Coarse-To-Fine Parsing for Expressive Grammar Formalisms

Christoph Teichmann and **Alexander Koller**
Saarland University, Saarbrücken
{cteichmann|koller}@coli.uni-saarland.de

Jonas Groschwitz
Macquarie University, Sydney
jonas.groschwitz@students.mq.edu.au

Abstract

We generalize coarse-to-fine parsing to grammar formalisms that are more expressive than PCFGs and/or describe languages of trees or graphs. We evaluate our algorithm on PCFG, PTAG, and graph parsing. While we achieve the expected performance gains on PCFGs, coarse-to-fine does not help for PTAG and can even slow down parsing for graphs. We discuss the implications of this finding.

1 Introduction

Coarse-to-fine (CTF) parsing (Charniak et al., 2006) is one of the most effective pruning techniques for PCFG parsing. Its basic idea is to simplify a grammar by systematically merging nonterminals together. One then parses the input with the simple grammar, and uses statistics calculated from the resulting parse chart to constrain parsing with the original grammar. This can speed up parsing by an order of magnitude at no loss in accuracy (Charniak et al., 2006; Petrov and Klein, 2007).

We present an algorithm for CTF parsing for grammar formalisms that are more expressive than PCFGs – to our knowledge, for the first time. More precisely, we extend CTF parsing to Interpreted Regular Tree Grammars (IRTGs, Koller and Kuhlmann (2011)), a very general grammar formalism which captures PCFGs, tree-adjoining grammars (TAGs, Joshi and Schabes (1997)), hyperedge replacement graph grammars (HRGs, Chiang et al. (2013)), and many others. Our direct application of CTF to expressive grammar formalism contrasts with related work (van Cranenburgh, 2012; Zhang and Krieger, 2011) which limits entries for the parse chart of the expressive formalism using the parse chart of a PCFG approximation.

We then evaluate our generalized CTF algorithm on a number of grammar formalisms. On PCFGs, we obtain the expected speedups. However, we observe no speedups for TAG parsing compared to unpruned parsing, and HRG parsing on abstract meaning representations (Banarescu et al., 2013) is even slowed down by CTF. We discuss these findings and show how the efficacy of CTF parsing relies on specific properties of PCFG grammars derived from treebanks. Because these properties do not depend on the specifics of our formalism, they would generalize to formalism specific implementations of CTF for TAG or HRG.

2 Interpreted Regular Tree Grammars

Interpreted Regular Tree Grammars (IRTGs, Koller and Kuhlmann (2011)) generalize a wide range of grammar formalisms, including (probabilistic) context-free grammars (PCFGs), tree-adjoining grammars (Joshi and Schabes, 1997; Koller and Kuhlmann, 2012), hyperedge replacement grammars (Chiang et al., 2013; Groschwitz et al., 2015), as well as synchronous and transducer versions of these formalisms. They achieve this by distinguishing carefully between the generation of grammatical derivation trees and the way in which these derivation trees are interpreted as values of some algebra.

Formally, a (monolingual) IRTG $\mathcal{G} = (G, h, \mathcal{A})$ consists of a weighted regular tree grammar (RTG, (Comon et al., 2007)) G over some signature Σ of node labels, an algebra $\mathcal{A}$ over some signature Δ into which the derivation trees are interpreted, and a tree homomorphism $h : T_\Sigma \to T_\Delta$ that maps derivation trees into terms over the algebra. The RTG G generates a language $L(G) \subseteq T_\Sigma$ of *derivation trees*. Based on these, the language of the IRTG, $L(\mathcal{G}) = \{[\![h(t)]\!]_\mathcal{A} \mid t \in L(G)\}$, is obtained by mapping each derivation tree $t \in L(\mathcal{G})$

Proceedings of the 15th International Conference on Parsing Technologies, pages 122–127,
Pisa, Italy; September 20–22, 2017.

into a term $h(t)$ over the algebra, and then evaluating this term in the algebra.

