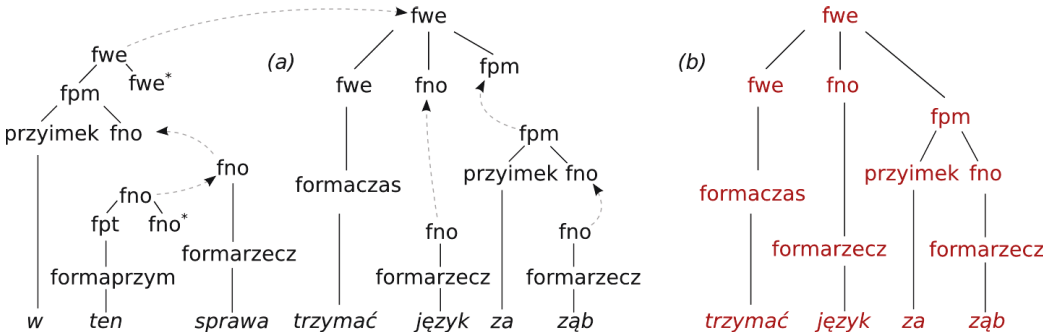


Figure 8.2: ETs extracted from the tree rooted in the right `fwe` node in Fig. 8.1. (a) The compositional derivation (the MWE annotation is ignored). (b) The ET corresponding to the MWE occurrence.

of η if: (i) x' = `fno` (nominal phrase) and x = `fpm` (prepositional phrase), or (ii) x' = `zdanie` (sentence) and x = `fzd` (embedded clause), or (iii) x' = `zdanie` and x = `zdanie`,[5] or (iv) x' = `spójnik`.[6] Otherwise, η' is an adjunct.

Let us note that, in our extraction procedure, we did not account for FSs and, consequently, the resulting grammar was a plain LTAG rather than a FS-based LTAG.

8.2 Evaluation

The evaluation of the MWE-promoting strategy consisted in parsing the 1566 Składnica sentences in which MWEs were identified[7] using the mapping procedure described in Sec. 8.1.1. The underlying TAG was extracted using the method described in 8.1.2, applied to all Składnica trees, so as to obtain ETs related to both the compositional derivations and the MWE-based ones.

As a baseline, we consider a "dummy" disambiguation strategy which consists in preserving all the derivations obtained via syntactic analysis. Put

[5]Occurring in coordinations of sentences.

[6]To avoid analyzing additional conjunctions – as in *Często chodzili do pubu, pili piwo, rozmawiali.* 'Often [they] were-going to pub, drunk beer, talked.', where only one of the commas is marked as a trunk-element – as adjuncts.

[7]Including the occurrences with a compositional reading, but not the false positives, based on the assumption that ETs corresponding to such compositional occurrences should also be present in the grammar.

differently, the baseline is a vanilla symbolic parser which does not perform any disambiguation and returns (in a compact, hypergraph form) all the grammar-compliant derivations. Since we perform the experiment over the trees from which the grammar was extracted, we obtain a baseline system with a perfect, 100% accuracy. Note that this is consistent with our choice of measuring the accuracy of the MWE-promoting disambiguation strategy with respect to the trees covered by the underlying grammar only.

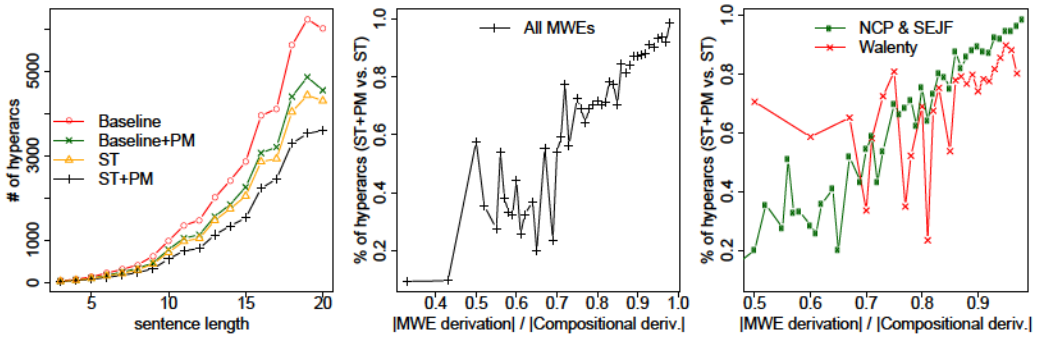
We assess the parser's efficiency (speed) in terms of the size of its parsing hypergraph (i.e., the number of its hyperarcs, cf. Def. 86). We believe it to be a more objective measure to compare different parsing strategies than the absolute parsing time, because it abstracts over the low-level implementation details which may significantly influence the parsing time depending on the optimization techniques involved. Each hyperarc corresponds to an application of an inference rule, i.e. to a basic parsing step, which we assume to be implementable in constant time.[8] Recall that the time complexity overhead related to computing the values of the heuristic h (see Def. 125) is at most linear in the size of the words required (cf. Def. 122) by a given chart item (cf. Prop. 36), provided that no grammar compression methods are used.

The baseline hypergraph is the one generated with the full grammar, when no MWE-promoting strategy is used and all grammar-compliant parses are generated for each sentence. The MWE-promoting (PM) hypergraph, compared to this baseline, includes mainly the optimal parses (the algorithm ensures that, in PM, all optimal parses are achieved, but some sub-optimal parses may also be reached, since a heuristic is an imperfect estimation of α), i.e., those in which the maximum number of words belongs to potential MWEs. To this end, Alg. 4 is slightly modified so as to stop when an item v with the total weight $\beta(v) + h(v)$ exceeding the total weight of the previously found final item is removed from the agenda.

The experiment was carried out on the dataset from which the grammar was extracted and on the basis of the heuristic h exclusively. Therefore, for each sentence, the baseline hypergraph contained both its gold (i.e., conforming to Składnica) parse (derived tree) and its gold MWE identification (derivation tree). The PM hypergraphs, in turn, contained the correct parses for virtually 100% of the sentences,[9] and correct MWE identification for

[8] We verified the rationality of this assumption, with respect to a version of the vanilla parser, in (Waszczuk et al., 2016a).

