Restricted Non-Projectivity: Coverage vs. Efficiency

Carlos Gómez-Rodríguez*
Universidade da Coruña

In the last decade, various restricted classes of non-projective dependency trees have been proposed with the goal of achieving a good tradeoff between parsing efficiency and coverage of the syntactic structures found in natural languages. We perform an extensive study measuring the coverage of a wide range of such classes on corpora of 30 languages under two different syntactic annotation criteria. The results show that, among the currently known relaxations of projectivity, the best tradeoff between coverage and computational complexity of exact parsing is achieved by either 1-endpoint-crossing trees or MH_k trees, depending on the level of coverage desired. We also present some properties of the relation of MH_k trees to other relevant classes of trees.

1. Introduction

A syntactic dependency tree is **projective** if the yield of each node is a substring of the sentence—or equivalently, if no dependencies cross when drawn above the words.[1] Projectivity is advantageous for efficient parsing: Exact inference for parsing models restricted to projective trees can be achieved in cubic time (Eisner 1996), and shift-reduce parsers can process them with very simple transitions in linear time (Nivre 2006). For this reason, and because crossing dependencies have traditionally been rare in corpora of languages like English, Chinese, or Japanese, many implementations of dependency parsers assume projectivity (Nivre 2006).

However, crossing dependencies are needed to represent some linguistic phenomena like topicalization, scrambling, *wh*-movement, or extraposition, so it is necessary for natural language parsers to support non-projectivity, especially when working with languages with flexible word order. Unfortunately, exact inference is intractable for models that support arbitrary non-projective trees, except under strong independence assumptions (McDonald and Satta 2007). For this reason, researchers have proposed various classes of **mildly non-projective** trees: restricted classes of trees that allow a limited degree of non-projectivity, permitting crossing dependencies only under certain

* Research Group on Language and Information Society (LyS), Departamento de Computación, Universidade da Coruña, Campus de A Coruña, 15071, A Coruña, Spain. E-mail: carlos.gomez@udc.es.

1 We will assume the conventional representation of syntactic dependency analyses as trees rooted at a dummy node. Its presence and location (Ballesteros and Nivre 2013) has no effect on the coverage of the considered classes of trees, except for 1-endpoint-crossing trees and crossing-interval trees. In these cases, we assume that the dummy root is located on the left, as in the papers in which they were defined.

Submission received: 4 February 2016; accepted for publication: 25 April 2016.

doi:10.1162/COLI_a_00267

conditions. The goal of these classes is to combine a high coverage of the syntactic phenomena found in real sentences with efficient parsing.[2]

In this article, we perform a comparison of a wide range of these relaxations of projectivity, with the goal of evaluating them in terms of the tradeoff between coverage and efficiency. For this purpose, we measure their coverage on a set of syntactic treebanks of 30 languages, analyzed under two different annotation criteria.

Thus, the main contribution of this work is that we provide homogeneous measurements of the coverage of a wide range of mildly non-projective classes of trees on a large collection of treebanks, relating them to their computational properties for parsing. To our knowledge, this is the first study providing an extensive comparison of such classes: Although Havelka (2007) also measured the coverage of several restrictions on non-projectivity, little was known at the time about which restrictions could be exploited for efficient parsing, so only a few of the classes discussed there are relevant for parsing. Furthermore, existing coverage data in the literature (both in that study and in the papers describing subsequently discovered classes of trees, cited herein) refer to small sets of treebanks that vary across reports, when reported at all.

Additionally, we present some results relating MH_k trees, one of the sets with the best coverage–efficiency tradeoff, with other classes of mildly non-projective trees.

2. Classes of Mildly Non-Projective Trees

We now list the classes of trees considered in this study, outlining them very briefly. A full description of each class, with all the required definitions, is outside the scope of this article. We refer the reader to the provided references for further information.

Projective. Projective dependency trees can be parsed in $O(n^3)$ (see Section 1). We will denote the set of projective trees by **Pr**.

Well-nested with Bounded Gap Degree. Well-nested trees (Bodirsky, Kuhlmann, and Möhl 2005) are those that do not contain disjoint subtrees whose yields interleave (those that do are called ill-nested). Well-nested trees whose gap degree (the number of discontinuities—or **gaps**—in a node's yield) does not exceed a constant k can be parsed in time $O(n^{5+2k})$ (Gómez-Rodríguez, Weir, and Carroll 2009; Gómez-Rodríguez, Carroll, and Weir 2011); and we will call them $\mathbf{WG}_k$ trees. WG_k trees have connections to constituent grammar formalisms, as tree-adjoining grammars induce WG_1 trees and coupled context-free grammars induce WG_k trees (Kuhlmann 2010).