Fig. 1 shows a simple IRTG. The RTG G is shown on the left; it derives, among others, the derivation tree $t = r_1(r_4, r_2(r_6, r_5)) \in L(G)$. The tree homomorphism shown on the right maps this to the term $h(t) = *(\mathsf{john}, *(\mathsf{loves}, \mathsf{mary}))$. By evaluating this term over a string algebra $\mathcal{A}$ which interprets the symbol $*$ as string concatenation, we obtain "John loves Mary" $\in L(\mathcal{G})$. For simplicity we will identify rules of G with their labels, i.e. we simply write r_1 for the first rule in Fig. 1.

IRTGs can capture different grammar formalisms by varying the algebra into which derivation trees are interpreted. For instance, TAG requires a string algebra with a string wrapping operation (Koller and Kuhlmann, 2012). It is also possible to extend the IRTG formalism in order to allow for multiple homomorphisms and algebras, which is useful for mapping between inputs and outputs and can be used e.g. in semantic parsing (Koller, 2015).

Grammars from different formalisms also tend to vary in the complexity of the homomorphism h. For instance, all binary rules of a PCFG in CNF map to simple concatenation (cf. r_1, r_2 in Fig. 1). By contrast, IRTG encodings of TAG grammars can use h to associate entire elementary trees with a single rule.

Parsing for IRTGs proceeds in three steps. First, given an input object $w \in \mathcal{A}$, we compute a *decomposition grammar* D_w which generates all terms over $\mathcal{A}$ that evaluate to w. Then we compute the *inverse homomorphism (invhom)* of D_w, i.e. an RTG I_w with $L(I_w) = \{t \in T_\Sigma \mid h(t) \in L(D_w)\}$. This RTG thus describes all derivation trees that are interpreted to w. Finally, we intersect I_w with G - the RTG from $\mathcal{G}$, obtaining an RTG M – the *parse chart* – which compactly describes the grammatically correct derivation trees. Similar to intersection constructions for e.g. finite state automata, the nonterminals of M are pairs AJ of nonterminals A of G and J of I_w and M has rules of the form $A_0J_0 \to r(A_1J_1, \ldots, A_nJ_n)$.

When the rules of G are assigned weights, as in Fig. 1, we can use the Viterbi algorithm to extract the highest-weight derivation tree from M. We can also compute inside and outside weights $\mathsf{in}(AJ)$ and $\mathsf{out}(AJ)$ for every nonterminal in M as usual.

$\mathrm{S} \to r_1(\mathrm{NP}, \mathrm{VP})$	[1]	$h(r_1) = *(x_1, x_2)$
$\mathrm{VP} \to r_2(\mathrm{VP}, \mathrm{NP})$	[0.5]	$h(r_2) = *(x_1, x_2)$
$\mathrm{VP} \to r_3(\mathrm{VP}, \mathrm{NP})$	[0.1]	$h(r_3) = *(\mathsf{to}, *(x_1, x_2))$
$\mathrm{NP} \to r_4$	[0.5]	$h(r_4) = \mathsf{john}$
$\mathrm{NP} \to r_5$	[0.5]	$h(r_5) = \mathsf{mary}$
$\mathrm{VP} \to r_6$	[0.4]	$h(r_6) = \mathsf{loves}$

Figure 1: An example IRTG.

3 Coarse-to-fine parsing for IRTGs

Coarse-to-fine parsing for PCFGs. In PCFG parsing, CTF parsing is an established pruning technique for computing the best parse tree of a sentence w given a PCFG G_F. We assume a *fine-to-coarse map* C, which maps the nonterminal symbols of G_F into a set of coarse-grained nonterminal symbols, potentially making two nonterminals of G_F the same. By merging rules of G_F that now have the same nonterminals on the left and right hand side, we obtain a smaller PCFG G_C (the *coarse* grammar). For instance, if we have $C(\mathrm{S}) = C(\mathrm{NP}) = C(\mathrm{VP}) = \mathrm{HP}$, then the rules $\mathrm{S} \to \mathrm{NP}\ \mathrm{VP}$ and $\mathrm{VP} \to \mathrm{VP}\ \mathrm{NP}$ are both mapped to the same rule, $\mathrm{HP} \to \mathrm{HP}\ \mathrm{HP}$. The fine-to-coarse mapping may have multiple *levels*, providing increasingly coarse-grained grammars.