[9] A sanity check showed that for 54 sentences the gold parse was not found, mainly



(a) (b) (c)

Figure 8.3: (a) Average number of hyperarcs explored depending on the parsing strategy (for clarity using only sentence of length < 20), (b) Average % of hyperarcs explored with the PM+ST strategy, using the ST strategy as a reference, and (c) Average % of hyperarcs explored depending on the type of MWEs.

around 95% of them (due to the idiomaticity rate equal to 0.95). Thus, the parsing efficiency gain due to the PM strategy occurred with no loss of syntactic parsing accuracy and with a 5% loss of MWE recognition accuracy.

The PM strategy is comparable to supertagging (ST), i.e. pre-selecting, for each sentence, a subset of ETs which have good chances to be used in the derivation, in order to reduce the parsing search space. We experimented with a simple form of ST, which restricts the grammar to ETs whose terminals occur in the given sentence. Namely, we examined the ST hypergraph containing all parses for each sentence, and the one in which ST was combined with PM (where mainly optimal parses were achieved). Recall that, ideally, we would like the supertagger to provide us with sentence-dependent weights representing the plausibility of using the pre-filtered ETs in a derivation.

Fig. 8.3a shows the absolute sizes of the hypergraphs for these 4 strategies as a function of the sentence length. The PM strategy brings enhancement regardless of whether supertagging is used or not. The supertagging alone outperforms, on average, the baseline MWEs-promoting strategy. Since the combination of ST and PM strategies proves the most efficient (in particular, it allows to reduce the search space by around 40% for the sentences of length

due to some abbreviation- and letter-case-related specificities, as well as to missing MWE annotations in Składnica.

20), we restrict further experiments to this version.

Note that Fig. 8.3a does not fully reflect the potential advantages of the PM strategy, whose behavior does not directly depend on the length of the parsed sentence, but rather on the number and the size of the MWEs potentially occurring in it. These 2 values can be together represented as the ratio of the size of the MWE-based derivation tree to the size of the corresponding compositional derivation tree (i.e. the one assuming no MWE occurrence). Expectedly, as shown in Fig. 8.3b, the lower this ratio (i.e. the more words in the sentence belong to MWEs, and the longer are these MWEs), the more significant the hypergraph size reductions. As we can see, the PM strategy allows to reduce the search space by around 85% for the sentences with the highest MWE density. Moreover, the resulting graph suggests that the hypergraph size reductions are linear with respect to the above-mentioned ratio. Note that the vertical axis now shows the proportional gain in the hypergraph size due to the ST+PM strategy with respect to the ST strategy alone.

Finally, we investigated the behavior of the PM+ST strategy for two types of MWEs independently: verbal MWEs from Walenty and compounds from NCP and SEJF. As shown in Fig. 8.3c, verbal MWEs, while less frequent, prove to be better in reducing ambiguity for sentences with low number of potential MWEs. It is hard to ascertain this claim for sentences with lower gold derivation size ratio. While compounds seem to outperform verbal MWEs in this case, sentences with verbal MWEs for which this ratio is low are also very short in our dataset (of length 5, on average, for the 20 sentences with the lowest ratio), and thus exhibit low syntactic ambiguity.

Chapter 9

Future work

Recall that the probabilistic characterization of the MWE-promoting strategy (cf. Sec. 7.2) was inspired by the work of Lewis and Steedman (2014), who showed that it can be used for both efficient and accurate CCG parsing. Their system relies on the assumption that the probability of a given derivation is a product of the probabilities of the participating elementary grammar units. This might seem like a strong and unrealistic assumption – clearly, the probability of a combination of two lexicalized elementary units should be different depending on the words they represent. In different terms, this model does not account for bilexical relations which, after all, are known to play an important role in syntactic disambiguation.

The high accuracy of Lewis and Steedman (2014)'s CCG parser stems from the fact that the probabilities of the individual elementary grammar units (EUs) are not global, but rather computed for each input sentence separately before syntactic parsing takes place. Namely, a sequential, probabilistic supertagger is used to assign the probabilities to the individual EUs which can be potentially attached to the individual input words, based on local contextual information. This makes sense from the parsing architecture point of view because sequential models are simpler than syntactic parsing models, which means that obtaining the probabilities is significantly less costly than it would be if they were computed on the basis of a full-fledged probabilistic parsing model. The fact that the most probable sequence of EUs does not necessarily lead to a correct analysis is then handled by the parsing algorithm, which accepts only the syntactically correct (in the symbolic sense, i.e., those which would be present in the results of syntactic analysis) derivations.

The situation gets significantly complicated when MWEs enter the scene.

On the one hand, because of their scarcity, even simple probabilistic models might have trouble to reliably estimate their weights. On the other hand, MWEs can span over several words in the input sentence, potentially distant form each other, which excludes the possibility of use of the simple sequential supertagging models.

In future work, we plan to address this issue by assuming a distinction between MWE-related and simplex ETs: probabilities of the latter could be handled using sequential probabilistic supertagging methods, as in (Lewis and Steedman, 2014), while MWE-related ETs could be systematically promoted as in our current proposition. An alternative would be to design an extension of the word-lattice approach capable of handling discontinuous MWEs. Recall that regular word-lattices (cf. Sec. 5.4.3) can handle segmentation ambiguities related to continuous MWEs. Fig. 9.1 shows an example of a pseudo-lattice extension which accounts for discontinuous MWEs as well. It preserves the property that each path from the starting node to the final node represents a possible segmentation of the sentence. A sequential model applied to such a representation would provide us with positive weights assigned to the individual ETs, possible to reuse in A* parsing.