Mild+1-Inherit and Gap-Minding. Gap inheritance (Pitler, Kannan, and Marcus 2012) is a restriction on the number of children of a node that can have arcs that cross a gap in its yield. Imposing gap inheritance bounds as additional restrictions on WG_1 trees, two relevant classes of trees are obtained: Mild+1-Inherit (**M1I**) trees can be parsed in $O(n^6)$, and Mild+0-Inherit (**M0I**) trees, or gap-minding trees, in $O(n^5)$.

Head-Split. The head-split property is a restriction that forbids trees where a node's yield has a gap that includes its head, but not the gap in its head's yield. This allows dynamic programming parsers to split subtrees into two at the position of their heads, reducing the complexity of parsing several subclasses of WG_1 trees: Satta and Kuhlmann (2013)

2 Another option is to use models that forgo exact inference, but still achieve competitive results for non-projective parsing in quadratic (Nivre 2008) or even linear time (Nivre 2009).

show how WG_1 trees with the head-split property (**WG_1S**) can be parsed in $O(n^6)$, whereas for M1I trees with the head-split property (**M1IS**) the complexity is $O(n^5)$.

Mildly Ill-nested. A superset of WG_k trees, mildly ill-nested trees of gap degree up to k (**MG_k**) include all the dependency trees that have at least one binarization of gap degree k. They can be parsed in time $O(n^{4+3k})$ (Gómez-Rodríguez, Carroll, and Weir 2011). Note that this is the same complexity as for WG_k for $k = 1$, but larger for $k > 1$.

Attardi Degree 2. The set of trees that can be parsed with the transitions of degree up to 2 in the transition system of Attardi (2006) is also amenable to dynamic programming parsing, in time $O(n^7)$ (Cohen, Gómez-Rodríguez, and Satta 2011). This set, which we will call **AD_2**, includes ill-nested trees and trees with unbounded gap degree.

MH_k trees. Gómez-Rodríguez, Carroll, and Weir (2011) define a generalization of the tabular algorithm obtained from the shift-reduce parser of Yamada and Matsumoto (2003), or from the arc-hybrid transition system (Gómez-Rodríguez, Carroll, and Weir 2008; Kuhlmann, Gómez-Rodríguez, and Satta 2011). This parser, called MH_k, has items representing a span dominated by several head nodes (hence the acronym, for "multi-headed"). It has complexity $O(n^k)$ and is projective for $k = 3$, but covers increasingly large sets of non-projective trees for values of $k > 3$, which we will call **MH_k** trees.

1-Endpoint-Crossing. Pitler, Kannan, and Marcus (2013) define 1-Endpoint-Crossing trees (**1EC** trees) as dependency trees such that all the arcs that cross a given arc have a common vertex. This set of trees includes trees that are ill-nested and have unbounded gap degree, and can be parsed in $O(n^4)$ (Pitler, Kannan, and Marcus 2013; Pitler 2014).

k-Planar. k-Planar trees (**k-P**, equivalent to k-page book embeddings in graph theory) are those whose non-dummy arcs can be partitioned into k sets (called **planes**), in such a way that arcs belonging to the same plane do not cross (Yli-Jyrä 2003). No globally optimal parser is known for these trees, but they can be handled by a linear-time transition-based parser with k stacks (Gómez-Rodríguez and Nivre 2010, 2013).

k-Crossing Interval. k-Crossing Interval trees (**k-C**) are defined by Pitler and McDonald (2015) with a restriction on intervals formed by crossing arcs. 2-C trees can be parsed accurately with a linear-time shift-reduce parser with two registers (Pitler and McDonald 2015). 2-C trees are a subset of 1EC trees, which in turn are a subset of 2-P trees.

3. Materials and Methods

Corpora. We evaluate the coverage of each class described in Section 2 on HamleDT 2.0 (Rosa et al. 2014), a collection of harmonized versions of existing treebanks of 30 diverse languages, under two different annotations: Prague and Universal Stanford dependencies. Both annotation styles are interesting for parsing: The former tends to be easier to learn for monolingual parsers, but the latter is advantageous in multilingual settings (see Rosa [2015] and references therein). Thus, apart from spanning a variety of languages, these data sets allow us to see the influence of annotation criteria on the coverage of different restrictions on non-projectivity.

Methodology. For the classes of trees that have a known characterization independent of their parsers (i.e., all except AD_2 and MH_k), we determine whether each tree in the treebanks belongs to the class by using scripts that check for the required conditions. In the case of AD_2, we run an implementation of the oracle by Cohen, Gómez-Rodríguez,

and Satta (2012) for the Attardi parser restricted to degree 2, which has been shown to recognize exactly the trees of AD_2 and its implementation checked against the dynamic programming algorithm of Cohen, Gómez-Rodríguez, and Satta (2011). Finally, in the case of MH_k, we run a dynamic programming implementation of the parser itself.