CTF parsing then proceeds by parsing w with respect to G_C and computing the inside and outside probabilities of all edges in the (coarse) parse chart. Edges whose probabilities are too low are pruned away. The others are refined into edges for a parse chart with respect to G_F. Thus if an edge $\mathrm{HP}[2{-}7] \to \mathrm{HP}[2{-}3]\ \mathrm{HP}[3{-}7]$ in the coarse chart (describing a split of the substring from 2 to 7 at position 3) is sufficiently likely, it will be refined into both $\mathrm{S}[2{-}7] \to \mathrm{NP}[2{-}3]\ \mathrm{VP}[3{-}7]$ and $\mathrm{VP}[2{-}7] \to \mathrm{VP}[2{-}3]\ \mathrm{NP}[3{-}7]$. The Viterbi parse tree of this fine chart will then be a parse tree of w with respect to G_F.

Coarsification for IRTGs. We generalize this procedure to IRTGs. In doing this, we need to pay special attention to the fact that the rules of an IRTG may differ not only in their nonterminal symbols, but also in their homomorphic interpretations. As mentioned above, this is prevalent in expressive grammar formalisms, such as TAG.

We define two rules $A_0 \to r_1(A_1, \ldots, A_n)$ and $B_0 \to r_2(B_1, \ldots, B_n)$ of an IRTG to be *equivalent* with respect to a fine-to-coarse map C iff $C(A_i) = C(B_i)$ for all i and $h(r_1) = h(r_2)$, i.e. both rules are interpreted in the same way by the

input homomorphism. Using the mapping C from above, we find that r_1 and r_2 in Fig. 1 are equivalent to each other, but not to r_3.

We can then partition the rules of a fine-grained IRTG $\mathcal{G}_F$ into their equivalence classes, and build a *coarse-grained* IRTG $\mathcal{G}_C$ over the same algebra with one rule for each equivalence class. Let $R = \{r_1, \ldots, r_k\}$ be an equivalence class containing the fine-grained rule $A_0 \to r_1(A_1, \ldots, A_n)$. Then $\mathcal{G}_C$ will contain the rule $H_0 \to R(H_1, \ldots, H_n)$, where $H_i = C(A_i)$ for all i, and $h_C(R) = h_F(r_1)$. The choice of r_1 among the elements of R does not matter, because equivalent rules have the same homomorphic image and map to the same coarse-grained nonterminals. In this paper, we will simply let the weight of the coarse rule be the sum of the weights of the fine rules in order to set the inside score of an item in a coarse chart to approximately the sum of the weights of the finer items it represents; if suitable data is available, these weights could also be re-estimated from a treebank (Charniak et al., 2006; Petrov and Klein, 2007). In the example, we obtain $\mathrm{HP} \to R_1(\mathrm{HP}, \mathrm{HP})$ and $\mathrm{HP} \to R_2(\mathrm{HP}, \mathrm{HP})$ with $R_1 = \{r_1, r_2\}$ and $R_2 = \{r_3\}$. This construction generalizes easily to multiple CTF levels.

Coarse-to-fine parsing with IRTGs. Given this precomputation, we can now perform coarse-to-fine parsing. Given an input object w, we first compute a complete parse chart M_C using the coarse-grained IRTG $\mathcal{G}_C$, e.g. using one of the parsing algorithms of Groschwitz et al. (2016). The entries e of this chart are rules $H_0 J_0 \to R(H_1 J_1, \ldots, H_n J_n)$, such that $H_0 \to R(H_1, \ldots, H_n)$ is a rule of $\mathcal{G}_C$ and the invhom grammar I_w contains a rule $J_0 \to r(J_1, \ldots, J_n)$ for one, and thus all, $r \in R$.