A clear weakness of the pseudo-lattice approach sketched above is that it does not preserve the order of the words in the input sentence, which may lead to poor estimations of the ETs' weights. Another possibility would be to use a variant of the lexical dependency tree model of Constant et al. (2016), which allows to capture deep lexical analyses like nested MWEs and to handle discontinuous MWEs. Segmentation ambiguities, including those related to discontinuous MWEs, could be then represented with a lexical dependency forest, obtained via a dependency parser modified so as to output a probability-annotated shared dependency forest rather than the most probably dependency tree. A challenge would be then to make such a dependency parser lexicon-driven, so that it outputs only (and all) the segmentations consistent with the ETs present in the underlying TAG.

We also plan to continue the grammar extraction experiments so as to obtain a high-quality, linguistically-motivated Polish TAG grammar. We think that an optimal method should rely not only on the syntactic treebank, but also on the other available lexical resources for Polish. In particular, the MWE-dedicated ETs could be directly compiled from the corresponding Walenty entries, rather than via their annotated occurrences in Składnica. This would allow to bypass the scarcity issue related to the fact that only a limited number of syntactic variants of the individual MWEs can be observed

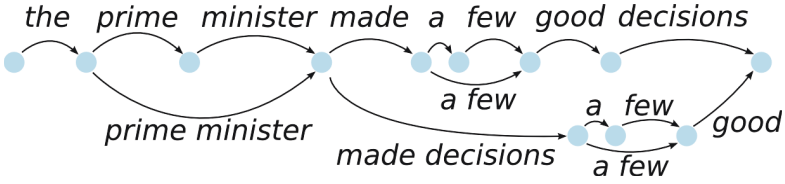


Figure 9.1: A variant of the lattice shown in Fig. 5.2, extended with an additional path related to the MWE interpretation of *made decisions*.

in such a small treebank. An alternative, promising approach we consider pursuing is to use a metagrammar framework, e.g. XMG (Crabbé et al., 2013), to obtain a core, linguistically-motivated TAG, ideally decorated with feature structures, and to use Składnica to: (i) complete it with alternative, automatically extracted ETs, so as to increase the grammar coverage, (ii) estimate the weights of the individual ETs.

We also plan to continue our work on ParTAGe. Firstly, we plan to integrate the FS-aware parser (cf. Sec. 7.5.3) with the A* parser (cf. Sec. 7.4.2) and to ascertain the advantages and disadvantages of the on-the-fly FS-unification in comparison with the alternative strategy – performing FS-unification in post-processing. Secondly, we plan to optimize the implementation of the A* parser so as to obtain the desired, close to constant-time behavior of the application of the inference rules when the grammar is compressed (cf. Sec. 7.5.2). Finally, we want to investigate the idea of the on-the-fly FS-unification in a context where FSs are potentially recursive. While in TAG FSs are typically restricted to flat structures, recursive FSs could be used to model semantic frames (Lichte and Petitjean, 2015). Furthermore, ParTAGe could then possibly serve as a parser for hybrid LFG/TAG grammars (Findlay, 2017).

Chapter 10

Conclusions

MWEs are linguistic objects containing two or more words and showing idiosyncratic behavior at different levels. Notably, their meaning is often not deducible from the meanings of their components and from their syntactic structure in a fully compositional way. Thus, interpretation-oriented NLP tasks, such as semantic calculus or translation, call for MWE-dedicated procedures. In this work, we focus on syntactic parsing, which often underlies such tasks. MWEs exhibit properties of both words and syntactic expressions, hence it is not clear what is an appropriate model to handle them in parsing, nor how they should be represented in syntactic treebanks (Rosén et al., 2015).

In order to better understand the relationship between MWE recognition and syntactic parsing, we first looked at how MWEs are represented in syntactic treebanks (cf. Ch. 4). We designed a novel classification of the MWE representation methods, with a particular focus on the MWE/syntax interface, and identified three appropriate representation approaches: *chunks* (for fixed, continuous MWEs), *overlay* (for syntactically regular expressions), and *bidirectional* (for both regular and irregular MWEs).

We also investigated the methods of dealing with MWEs in the existing parsing systems (cf. Ch. 5). In case of statistical methods, we used a classification based on the so-called orchestration question – at which point the MWE identification should take place: before, after, or during syntactic parsing (Constant et al., 2017), a dynamic counterpart of the MWE/syntax interface question examined before. Based on several MWE-related issues, we identified the *MWE-aware* approach (nota bene, the only approach which supports the bidirectional MWE/syntax interface) as the most promising in

dealing with both MWEs and syntactic structures. Two other approaches turned out as interesting alternatives – the word-lattice approach (related to the chunking interface and, thus, dealing with fixed MWEs only) and the re-ranking approach (related to the overlay approach).

However, among the statistical MWE-aware parsing methods we examined, none provides the precision necessary to fully account for the various MWE-related idiosyncrasies.[1] Numerous symbolic parsing frameworks exhibit a significantly extended domain of locality in comparison with purely statistical systems and, thus, allow to better deal with this issue. Among the symbolic formalisms we considered in our work, TAG stands out in allowing to describe MWEs and their idiosyncratic properties in a particularly natural way – i.e., as elementary grammar units which represent information about the corresponding subcategorization requirements, possible syntactic configurations, potentially irregular internal structure, lexical and morphosyntactic constraints, etc. From the parsing point of view, such elementary units can be seen as possibly discontinuous chunks which can be pre-recognized, thanks to their lexical nature, before syntactic parsing takes place. TAG, therefore, served as a natural framework to design a parsing architecture in which MWEs are exploited so as to make parsing both more accurate and faster.

In the domain of symbolic parsing, a growing interest is dedicated to parsing strategies – e.g., A* parsing – which allow to compute the most plausible parse tree(s) without having to generate the space of all the grammar-compliant solutions in advance. Such strategies enable efficient parsing within the context of large grammars and/or complex symbolic formalisms (Angelov and Ljunglöf, 2014), without sacrificing the parser's accuracy (Lewis and Steedman, 2014). However, the existing symbolic parsing strategies rarely pay attention to MWEs, even though the latter can account for more than 40% of lexical items in a natural language (Savary, 2014), potential MWE occurrences can help the parser to make better disambiguation decisions (Wehrli, 2014), and such occurrences are relatively easy to spot, due to their lexical nature.