All programs to measure coverage have been extensively tested with examples from the literature, custom-built sets of cases, known relations between classes ($MH_3 = Pr$, 2-C $\subseteq$ 1EPC $\subseteq$ 2-P, $WG_k \subseteq MG_k$, etc.), runs on other treebanks to compare with previously reported coverages, and in some cases, comparison of more than one implementation.

4. Results

The results of the coverage analysis are shown in Tables 1 and 2. For space reasons, we omit some of the classes with less direct practical interest: 1-P (a very mild relaxation of projectivity, of limited interest for expanding coverage), k-C and k-P for $k > 2$ (transition systems for them are possible in theory, but likely impractical due to the extra transitions needed), and those whose best known parser is slower than $O(n^9)$, like WG_k for $k > 2$.

The results provide interesting insights into the coverage of the different classes on a diverse set of corpora with varying amounts of non-projectivity, ranging from the total projectivity of the Prague-style Romanian treebank to the very high non-projectivity in the corpora of classical languages—probably influenced by the presence of poetic texts in them—or in the Stanford-annotated Arabic data set.

Annotation criteria have a large influence on the adequacy of the different restrictions on non-projectivity. The Stanford treebanks not only tend to contain more non-projectivity than the Prague ones, but also more ill-nested trees and trees with higher gap degree. For example, the average proportion of trees that are not in WG_1 is more than double on Stanford than on Prague treebanks, with huge differences in some cases (e.g., 14.84% vs. 0.20% on the Arabic corpora). The same trend appears in the other classes requiring well-nestedness and bounded gap degree. The finding that WG_1 covers almost all phenomena found in treebanks, reported in smaller data sets in the past (Kuhlmann 2010; Gómez-Rodríguez, Carroll, and Weir 2011), is questionable for Stanford treebanks, as it excludes more than 5% of the trees in nine languages.

However, this does not mean that Stanford dependencies are less amenable to mildly non-projective parsing in general, as the Attardi parser and the MH_k parsers for $k > 4$ have better coverage for the Stanford than for the Prague-annotated treebanks. Thus, the lower coverage of the well-nested parsers on the Stanford treebanks does not exclusively owe to them having more non-projectivity in a general sense, but rather different kinds of non-projectivity that are better captured with different restrictions.

Overall, the class with the best coverage among those with known globally optimal parsers running in time $O(n^4)$ is 1EC, which even surpasses WG_1 ($O(n^7)$) on average on the Stanford treebanks. But if we are willing to accept larger complexities, the best tradeoff is achieved with the MH_k parsers. The average coverage is close to 99.5% for MH_5, and practically full for MH_7, only excluding 177 trees out of the more than 800,000 analyzed overall. MH_{12} (not shown in the tables) has full coverage of the 60 treebanks.

The results for the 2-P and 2-C classes, parsable with transition systems, are less surprising, with similar coverage to that reported for smaller sets of treebanks in the respective papers (Gómez-Rodríguez and Nivre 2013; Pitler and McDonald 2015). Note that, although 2-C has notably less coverage than 2-P, its transition system has been

Table 1
Loss of coverage for each of the classes of restricted non-projective trees that have exact parsing algorithms running in $O(n^k)$ for $k <= 7$. The value of k is shown below the name of the class. For each treebank and class, we report the *percentage* of trees that *do not* belong to the class (i.e., lower is better). The best coverage for each complexity bound is highlighted in **boldface**.