We compute $\mathsf{in}(AJ)$ and $\mathsf{out}(AJ)$ for every nonterminal AJ of M_C, and use them to calculate a score

$$s(e) = \mathsf{out}(H_0 J_0) \cdot w(R) \cdot \mathsf{in}(H_1 J_1) \cdot \ldots \cdot \mathsf{in}(H_n J_n)$$

for each chart entry e, where $w(R)$ is the rule weight in $\mathcal{G}_C$. We let $Z = \mathsf{in}(HJ)$ be the total inside weight of the start nonterminal of M_C, which combines the start nonterminal H of $\mathcal{G}_C$ and the start nonterminal J of I_w.

Then we refine the coarse-grained chart M_C into a fine-grained chart M_F. If $s(e) < \theta \cdot Z$ for some fixed threshold θ, we discard e. Otherwise, we add an entry $A_0 J_0 \to r(A_1 J_1, \ldots, A_n J_n)$ to M_F for each rule $\mathrm{A}_0 \to r(\mathrm{A}_1, \ldots, \mathrm{A}_1)$ in the fine-grained IRTG $\mathcal{G}_F$ with $r \in R$.

If we have k coarse-to-fine levels ($k-1$ = coarsest, 0 = finest), we repeat this refinement step $k-1$ times to obtain a chart for the original IRTG and then find the best derivation using Viterbi decoding.

4 Evaluation

Using this algorithm, we can do CTF parsing for all grammar formalisms that can be encoded as IRTGs. We evaluate it on PCFG, TAG, and graph parsing, using the efficient algorithms of Groschwitz et al. (2016) to compute the coarsest charts. These algorithms are lazy and try to avoid computing rules of the inverse homomorphism grammar which cannot participate in a derivation. This means that the number of rules in the inverse automaton differs depending on the grammar with which we are parsing. The evaluation grammars and fine-to-coarse mappings are available as supplementary material for this paper, and our coarse-to-fine parser is implemented as part of the Alto Toolkit.[1]

PCFG evaluation. First, we reproduce the known result that CTF parsing speeds up PCFG parsing. We read off a PCFG from the parse trees of the WSJ portion of the Penn Treebank (Sections 02–21), using the gold POS tags as terminal symbols; binarize it with the "inside" binarization strategy of Klein and Manning (2003); and convert it to an IRTG. This yields an IRTG grammar with 23817 rules and 8202 nonterminals, of which 8131 were created during binarization. We then parsed the sentences in Section 23 of up to 40 words, both without pruning and with the CTF parser (longer sentences were infeasible with the unpruned parser). For CTF we used the four-level fine-to-coarse mapping from Charniak et al. (2006) and a threshold of $\theta = 10^{-5}$. We also apply the fine-to-coarse mapping to the nonterminals introduced during binarization. If the nonterminal 'NP' is mapped to 'HP' for the level k, then a nonterminal 'NP⟩⟩NP' – which is created during binarization to represent a sequence of two 'NP' children, signified with the ⟩⟩ notation – would correspond to a nonterminal 'HP⟩⟩HP' on the level

[1] Alto is available at https://bitbucket.org/tclup/alto and the grammars and fine-to-coarse mappings used are available at https://bitbucket.org/tclup/alto/downloads/coarse_to_fine_experiments_grammars_and_mappings.zip

Approach	f-score	time	invhom	sat
PCFG				
Unpruned	73.7	1230	2190	0.22
CTF	73.9	58	2260	–
TAG				
Unpruned	70.9	11258	159203	0.03
CTF:10^{-5}	51.8	11178	159203	–
CTF:10^{-9}	68.5	11198	159203	–

Table 1: Results for PCFG and TAG parsing, with mean runtime (in ms), invhom rules used in the chart, and saturation per sentence.

k. The results are shown in Table 1 (top): For IRTG encodings of treebank PCFGs, we obtain a 20x speedup at no loss in f-score.