Following the trend of improving efficiency in symbolic parsing, we designed a simple probabilistic characterization of the MWE promoting strategy and an A* heuristic which straightforwardly implements this strategy (cf. Ch. 7). Our experiments (cf. Ch. 8) confirmed the hypothesis that promoting MWEs

[1] Even if, as we believe, MWE-aware methods could be leveraged so as to overcome this challenge.

(more precisely, preferring TAG derivations based on MWE-related elementary trees) is a profitable strategy for a wide range of MWE types. We showed that up to 85% search-space reduction can be gained by guiding the parser to focus on the MWE-related derivations, and that the size of the parsing search space drops linearly in the number and size of the MWEs present in the sentence (Waszczuk et al., 2016b). We made use of additional grammar compression techniques so as to obtain results closer to a real-world setting, where different parsing enhancements should ideally combine to provide an optimal solution. Let us recall that the weighting scheme underlying our A* heuristic can be adapted so as to take corpus-based estimations of the probabilities of the individual ETs into account. Such estimations would then lead to further search-space reductions, independent from the MWE-related efficiency gains.

More importantly, we showed that speed-up gains can be achieved with virtually no loss in syntactic parsing accuracy, a relatively low-hanging fruit which is, nevertheless, beyond the reach of many symbolic A* parsers. Such parsers should be, therefore, extended with appropriate, MWE-aware A* components which would allow them to benefit from the possitive effects of promoting MWEs. Moreover, our results show the need for appropriate MWE-aware supertagging methods. In future work (cf. Ch. 9), we wish to investigate the lexical segmentation representations capable of handling both continuous and discontinuous MWEs, as well as the corresponding supertagging methods which would allow us to control the weights assigned to the individual ETs in a sentence-dependent manner (Lewis and Steedman, 2014). Such an approach would help in discriminating between the real, the literal, and the accidental occurrences of MWEs. In our particular case, this would help to deal with the 5% loss in the MWE recognition accuracy we observed in our experiments (Savary and Waszczuk, 2017).

Designing a symbolic parsing architecture which allows to promote MWEs in a principled way, so that it can be incorporated in a large-scale parsing system, is not a trivial task. We, thus, formalized a number of extensions (cf. Sec. 7.5) in order to show its practicality in a real-world setting, as well as to lay out the theoretical foundations for confirming our hypothesis (the benefits of promoting MWEs) with a real-sized TAG grammar. Firstly, we proposed an improved variant of the MWE-promoting A* heuristic which not only has better theoretical properties, but also is easier to compute. Its monotonicity guarantees the correctness of the parsing results and makes it safe to combine it with other A* heuristics. Secondly, we showed how to combine our A*

parsing strategy with grammar compression techniques. It remains to be proved that grammar compression and A* parsing are fully composable, but we believe that obtaining a constant-time behavior of the application of the inference rules is mostly a matter of low-level implementation optimizations. Finally, we showed how to extend the parser so as to account for feature structures (FSs), which are commonly used in practical TAGs and allow to model the idiosyncratic properties of MWEs. The FS-aware parser performs FS unification on the fly, which allows to combine it with A* parsing strategies.

Both the basic parsing architecture and its extensions were implemented in ParTAGe, a Haskell library available[2] under the BSD2 license. A front-end, command-line tool called ParTAGe4XMG, which allows to use ParTAGe with FS-aware TAGs generated with XMG (Crabbé et al., 2013), is also available[3] under the BSD2 license. The experiments we performed to verify our hypothesis were carried out on Składnica, a Polish constituency treebank. The automatic projection of three high-quality, MWE-aware Polish resources on this treebank resulted in a manually validated resource containing over 2,000 verbal MWEs in about 9,000 constituency trees, and available[4] under the GPL v3 license.

To conclude, we showed that the challenge of the MWE-related idiosyncrasies can be turned into an advantage in practical symbolic parsing. Namely, with TAGs, which provide first-class support for MWEs, and A* search strategies, considerable speed-up gains can be achieved by promoting MWE-based analyses with virtually no loss in syntactic parsing accuracy. This is in contrast to purely statistical state-of-the-art parsers, which, despite efficiency, provide no satisfactory support for MWEs. We contribute a TAG-A*-MWE-aware parsing architecture with facilities (grammar compression and feature structures) enabling real-world applications, easily extensible to a fully probabilistic framework.

[2] `https://github.com/kawu/partage`

[3] `https://github.com/kawu/partage4xmg`

[4] `http://zil.ipipan.waw.pl/Sk%C5%82adnicaMWE`

Bibliography

Abeillé, A. and Schabes, Y. (1989). Parsing Idioms in Lexicalized TAGs. In *Proceedings of the Fourth Conference on European Chapter of the Association for Computational Linguistics*, EACL '89, pages 1–9, Stroudsburg, PA, USA. Association for Computational Linguistics.

Al-Haj, H., Itai, A., and Wintner, S. (2014). Lexical Representation of Multiword Expressions in Morphologically-complex Languages. *International Journal of Lexicography*, 27(2):130.

Alahverdzhieva, K. (2008). XTAG using XMG. Master Thesis, Nancy Université.

Alonso, M., Cabrero, D., de la Clergerie, E. V., and Ferro, M. V. (1999). Tabular algorithms for TAG parsing. In *EACL 1999*, pages 150–157.

Angelov, K. and Ljunglöf, P. (2014). Fast statistical parsing with parallel multiple context-free grammars. In *EACL*, volume 14, pages 368–376.

Arun, A. and Keller, F. (2005). Lexicalization in Crosslinguistic Probabilistic Parsing: The Case of French. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 306–313. Association for Computational Linguistics.

Asudeh, A., Dalrymple, M., and Toivonen, I. (2013). Constructions with lexical integrity. *Journal of Language Modelling*, 1(1):1–54.