Tree-bank	Trees	Pr 3	1EC 4	MH_4 4	M0I 5	M1IS 5	MH_5 5	M1I 6	WG_1S 6	MH_6 6	WG_1 7	MG_1 7	AD_2 7	MH_7 7
Stanford annotation														
ar	7541	72.19	**2.84**	4.36	16.25	14.92	**0.21**	14.84	14.92	**0.08**	14.84	14.64	0.86	**0.040**
bg	13221	17.56	**0.51**	1.28	1.23	1.03	**0.05**	1.03	1.03	**0.01**	1.03	1.01	0.25	**0.008**
bn	1129	7.00	**0.18**	0.44	0.89	0.27	**0.00**	0.27	0.27	**0.00**	0.27	0.09	0.27	**0.000**
ca	14924	23.69	**0.97**	2.61	2.37	2.29	**0.02**	2.28	2.29	**0.00**	2.28	2.12	0.31	**0.000**
cs	87913	26.28	**2.22**	3.00	3.37	2.71	**0.17**	2.65	2.71	**0.01**	2.65	2.32	1.00	**0.002**
da	5512	29.75	**2.99**	5.59	5.77	3.37	**0.65**	3.32	3.37	**0.11**	3.32	2.54	3.18	**0.018**
de	38020	36.21	**5.13**	5.77	9.19	6.43	**0.61**	5.57	6.43	**0.08**	5.57	4.44	3.42	**0.011**
el	2902	34.36	**3.48**	4.20	5.17	3.86	**0.10**	3.48	3.86	**0.00**	3.48	3.14	0.90	**0.000**
en	18791	23.18	**1.24**	2.80	3.48	3.24	**0.27**	3.23	3.24	**0.02**	3.23	2.72	0.94	**0.011**
es	15984	24.46	**1.33**	2.24	1.71	1.69	**0.01**	1.69	1.69	**0.00**	1.69	1.51	0.30	**0.000**
et	1315	2.13	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.000**
eu	11225	25.58	2.69	**2.17**	7.34	6.25	**0.23**	6.17	6.25	**0.03**	6.17	0.87	1.28	**0.000**
fa	12455	23.85	**2.38**	2.72	4.34	2.40	**0.49**	2.18	2.40	**0.16**	2.18	1.46	1.68	**0.048**
fi	4307	14.95	**1.49**	2.02	2.28	1.93	**0.21**	1.93	1.93	**0.12**	1.93	1.83	1.28	**0.093**
grc	21173	67.84	30.64	**26.55**	31.81	27.42	**3.64**	26.21	27.40	**0.23**	26.19	12.44	8.45	**0.009**
hi	13274	23.11	3.36	**3.33**	5.58	2.92	**0.62**	2.76	2.92	**0.17**	2.76	1.36	2.66	**0.045**
hu	6424	31.97	**5.74**	10.13	12.16	7.22	**1.90**	7.19	7.22	**0.30**	7.19	6.74	7.38	**0.078**
it	3359	31.08	3.16	**1.67**	2.89	3.01	**0.18**	2.14	3.01	**0.00**	2.14	1.76	0.83	**0.000**
ja	17753	29.44	5.04	**2.43**	5.05	3.68	**0.25**	3.59	3.66	**0.03**	3.57	1.04	1.17	**0.011**
la	3473	50.13	15.98	**15.78**	17.48	11.49	**2.74**	10.48	11.46	**0.32**	10.45	6.82	8.52	**0.029**
nl	13735	43.25	**8.24**	9.07	8.85	6.50	**1.01**	5.03	6.50	**0.17**	5.03	4.54	5.81	**0.058**
pt	9359	30.11	**2.09**	6.03	7.84	6.32	**0.42**	6.21	6.32	**0.04**	6.21	6.09	2.29	**0.011**
ro	4042	3.66	**0.00**	0.05	0.05	0.05	**0.00**	0.05	0.05	**0.00**	0.05	0.05	0.03	**0.000**
ru	34895	18.34	**0.86**	1.02	0.97	0.55	**0.03**	0.52	0.54	**0.00**	0.52	0.34	0.46	**0.000**
sk	57408	23.17	**2.31**	2.78	3.44	2.87	**0.23**	2.71	2.87	**0.02**	2.70	2.48	1.00	**0.005**
sl	1936	27.27	**3.98**	4.18	4.34	3.20	**0.57**	3.05	3.20	**0.05**	3.05	2.74	2.22	**0.000**
sv	11431	21.25	**2.07**	2.19	3.17	2.09	**0.44**	2.00	2.07	**0.18**	1.97	1.11	1.26	**0.061**
ta	600	3.33	**0.00**	**0.00**	0.83	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.000**
te	1450	1.79	**0.00**	**0.00**	0.14	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.000**
tr	5935	24.72	3.30	**3.17**	8.91	6.87	**0.73**	6.82	6.87	**0.24**	6.82	5.81	1.90	**0.051**