TAG evaluation. To assess the efficacy of CTF parsing on more expressive grammar formalism, we first evaluated it on the probabilistic TAG grammar induced from WSJ Section 00 by Chen and Vijay-Shanker (2004), binarized with the "inside" strategy and converted to an IRTG. To avoid data sparsity issues, we also evaluated the grammar on Section 00, thus f-scores should be read with care. We used a variant of the four-level fine-to-coarse mapping from Charniak et al. (2006), which always preserves the distinction between nonterminals at the root of initial trees and those at the root of auxiliary trees.[2] We tried the threshold values $\theta = 10^{-5}$ and $\theta = 10^{-9}$. The results are shown in Table 1 (bottom). Unexpectedly, while CTF pruning with these thresholds already reduces the f-score, the parsing time barely improves.

HRG evaluation. We also evaluated CTF on parsing Abstract Meaning Representation (AMR) graphs with hyperedge replacement grammars (HRGs, Chiang et al. (2013)). We use the HRG grammar of Groschwitz et al. (2015), which was induced from the "Little Prince" AMR corpus (Banarescu et al., 2013) and converted to an IRTG. This grammar describes how a graph can be constructed from atomic parts. It uses complex nonterminal symbols such as $N_0\{0,\underline{1},2\}$, indicating that the nonterminal should derive a subgraph with three sources 0, 1, and 2 (these describe nodes at which further edges can be added during the derivation), and the 1-source should be the AMR's "root" node. The symbol before the curly brackets can be one of N_0 or N_1, to allow the grammar to make finer distinctions beyond the source information. In total, the grammar has 39 nonterminals and 13681 rules.

[2]For full details on the mappings used throughout the paper see the supplementary data.

Approach	best %	time	invhom	sat
Unpruned	100.0	622	9042	0.04
Self	100.0	640	–	–
– Level 1	–	629	8963	0.04
– Level 0	–	7	12	0.03
Unsplit	95.8	751	–	–
– Level 1	–	739	9930	0.05
– Level 0	–	9	13	0.04
Unroot	96.0	3279	–	–
– Level 1	–	3245	35476	0.12
– Level 0	–	31	13	0.03
Both	88.8	3926	–	–
– Level 2	–	3887	34353	0.13
– Level 1	–	36	15	0.05
– Level 0	–	1	9	0.04

Table 2: Results for HRG parsing, with mean percent best found, runtime (in ms), invhom rules used in the chart, and saturation per graph.

We tried a number of fine-to-coarse mappings in parsing Groschwitz et al.'s corpus. The "Unsplit" mapping removes the distinction between N_0 and N_1, so the above nonterminal coarsifies to $N\{0,\underline{1},2\}$. "Unroot" removes the marking of the root source, i.e. coarsifies to $N_0\{0,1,2\}$. "Both" applies the two in sequence, i.e. coarsifies to $N\{0,1,2\}$. As a sanity check, we also looked at a "Self" mapping, which "coarsifies" every nonterminal to itself. We used an aggressive pruning threshold of $\theta = 10^{-2}$.

The results are shown in Table 2. Because we do not have access to gold standard derivation trees in this dataset, we report the percentage of sentences on which a CTF parser produced the same Viterbi derivation tree as the unpruned parser ("best %"). We find that the Unsplit and Unroot mappings produce high-quality parses. However, in striking contrast to the PCFG case, all nontrivial mappings make the parser slower than the unpruned one – in the case of Unroot and Both, dramatically so.

5 Discussion

The fact that we find no speed improvements for TAG parsing and observe slowdowns for HRG parsing when we use CTF is a surprising negative result. To understand it, we first note that it is a result about CTF parsing in general and not

about our implementation: We do obtain the expected performance gains on PCFGs, and the Self mapping yields comparable HRG performance to the unpruned parser. IRTGs allow us to use the same infrastructure for CTF parsing with TAGs and HRGs which we used for CTF parsing with PCFGs. There are systematic structural differences between the PCFG, PTAG, and HRG grammars which explain the differences in the usefulness of CTF.