Attia, M. A. (2006). Accommodating Multiword Expressions in an Arabic LFG Grammar. In *Proceedings of the 5th International Conference on Advances in Natural Language Processing*, FinTAL'06, pages 87–98, Berlin, Heidelberg. Springer-Verlag.

Baldwin, T. and Kim, S. N. (2010). Multiword Expressions. In Indurkhya, N. and Damerau, F. J., editors, *Handbook of Natural Language Processing, Second Edition*, pages 267–292. CRC Press, Taylor and Francis Group, Boca Raton, FL. ISBN 978-1420085921.

Bangalore, S. and Joshi, A. K. (1999). Supertagging: An Approach to Almost Parsing. *Comput. Linguist.*, 25(2):237–265.

Bargmann, S. (2015). Syntactic flexibility of non-decomposable idioms. Poster presented at the 4th PARSEME general meeting, 18–19 March, Valletta, Malta.

Bejček, E., Panevová, J., Popelka, J., Straňák, P., Ševčíková, M., Štěpánek, J., and Žabokrtský, Z. (2012). Prague dependency treebank 2.5 – a revisited version of pdt 2.0. In *In Proceedings of the 24th International Conference on Computational Linguistics (Coling 2012)*, pages 231–246.

Bejček, E. and Straňák, P. (2016). Named Entities within Multiword Addresses. In *IC1207 COST PARSEME 6th general meeting*.

Candito, M. and Constant, M. (2014). Strategies for Contiguous Multiword Expression Analysis and Dependency Parsing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, Volume 1: Long Papers*, pages 743–753.

Candito, M. H. (1999). *Organisation modulaire et paramétrable de grammaires électroniques lexicalisées. Application au français et à l'italien*. Ph.D. dissertation, l'Université Paris 7.

Carreras, X. (2007). Experiments with a Higher-Order Projective Dependency Parser. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*.

Carroll, J., Nicolov, N., Shaumyan, O., Smets, M., and Weir, D. (1998). Grammar Compaction and Computation Sharing in Automaton-based Parsing. In *Proceedings of Tabulation in Parsing and Deduction (TAPD'98)*, pages 16–25, Paris, France.

Chen, J. and Shanker, V. K. (2004). Automated Extraction of Tags from the Penn Treebank. In Bunt, H., Carroll, J., and Satta, G., editors, *New Developments in Parsing Technology*, pages 73–89. Kluwer Academic Publishers, Norwell, MA, USA.

Clark, S. and Curran, J. R. (2007). Wide-Coverage Efficient Statistical Parsing with CCG and Log-Linear Models. *Computational Linguistics*, 33(4).

Constant, M., Candito, M., and Seddah, D. (2013a). The LIGM-Alpage Architecture for the SPMRL 2013 Shared Task: Multiword Expression Analysis and Dependency Parsing. In *Fourth Workshop on Statistical Parsing of Morphologically Rich Languages*, pages 46–52, Seattle, United States.

Constant, M., Eryiğit, G., Ramisch, C., Rosner, M., and Schneider, G. (2017). Statistical MWE-aware parsing. In Parmentier, Y. and Waszczuk, J., editors, *MWE Representation and Parsing*. Language Science Press (to appear).

Constant, M., Le Roux, J., and Tomeh, N. (2016). Deep Lexical Segmentation and Syntactic Parsing in the Easy-First Dependency Framework. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1095–1101. Association for Computational Linguistics.

Constant, M. and Nivre, J. (2016). A Transition-Based System for Joint Lexical and Syntactic Analysis. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 161–171, Berlin, Germany. Association for Computational Linguistics.

Constant, M., Roux, J. L., and Sigogne, A. (2013b). Combining compound recognition and PCFG-LA parsing with word lattices and conditional random fields. *ACM Trans. Speech Lang. Process.*, 10(3):8:1–8:24.

Constant, M., Sigogne, A., and Watrin, P. (2012). Discriminative strategies to integrate multiword expression recognition and parsing. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers - Volume 1*, ACL '12, pages 204–212, Stroudsburg, PA, USA. Association for Computational Linguistics.

Crabbé, B. (2014). An lr-inspired generalized lexicalized phrase structure parser. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 541–552. Dublin City University and Association for Computational Linguistics.

Crabbé, B., Duchier, D., Gardent, C., Le Roux, J., and Parmentier, Y. (2013). XMG: eXtensible MetaGrammar. *Computational Linguistics*, 39(3):591–629.

Czerepowicka, M. and Kosek, I. (2011). Problemy opisu związków frazeologicznych w formalizmie "Multifleks" (na przykładzie rodzaju wyrażeń frazeologicznych). In Bańko, M. and Kopcińska, D., editors, *Różne formy, różne treści*, pages 117—-126. Warszawa, Polska: Wydział Polonistyki Uniwersytetu Warszawskiego.

Czerepowicka, M. and Savary, A. (2015). SEJF - a Grammatical Lexicon of Polish Multi-Word Expression. In *Proceedings of Language and Technology Conference (LTC'15), Poznań, Poland*. Wydawnictwo Poznańskie.

Dalrymple, M. (2006). Lexical Functional Grammar. In Brown, K., editor, *Encyclopedia of Language & Linguistics (Second Edition)*, pages 82 – 94. Elsevier, Oxford, second edition edition.

Dyvik, H., Losnegaard, G. S., and Rosén, V. (2017). Multiword expressions in NorGram. In Parmentier, Y. and Waszczuk, J., editors, *MWE Representation and Parsing*. Language Science Press (to appear).

El Maarouf, I. and Oakes, M. (2015). Statistical Measures for Characterising MWEs. In *IC1207 COST PARSEME 5th general meeting*.

Evans, R. and Weir, D. (1997). Automaton-based Parsing for Lexicalized Grammars. In *In Proceedings of the Fifth International Workshop on Parsing Technologies*, pages 66–76, Boston, MA.

Findlay, Y. J. (2017). *Proceedings of the 13th Workshop on Multiword Expressions (MWE 2017)*, chapter Multiword expressions and lexicalism: the view from LFG, pages 73–79. Association for Computational Linguistics.