Macro avg		26.39	**3.81**	4.25	5.90	4.49	**0.53**	4.25	4.48	**0.08**	4.24	3.07	1.99	**0.020**
Prague annotation														
ar	7541	11.50	**0.29**	0.62	3.29	0.21	**0.08**	0.20	0.21	**0.01**	0.20	0.16	0.54	**0.000**
bg	13221	11.10	**0.17**	4.32	4.33	**0.16**	0.17	0.16	0.16	**0.08**	0.16	0.16	4.30	**0.015**
bn	1129	5.93	**0.18**	0.35	0.97	0.18	**0.00**	0.18	0.18	**0.00**	0.18	0.00	0.27	**0.000**
ca	14924	5.87	**0.02**	0.47	0.22	0.09	0.03	0.09	0.09	**0.01**	0.09	0.09	0.45	**0.000**
cs	87913	23.61	**1.33**	1.59	2.82	0.74	**0.10**	0.59	0.74	**0.01**	0.59	0.49	0.96	**0.003**
da	5512	15.62	**0.71**	3.05	3.83	**0.35**	0.93	0.33	0.35	**0.31**	0.33	0.22	2.85	**0.109**
de	38020	37.01	**5.48**	6.10	10.65	5.74	**0.67**	4.77	5.74	**0.10**	4.77	3.96	4.04	**0.024**
el	2902	21.57	**1.34**	2.14	5.31	1.00	**0.14**	0.79	1.00	**0.03**	0.79	0.79	1.55	**0.000**
en	18791	6.38	**0.71**	1.01	1.20	0.63	**0.15**	0.62	0.63	**0.03**	0.62	0.08	0.95	**0.011**
es	15984	7.31	**0.02**	0.34	0.69	0.16	0.03	0.16	0.16	**0.00**	0.16	0.16	0.26	**0.000**
et	1315	0.84	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.000**
eu	11225	17.53	**1.61**	1.97	3.71	0.86	**0.29**	0.72	0.86	**0.05**	0.72	0.45	1.39	**0.009**
fa	12455	20.72	**2.39**	3.01	6.79	1.46	**0.72**	1.30	1.46	**0.30**	1.30	0.68	2.51	**0.120**
fi	4307	11.82	**0.58**	0.79	1.16	0.16	**0.14**	**0.12**	0.16	**0.12**	0.12	0.12	0.63	**0.070**
grc	21173	77.44	**30.97**	31.61	30.86	17.80	**4.42**	10.22	17.80	**0.28**	10.22	8.65	10.67	**0.019**
hi	13274	29.95	**2.52**	5.94	7.70	1.82	**1.79**	1.73	1.81	**0.49**	1.72	1.21	5.14	**0.196**
hu	6424	27.41	**4.39**	9.17	11.40	5.98	**1.73**	5.95	5.98	**0.23**	5.95	5.62	7.22	**0.125**
it	3359	8.16	**0.45**	1.07	2.50	0.54	**0.09**	0.54	0.54	**0.06**	0.54	0.36	0.92	**0.060**
ja	17753	5.29	1.43	**0.45**	4.05	0.57	**0.04**	0.57	0.57	**0.00**	0.57	0.57	0.19	**0.000**
la	3473	50.45	15.06	**14.77**	21.16	11.09	**2.45**	9.56	11.03	**0.32**	9.50	5.82	10.28	**0.086**
nl	13735	35.75	**4.27**	9.55	9.54	4.40	**1.51**	3.44	4.40	**0.24**	3.44	3.36	7.82	**0.066**
pt	9359	19.95	**0.67**	4.57	7.01	4.77	**0.49**	4.73	4.77	**0.10**	4.73	4.68	2.56	**0.032**
ro	4042	0.00	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.000**
ru	34895	9.61	**0.29**	0.70	1.14	0.21	**0.04**	0.20	0.21	**0.00**	0.20	0.10	0.54	**0.000**
sk	57408	17.84	**0.97**	1.32	2.92	0.77	**0.12**	0.68	0.77	**0.02**	0.68	0.56	0.87	**0.007**
sl	1936	20.87	**1.14**	1.96	3.41	0.88	**0.26**	0.78	0.88	**0.00**	0.78	0.67	1.65	**0.000**
sv	11431	11.03	**1.44**	1.55	3.26	1.02	**0.35**	0.98	1.01	**0.14**	0.97	0.50	1.11	**0.061**
ta	600	2.00	**0.00**	**0.00**	1.00	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.000**
te	1450	0.83	**0.00**	**0.00**	0.14	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.000**
tr	5935	31.95	**6.86**	8.16	16.18	9.08	**2.21**	9.00	9.00	**0.54**	8.91	8.69	4.72	**0.152**
Macro avg		18.18	**2.84**	3.89	5.57	2.35	**0.63**	1.95	2.35	**0.12**	1.94	1.60	2.48	**0.039**