Finkel, J. R. and Manning, C. D. (2009). Joint Parsing and Named Entity Recognition. In *HLT-NAACL*, pages 326–334. The Association for Computational Linguistics.

Francez, N. and Wintner, S. (2012). *Unification Grammars*. Cambridge University Press, New York, NY, USA.

Gallo, G., Longo, G., Pallottino, S., and Nguyen, S. (1993). Directed hypergraphs and applications. *Discrete Appl. Math.*, 42(2-3):177–201.

Gardent, C. and Kallmeyer, L. (2003). Semantic construction in feature-based TAG. In *10th meeting of the European Chapter of the Association for Computational Linguistics - EACL'03*, page 8 p, Budapest, Hungary. Colloque avec actes et comité de lecture. internationale.

Gardent, C. and Narayan, S. (2015). Multiple Adjunction in Feature-Based Tree-Adjoining Grammar. *Computational Linguistics*, 41(1):41–70.

Green, S., de Marneffe, M.-C., and Manning, C. D. (2013). Parsing Models for Identifying Multiword Expressions. *Computational Linguistics*, 39(1):195–227.

Hajič, J., Panevová, J., Urešová, Z., Bémová, A., Kolárová, V., and Pajas, P. (2003). PDT-VALLEX: Creating a large-coverage valency lexicon for treebank annotation. In *Proceedings of the second workshop on treebanks and linguistic theories*, volume 9, pages 57–68.

Hajič, J. and Urešová, Z. (2003). Linguistic annotation: from links to cross-layer lexicons.

Huang, L. and Chiang, D. (2005). *Proceedings of the Ninth International Workshop on Parsing Technology*, chapter Better k-best Parsing, pages 53–64. Association for Computational Linguistics.

Joshi, A. K. and Schabes, Y. (1997). Tree-Adjoining Grammars. In Rozenberg, G. and Salomaa, A., editors, *Handbook of Formal Languages*, pages 69–123. Springer Berlin Heidelberg.

Jurafsky, D. and Martin, J. H. (2008). *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Pearson, 2nd edition.

Kahane, S. (2012). Why to choose dependency rather than constituency for syntax: a formal point of view. In Apresjan, J. D., Boguslavsky, I., L'Homme, M.-C., Iomdin, L., Milicevic, J., Polguère, A., and Wanner,

L., editors, *Meanings, Texts, and other exciting things: A Festschrift to Commemorate the 80th Anniversary of Professor Igor A. Mel'cuk*, pages 257–272.

Kallmeyer, L. (2010). *Parsing Beyond Context-Free Grammars*. Springer Publishing Company, Incorporated, 1st edition.

Klein, D. and Manning, C. D. (2001a). Parsing and Hypergraphs. In *Proceedings of the Seventh International Workshop on Parsing Technologies (IWPT-2001), 17-19 October 2001, Beijing, China*. Tsinghua University Press.

Klein, D. and Manning, C. D. (2001b). Parsing with Treebank Grammars: Empirical Bounds, Theoretical Models, and The Structure of the Penn Treebank.

Klein, D. and Manning, C. D. (2003). A* Parsing: Fast Exact Viterbi Parse Selection. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*.

Korkontzelos, I. and Manandhar, S. (2010). Can recognising multiword expressions improve shallow parsing? In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 636–644. Association for Computational Linguistics.

Krasnowska, K. (2013). Towards a polish LTAG grammar. In Klopotek, M. A., Koronacki, J., Marciniak, M., Mykowiecka, A., and Wierzchon, S. T., editors, *Language Processing and Intelligent Information Systems - 20th International Conference, IIS 2013, Warsaw, Poland, June 17-18, 2013. Proceedings*, volume 7912 of *Lecture Notes in Computer Science*, pages 16–21. Springer.

Le Roux, J., Constant, M., and Rozenknop, A. (2014). Syntactic Parsing and Compound Recognition via Dual Decomposition: Application to French. In *COLING 2014, the 25th International Conference on Computational Linguistics*, pages 1875–1885, Dublin, Ireland.

Lewis, M. and Steedman, M. (2014). A* CCG Parsing with a Supertag-factored Model. In *Proceedings of the 2014 Conference on Empirical Methods*

in Natural Language Processing (EMNLP), pages 990–1000. Association for Computational Linguistics.

Lichte, T. and Petitjean, S. (2015). Implementing semantic frames as typed feature structures with XMG. *Journal of Language Modelling*, 3(1):185–228.

Lichte, T., Petitjean, S., Savary, A., and Waszczuk, J. (2017). Lexical encoding formats for multiword expressions: The challenge of "irregular" regularities. In Parmentier, Y. and Waszczuk, J., editors, *MWE Representation and Parsing*. Language Science Press (to appear).

Losnegaard, G. S., Sangati, F., Escartín, C. P., Savary, A., Bargmann, S., and Monti, J. (2016). PARSEME Survey on MWE Resources. In Calzolari, N., Choukri, K., Declerck, T., Grobelnik, M., Maegaard, B., Mariani, J., Moreno, A., Odijk, J., and Piperidis, S., editors, *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, Paris, France. European Language Resources Association (ELRA).

McDonald, R., Pereira, F., Ribarov, K., and Hajič, J. (2005). Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 523–530. Association for Computational Linguistics.

Mel'čuk, I. (2012). Phraseology in the language, in the dictionary, and in the computer. *Yearbook of Phraseology*, 3(1):31–56.

Miyao, Y. and Tsujii, J. (2008). Feature forest models for probabilistic HPSG parsing. *Computational linguistics*, 34(1):35–80.

Nagy T., I. and Vincze, V. (2014). VPCTagger: Detecting Verb-Particle Constructions With Syntax-Based Methods. In *Proceedings of the 10th Workshop on Multiword Expressions (MWE)*, pages 17–25, Gothenburg, Sweden. Association for Computational Linguistics.

Nasr, A., Béchet, F., Rey, J.-F., Favre, B., and Le Roux, J. (2011). MACAON : An NLP Tool Suite for Processing Word Lattices. In *ACL 2011*, pages 86–91, United States.

Nasr, A. and Rambow, O. (2010). *Non-lexical chart parsing for TAG*, chapter Complexity of Lexical Descriptions and its Relevance to Natural Language Processing: A Supertagging Approach, pages 193–217. MIT Press.

Nasr, A., Ramisch, C., Deulofeu, J., and André, V. (2015). Joint Dependency Parsing and Multiword Expression Tokenisation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL'15)*.

Nederhof, M.-J. (1998). An alternative LR algorithm for TAGs. In *COLING 1998 Volume 2: The 17th International Conference on Computational Linguistics*.

Nederhof, M.-J. (2003). Weighted Deductive Parsing and Knuth's Algorithm. *Comput. Linguist.*, 29(1):135–143.

Nielsen, L. R., Andersen, K. A., and Pretolani, D. (2005). Finding the K Shortest Hyperpaths. *Comput. Oper. Res.*, 32(6):1477–1497.

Nivre, J. (2005). Dependency grammar and dependency parsing. MSI report 5133.1959.

Nivre, J., de Marneffe, M.-C., Ginter, F., Goldberg, Y., Hajič, J., Manning, C. D., McDonald, R., Petrov, S., Pyysalo, S., Silveira, N., Tsarfaty, R., and Zeman, D. (2016). Universal Dependencies v1: A Multilingual Treebank Collection. In Chair), N. C. C., Choukri, K., Declerck, T., Goggi, S., Grobelnik, M., Maegaard, B., Mariani, J., Mazo, H., Moreno, A., Odijk, J., and Piperidis, S., editors, *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, Paris, France. European Language Resources Association (ELRA).

Nivre, J. and Nilsson, J. (2004). Multiword Units in Syntactic Parsing. In *Proceedings of MEMURA 2004 – Methodologies and Evaluation of Multiword Units in Real-World Applications, Workshop at LREC 2004, May 25, 2004, Lisbon, Portugal*, pages 39–46, Lisbon, Portugal.

Parmentier, Y., Kallmeyer, L., Lichte, T., Maier, W., and Dellert, J. (2008). TuLiPA: A Syntax-Semantics Parsing Environment for Mildly Context-Sensitive Formalisms. In *9th International Workshop on Tree-Adjoining Grammar and Related Formalisms (TAG+9)*, pages 121–128, Tübingen, Germany.

Patejuk, A. (2015). *Unlike coordination in Polish: an LFG account*. Ph.D. dissertation, Institute of Polish Language, Polish Academy of Sciences, Cracow.

Patejuk, A. (2016). Integrating a rich external valency dictionary with an implemented XLE/LFG grammar. In Arnold, D., Butt, M., Crysmann, B., King, T. H., and Müller, S., editors, *Proceedings of the Joint 2016 Conference on Head-driven Phrase Structure Grammar and Lexical Functional Grammar*, pages 520–540, Stanford, CA. CSLI Publications.

Pauls, A. and Klein, D. (2009). K-Best A* Parsing. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 958–966. Association for Computational Linguistics.

Prolo, C. A. (2002). Fast LR Parsing Using Rich (Tree Adjoining) Grammars. In *Proceedings of EMNLP'02*, pages 103–110.

Przepiórkowski, A., Bańko, M., Górski, R. L., and Lewandowska-Tomaszczyk, B., editors (2012). *Narodowy Korpus Języka Polskiego [Eng.: National Corpus of Polish]*. Wydawnictwo Naukowe PWN, Warsaw.

Przepiórkowski, A., Hajič, J., Hajnicz, E., and Urešová, Z. (2017). Phraseology in Two Slavic Valency Dictionaries: Limitations and Perspectives. *International Journal of Lexicography*, 30(1):1.

Przepiórkowski, A., Hajnicz, E., Patejuk, A., and Woliński, M. (2014). Extended phraseological information in a valence dictionary for NLP applications. In *Proceedings of the Workshop on Lexical and Grammatical Resources for Language Processing (LG-LP 2014)*, pages 83–91, Dublin, Ireland. Association for Computational Linguistics and Dublin City University.

Rambow, O. (2010). The Simple Truth about Dependency and Phrase Structure Representations: An Opinion Piece. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 337–340. Association for Computational Linguistics.

Resnik, P. (1992). Probabilistic tree-adjoining grammar as a framework for statistical natural language processing. In *COLING 1992 Volume 2: The 15th International Conference on Computational Linguistics*.

Rosén, V., Losnegaard, G. S., De Smedt, K., Bejček, E., Savary, A., Przepiórkowski, A., Osenova, P., and Barbu Mitetelu, V. (2015). A survey of multiword expressions in treebanks. In *Proceedings of the 14th International Workshop on Treebanks & Linguistic Theories conference*, Warsaw, Poland.

Sag, I. A., Baldwin, T., Bond, F., Copestake, A., and Flickinger, D. (2002). Multiword expressions: A pain in the neck for NLP. In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 1–15. Springer Berlin Heidelberg.

Sagot, B. and Boullier, P. (2005). From Raw Corpus to Word Lattices: Robust Pre-parsing Processing with SxPipe. *Archives of Control Sciences*, 15(4):653–662.

Satta, G. (1994). Tree-Adjoining Grammar Parsing and Boolean Matrix Multiplication. *Computational Linguistics, Volume 20, Number 2, June 1994*.

Savary, A. (2005). A Formalism for the Computational Morphology of Multi-Word Units. *Archives of Control Sciences*, Vol. 15, no. 3:437–449.

Savary, A. (2014). *Representation and Processing of Composition, Variation and Approximation in Language Resources and Tools*. Habilitation à diriger des recherches, Université François Rabelais Tours.

Savary, A. and Waszczuk, J. (2012). Narzędzia do anotacji jednostek nazewniczych. In Przepiórkowski et al. (2012), pages 225–252.