Table 2
Loss of coverage for classes of restricted non-projective trees not shown in Table 1. For classes that have exact parsing algorithms running in $O(n^k)$, the value of k is shown below the name of the class. For each treebank and class, we report the *percentage* of trees that *do not* belong to the class (i.e., lower is better). The best coverage for each complexity bound is shown in **boldface**.

Treebank	MH_8	WG_2	MH_9	2-P	2-C	Treebank	MH_8	WG_2	MH_9	2-P	2-C
Stanford ann.	8	9	9	n/a	n/a	*Prague ann.*	8	9	9	n/a	n/a
ar (Arabic)	**0.027**	3.819	**0.027**	0.278	31.044	ar	**0.000**	0.066	**0.000**	0.013	0.756
bg (Bulgarian)	**0.008**	0.098	**0.000**	0.045	1.482	bg	**0.008**	**0.000**	**0.000**	0.015	0.613
bn (Bengali)	**0.000**	0.177	**0.000**	0.000	0.266	bn	**0.000**	0.177	**0.000**	0.000	0.354
ca (Catalan)	**0.000**	0.281	**0.000**	0.067	2.627	ca	**0.000**	0.013	**0.000**	0.000	0.181
cs (Czech)	**0.001**	0.543	**0.001**	0.415	4.094	cs	**0.002**	0.131	**0.002**	0.099	2.894
da (Danish)	**0.000**	0.998	**0.000**	0.744	5.443	da	**0.036**	0.127	**0.000**	0.018	1.361
de (German)	**0.008**	2.078	**0.003**	1.231	8.951	de	**0.000**	1.228	**0.000**	1.021	9.771
el (Mod. Greek)	**0.000**	0.965	**0.000**	0.379	7.271	el	**0.000**	0.103	**0.000**	0.172	2.757
en (English)	**0.005**	0.841	**0.005**	0.644	2.522	en	**0.011**	0.596	**0.005**	0.553	0.798
es (Spanish)	**0.000**	0.282	**0.000**	0.069	2.803	es	**0.000**	0.025	**0.000**	0.000	0.119
et (Estonian)	**0.000**	**0.000**	**0.000**	0.000	0.000	et	**0.000**	**0.000**	**0.000**	0.000	0.000
eu (Basque)	**0.000**	5.826	**0.000**	0.249	4.686	eu	**0.000**	0.374	**0.000**	0.143	2.601
fa (Persian)	**0.024**	1.124	**0.008**	0.385	4.906	fa	**0.064**	0.923	**0.032**	0.225	3.605
fi (Finnish)	**0.023**	0.395	**0.000**	0.859	1.788	fi	**0.023**	0.023	**0.000**	0.070	0.882
grc (Anc. Greek)	**0.005**	20.247	**0.000**	6.282	36.565	grc	**0.000**	2.829	**0.000**	3.495	39.347
hi (Hindi)	**0.015**	1.944	**0.000**	0.203	4.821	hi	**0.045**	0.768	**0.023**	0.105	5.176
hu (Hungarian)	**0.000**	2.086	**0.000**	0.794	8.266	hu	**0.016**	1.666	**0.016**	0.607	5.791
it (Italian)	**0.000**	0.804	**0.000**	0.298	6.490	it	**0.000**	0.268	**0.000**	0.060	0.744
ja (Japanese)	**0.000**	3.002	**0.000**	0.107	6.095	ja	**0.000**	0.101	**0.000**	0.000	1.594
la (Latin)	**0.000**	5.845	**0.000**	3.974	21.250	la	**0.000**	5.644	**0.000**	3.858	21.653
nl (Dutch)	**0.015**	1.172	**0.007**	0.961	14.503	nl	**0.007**	0.197	**0.000**	1.201	9.661
pt (Portuguese)	**0.000**	1.036	**0.000**	0.224	5.204	pt	**0.032**	0.801	**0.011**	0.107	1.667
ro (Romanian)	**0.000**	**0.000**	**0.000**	0.000	0.049	ro	**0.000**	**0.000**	**0.000**	0.000	0.000
ru (Russian)	**0.000**	0.201	**0.000**	0.077	1.656	ru	**0.000**	0.100	**0.000**	0.026	0.516
sk (Slovak)	**0.002**	0.597	**0.000**	0.493	4.053	sk	**0.002**	0.172	**0.000**	0.099	2.045
sl (Slovenian)	**0.000**	0.723	**0.000**	1.188	6.095	sl	**0.000**	0.155	**0.000**	0.052	2.583
sv (Swedish)	**0.026**	1.233	**0.009**	0.744	3.351	sv	**0.026**	0.656	**0.026**	0.542	2.021
ta (Tamil)	**0.000**	**0.000**	**0.000**	0.000	0.167	ta	**0.000**	**0.000**	**0.000**	0.000	0.000
te (Telugu)	**0.000**	**0.000**	**0.000**	0.000	0.000	te	**0.000**	**0.000**	**0.000**	0.000	0.000
tr (Turkish)	**0.017**	4.482	**0.017**	0.253	5.173	tr	**0.034**	3.370	**0.000**	0.084	10.025
Macro average	**0.006**	2.027	**0.003**	0.699	6.721	Macro avg.	**0.010**	0.684	**0.004**	0.419	4.317

shown to have very good empirical accuracy, probably because it is an easier to learn model.

5. Discussion

We have measured the coverage of a wide range of classes of mildly non-projective dependency trees on a large collection of treebanks with two different annotation styles, providing valuable data to compare said classes in terms of balance between coverage and efficiency. The relative coverage of the different classes varies across languages and annotation criteria. Explaining the concrete factors affecting it for each individual language is outside the scope of this work, and an interesting subject for studies focused on particular languages and corpora. However, despite this variability, there are very clear trends in the results. A relevant one is that the best general tradeoff is achieved by 1-Endpoint-Crossing trees (for complexity $O(n^4)$) and MH_k trees (for larger polynomial complexities).

Although we have focused on the coverage-efficiency tradeoff, there are other aspects of mildly non-projective classes that one may wish to take into account, like their relation to constituency grammar formalisms (Kuhlmann 2010) or characterizability

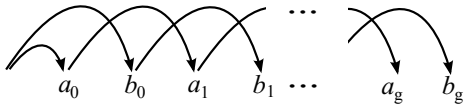

Figure 1
An ill-nested dependency tree with gap degree g that is in MH_4. It can be parsed by the MH_4 parser starting by building the item $[b_{g-1}, a_g, b_g, b_g + 1]$ and then proceeding from right to left.

(Pitler and McDonald 2015). In this sense, it is worth noting that no characterization independent of the parsing algorithm itself is known for the MH_k classes, for $k > 3$, just as happens with Attardi trees. In fact, MH_k trees have been very little studied, and their empirical coverage was unknown prior to this work. Because it is notably high, finding a simple characterization of MH_k trees is an interesting open problem, which may be solvable as MH_k trees have some desirable formal properties that AD_2 trees lack, like left-right symmetry (reversing the order of the words of an MH_k tree produces an MH_k tree).

Two novel observations about the relation of MH_k trees to other classes of trees are that: (1) MH_4 contains ill-nested trees with unbounded gap degree (and therefore, the same can be said of MH_k for $k > 4$, as $MH_{k-1} \subseteq MH_k$ for all k); and (2) $MH_4 \subseteq$ 2-P.

Observation (1) can be shown by example, with the tree in Figure 1, and an outline of the proof for (2) follows: given the MH_4 parser (shown in Figure 2), we build a variant MH_4^{2P} that associates each arc with a plane $\in \{P_0, P_1\}$, satisfying the 2-P constraint. To do so, we annotate each index (node) on items with a forbidden plane, such that steps creating an arc $h \rightarrow d$ always do so on a plane not forbidden for h or d. If both planes are allowed, then if the item has a node x between h and d with a forbidden plane, the arc is created on that plane (to avoid forbidding both planes on x at the same time), otherwise an arbitrary plane is chosen. Initial items do not have any restrictions, but when we create a right arc $A = h_1 \rightarrow h_3$ with a *Link* step $[h_1, h_2, h_3, h_4] \vdash [h_1, h_2, h_4]$ on plane P_i, we forbid P_i on node h_2 (located between h_1 and h_3), which prevents arcs that cross A from being created on the same plane as A, and the symmetric is done for left arcs. Annotations are propagated across deductions together with their nodes.

Parsing a tree T with MH_4^{2P} always produces a valid partition of T into planes, that is, it never reaches a situation where an arc cannot be created without violating 2-planarity because both planes are forbidden by the restrictions. This is shown by proving that each item has at most one node with a forbidden plane. To see this, note that the first and last nodes of an item cannot have any forbidden plane: by construction, restrictions always originate on the central node of a 3-node item, and no steps in the parser can move a node in the middle to the first or last position. Restrictions can

Item form: $[h_1, h_2, \ldots, h_m]$, $0 \leq h_1 < h_2 < \ldots < h_m \leq n+1$; $2 \leq m \leq 4$

Goal: $[0, n+1]$ **Axioms:** $[i, i+1]$, $0 \leq i \leq n$

Deduction steps: $\dfrac{[h_1, h_2, \ldots, h_m][h_m, h_{m+1}, \ldots, h_p]}{[h_1, h_2, \ldots, h_p]}$ (*Combine*), $p \leq 4$

$\dfrac{[h_1, h_2, \ldots, h_m]}{[h_1, h_2, \ldots, h_{j-1}, h_{j+1}, \ldots, h_m]}$ (*Link*; $h_i \rightarrow h_j$), $1 < j < m, 1 \leq i \leq m, j \neq i$

Figure 2
Deduction system for the MH_4 parser for a string of length n.

propagate to 4-node items, but these always come from applying a *Combine* step on a 3- and a 2-node item, so again at most one node (the central one in the 3-node item) can have a forbidden plane. Thus, we can always associate a plane to an arc without violating restrictions, as there can be only one restriction per item and therefore at least one plane is allowed.

Note that this proof implicitly relies on the fact that, when the MH_4 parser creates an arc A, any subsequently built arcs crossing A must share an endpoint: for example, after the arc $A = h \rightarrow d$ is created by a deduction step $[h, x, d, y] \vdash [h, x, y]$, the only endpoint located between h and d remaining available is x, so any subsequent arcs crossing A must be incident to x. This restriction is interestingly similar to the definition of 1EC trees, although weaker because it only affects arcs created after A.

The relation of MH_4 with the 2-P class, as well as its indirect relation with 1EC, may help obtain a characterization for the set of MH_k trees. Their good balance between coverage and parsing efficiency makes this class, together with 1EC trees, very interesting for modeling the non-projectivity found in natural languages.

Acknowledgments

The author is partially funded by the TELEPARES-UDC project from MINECO (FFI2014-51978-C2-2-R) and an Oportunius program grant from Xunta de Galicia.

References

Attardi, Giuseppe. 2006. Experiments with a multilanguage non-projective dependency parser. In *Proceedings of CoNLL*, pages 166–170, New York.