Savary, A. and Waszczuk, J. (2017). Projecting Multiword Expression Resources on a Polish Treebank. In *Proceedings of the 6th Workshop on Balto-Slavic Natural Language Processing*.

Savary, A., Waszczuk, J., and Przepiórkowski, A. (2010). Towards the Annotation of Named Entities in the National Corpus of Polish. In *Proceedings of the Seventh conference on International Language Resources and Evaluation (LREC'10)*. European Languages Resources Association (ELRA).

Schabes, Y. and C. Waters, R. (1995). Tree Insertion Grammar: A Cubic-Time, Parsable Formalism that Lexicalizes Context-Free Grammar without Changing the Trees Produced. *Computational Linguistics, Volume 21, Number 4, December 1995*.

Schabes, Y. and M. Shieber, S. (1994). An Alternative Conception of Tree-Adjoining Derivation. *Computational Linguistics, Volume 20, Number 1, March 1994*.

Schneider, N., Danchik, E., Dyer, C., and Smith, N. A. (2014). Discriminative Lexical Semantic Segmentation with Gaps: Running the MWE Gamut. *Transactions of the Association for Computational Linguistics*, 2:193–206.

Sgall, P. and Hajicová, Eva andPanevová, J. (1986). *The Meaning of the Sentence in Its Semantic and Pragmatic Aspects*. Springer.

Sheinfux, L. H., Greshler, T. A., Melnik, N., and Wintner, S. (2015). Hebrew Verbal Multi-Word Expressions. *Head-Driven Phrase Structure Grammar*, page 122.

Shen, L. and Joshi, A. (2005). Incremental LTAG Parsing. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*.

Shieber, S. M., Schabes, Y., and Pereira, F. C. (1995). Principles and implementation of deductive parsing. *The Journal of logic programming*, 24(1):3–36.

Sutton, C. and McCallum, A. (2011). An Introduction to Conditional Random Fields. *Machine Learning*, 4(4):267–373.

Swanson, B., Yamangil, E., Charniak, E., and Shieber, S. (2013). A Context Free TAG Variant. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 302–310. Association for Computational Linguistics.

Tesnière, L. (1959). *Éléments de syntaxe structurale*. Klincksieck.

Urešová, Z. (2009). Building the PDT-VALLEX valency lexicon. In *Online proceedings of the fifth Corpus Linguistics Conference. University of Liverpool*.

Vijay-Shanker, K. (1987). *A Study of Tree Adjoining Grammars.* Ph.D. dissertation, Department of Computer and Information Science, University of Pennsylvania, Philadelphia.

Vijay-Shanker, K. and Joshi, A. (1988). Feature Structures Based Tree Adjoining Grammars. In *Coling Budapest 1988 Volume 2: International Conference on Computational Linguistics*.

Villemonte de La Clergerie, É. (2010). Building factorized TAGs with meta-grammars. In *The 10th International Conference on Tree Adjoining Grammars and Related Formalisms - TAG+10*, New Haven, CO, USA.

Villemonte De La Clergerie, É. (2013). Improving a symbolic parser through partially supervised learning. In *The 13th International Conference on Parsing Technologies (IWPT)*, Naria, Japan.

Vincze, V., Zsibrita, J., and Nagy, I. (2013). Dependency Parsing for Identifying Hungarian Light Verb Constructions. In *IJCNLP*, pages 207–215.

Waszczuk, J. (2012). Harnessing the CRF Complexity with Domain-Specific Constraints. The Case of Morphosyntactic Tagging of a Highly Inflected Language. In *Proceedings of COLING 2012*, pages 2789–2804. The COLING 2012 Organizing Committee.

Waszczuk, J., Głowińska, K., Savary, A., Przepiórkowski, A., and Lenart, M. (2013). Annotation tools for syntax and named entities in the National Corpus of Polish. *International Journal of Data Mining, Modelling and Management*, 5(2):103–122.

Waszczuk, J., Savary, A., and Parmentier, Y. (2016a). Enhancing practical TAG parsing efficiency by capturing redundancy. In *21st International Conference on Implementation and Application of Automata (CIAA 2016)*, Proceedings of the 21st International Conference on Implementation and Application of Automata (CIAA 2016), Séoul, South Korea.

Waszczuk, J., Savary, A., and Parmentier, Y. (2016b). Promoting multiword expressions in A* TAG parsing. In *COLING 2016, 26th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, December 11-16, 2016, Osaka, Japan*, pages 429–439.

Weeds, J., Dowdall, J., Schneider, G., Keller, B., and Weir, D. (2007). Using distributional similarity to organise biomedical terminology. *Application-Driven Terminology Engineering*, 2(97):107–41.

Wehrli, E. (2014). The Relevance of Collocations for Parsing. In *Proceedings of the 10th Workshop on Multiword Expressions (MWE)*, pages 26–32, Gothenburg, Sweden. Association for Computational Linguistics.

Woliński, M. (2006). Morfeusz — a Practical Tool for the Morphological Analysis of Polish. In Kłopotek, M. A., Wierzchoń, S. T., and Trojanowski, K., editors, *Intelligent Information Processing and Web Mining*, Advances in Soft Computing, pages 503–512. Springer-Verlag, Berlin.

Woliński, M. (2014). Morfeusz Reloaded. In Calzolari, N., Choukri, K., Declerck, T., Loftsson, H., Maegaard, B., Mariani, J., Moreno, A., Odijk, J., and Piperidis, S., editors, *Proceedings of the Ninth International Conference on Language Resources and Evaluation, LREC 2014*, pages 1106–1111, Reykjavík, Iceland. ELRA.

Świdziński, M. and Woliński, M. (2010). Towards a Bank of Constituent Parse Trees for Polish. In Sojka, P., Horák, A., Kopeček, I., and Pala, K., editors, *Text, Speech and Dialogue: 13th International Conference, TSD 2010, Brno, Czech Republic*, volume 6231 of *Lecture Notes in Artificial Intelligence*, pages 197–204, Heidelberg. Springer-Verlag.