Ballesteros, Miguel and Joakim Nivre. 2013. Going to the roots of dependency parsing. *Computational Linguistics*, 39(1):5–13.

Bodirsky, Manuel, Marco Kuhlmann, and Mathias Möhl. 2005. Well-nested drawings as models of syntactic structure. In *Proceedings of FG-MoL*, pages 195–203, Edinburgh.

Cohen, Shay B., Carlos Gómez-Rodríguez, and Giorgio Satta. 2011. Exact inference for generative probabilistic non-projective dependency parsing. In *Proceedings of EMNLP*, pages 1234–1245, Edinburgh.

Cohen, Shay B., Carlos Gómez-Rodríguez, and Giorgio Satta. 2012. Elimination of spurious ambiguity in transition-based dependency parsing. CoRR,abs/1206.6735.

Eisner, Jason. 1996. Three new probabilistic models for dependency parsing: An exploration. In *Proceedings of COLING*, pages 340–345, San Francisco, CA.

Gómez-Rodríguez, Carlos, John Carroll, and David Weir. 2008. A deductive approach to dependency parsing. In *Proceedings of ACL-HLT*, pages 968–976, Columbus, OH.

Gómez-Rodríguez, Carlos, John A. Carroll, and David J. Weir. 2011. Dependency parsing schemata and mildly non-projective dependency parsing. *Computational Linguistics*, 37(3):541–586.

Gómez-Rodríguez, Carlos and Joakim Nivre. 2010. A transition-based parser for 2-planar dependency structures. In *Proceedings of ACL*, pages 1492–1501, Uppsala.

Gómez-Rodríguez, Carlos and Joakim Nivre. 2013. Divisible transition systems and multiplanar dependency parsing. *Computational Linguistics*, 39(4):799–845.

Gómez-Rodríguez, Carlos, David Weir, and John Carroll. 2009. Parsing mildly non-projective dependency structures. In *Proceedings of EACL*, pages 291–299, Athens.

Havelka, Jiří. 2007. Beyond projectivity: Multilingual evaluation of constraints and measures on non-projective structures. In *Proceedings of ACL*, pages 608–615, Prague.

Kuhlmann, Marco. 2010. *Dependency Structures and Lexicalized Grammars. An Algebraic Approach*, volume 6270 of *Lecture Notes in Computer Science*. Springer.

Kuhlmann, Marco, Carlos Gómez-Rodríguez, and Giorgio Satta. 2011. Dynamic programming algorithms for transition-based dependency parsers. In *Proceedings of ACL*, pages 673–682, Portland, OR.

McDonald, Ryan and Giorgio Satta. 2007. On the complexity of non-projective data-driven dependency parsing. In *Proceedings of IWPT*, pages 121–132, Prague.

Nivre, Joakim. 2006. *Inductive Dependency Parsing*. Springer.

Nivre, Joakim. 2008. Algorithms for Deterministic Incremental Dependency Parsing. *Computational Linguistics*, 34(4):513–553.

Nivre, Joakim. 2009. Non-projective dependency parsing in expected linear time. In *Proceedings of ACL*, pages 351–359, Singapore.

Pitler, Emily. 2014. A crossing-sensitive third-order factorization for dependency parsing. *Transactions of the Association for Computational Linguistics*, 2:41–54.

Pitler, Emily, Sampath Kannan, and Mitchell Marcus. 2012. Dynamic programming for higher order parsing of gap-minding trees. In *Proceedings of EMNLP-CoNLL*, pages 478–488, Jeju Island.

Pitler, Emily, Sampath Kannan, and Mitchell Marcus. 2013. Finding optimal 1-endpoint-crossing trees. *Transactions of the ACL*, 1:13–24.

Pitler, Emily and Ryan McDonald. 2015. A linear-time transition system for crossing interval trees. In *Proceedings of NAACL-HLT*, pages 662–671, Denver, CO.

Rosa, Rudolf. 2015. Multi-source cross-lingual delexicalized parser transfer: Prague or Stanford? In *Proceedings of DepLing*, pages 281–290, Uppsala.

Rosa, Rudolf, Jan Mašek, David Mareček, Martin Popel, Daniel Zeman, and Zdeněk Žabokrtský. 2014. HamleDT 2.0: Thirty dependency treebanks stanfordized. In *Proceedings of LREC*, pages 2334–2341, Reykjavik.

Satta, Giorgio and Marco Kuhlmann. 2013. Efficient parsing for head-split dependency trees. *Transactions of the ACL*, 1:267–278.

Yamada, Hiroyasu and Yuji Matsumoto. 2003. Statistical dependency analysis with support vector machines. In *Proceedings of IWPT*, pages 195–206, Nara.

Yli-Jyrä, Anssi Mikael. 2003. Multiplanarity — a model for dependency structures in treebanks. In *Proceedings of TLT*, pages 189–200, Växjö.