One difference is the number of nonterminals from which a substructure can be derived. In treebank-induced PCFGs, most substrings of sufficient length can be derived from almost every phrasal nonterminal (Klein and Manning, 2001). This is reflected in a measure called *saturation*, which we formalize as the number of chart nonterminals (A_0, J_0) that occur in the edges of the chart, divided for comparability by the total number of nonterminals A in $\mathcal{G}_C$ and the number of invhom nonterminals J used in the chart. We only compute this measure for nonterminals A_0 that were present in the grammar before binarization. For the unpruned PCFG parser, we obtain a mean saturation of 0.22, confirming Klein and Manning's findings. By contrast, mean saturation is 0.04 for the unpruned HRG parser and 0.03 for the unpruned TAG parser. Thus, the TAG and HRG grammars are more restrictive than the PCFG. The TAG grammar only derives each substring from a few nonterminals as each lexical anchor determines the root of its elementary tree; in the HRG, the annotation for the nonterminals encodes what sources are available for a merge operation. This leaves little room for CTF to increase parsing speed.

A second difference between the HRG and PCFG grammars is that only a small fraction of all of a graph's subgraphs can be derived from *any* nonterminal with the HRG grammar in the first place. This would be comparable to a setting for PCFG parsing for which most spans are ungrammatical. This is quantified by the "invhom" column in Table 2, which shows the mean number of rules of the invhom grammar that are enumerated during parsing; for HRGs, each of these describes how to split a subgraph into parts. The "Unroot" and "Both" mappings delete source information, which substantially increases the number of subgraph combinations the parser explores. PCFGs tend not to rule out many subspans, which is not a problem as the growth of subspans is only quadratic in the length of any sentence being parsed. For HRG parsing the number of possible subgraphs grows with a larger exponent, which depends on the grammar, and this means that parsing is only feasible as long as many small subgraphs can be identified as ungrammatical and many larger subgraphs are never considered. When ruling out substructures is a key element of efficient parsing, then pruning techniques other than CTF are needed for speed-ups, because the coarser grammars will generally be more permissive and therefore increase parsing times. This also makes it clear that the slowdown under CTF parsing is not a result of the particular implementation, as the number of substructures that need to be considered will be a bottleneck for any parser.

6 Conclusion

In this paper, we have defined an algorithm for coarse-to-fine parsing with IRTG grammars and, for the first time, applied CTF parsing to grammar formalisms that are more expressive than PCFGs. However, we have not observed the expected efficiency gains for such grammars, as fewer rules are equivalent and nonterminals are more informative, at least in the grammars used in our evaluation. Indeed, treebank PCFGs are especially well-suited to CTF parsing because they have many rules which only concatenate strings without introducing any terminal symbols and their nonterminals can derive almost arbitrary substrings. Neither is true for the TAG or HRG grammars we used.

Our results offer guidance on grammar requirements for successful use of CTF parsing and provide a general algorithm that will work when they are met. In the future, it would be interesting to extend CTF parsing to work in the absence of these requirements, e.g. by broadening the notion of rule equivalence, or by giving feedback from the fine level to the coarser levels in a priority search.

Acknowledgements We thank the anonymous reviewers for their comments. We are grateful to Johannes Gontrum for some early implementation work and to Mark Johnson for discussions about the paper. This work was supported by the DFG grant KO 2916/2-1. Jonas Groschwitz was supported by a Macquarie University Research Excellence Scholarship.

References

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract Meaning Representation for sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*. http://www.aclweb.org/anthology/W13-2322.

Eugene Charniak, Mark Johnson, Micha Elsner, Joseph Austerweil, David Ellis, Isaac Haxton, Catherine Hill, R. Shrivaths, Jeremy Moore, Michael Pozar, and Theresa Vu. 2006. Multilevel coarse-to-fine PCFG parsing. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the ACL*. http://www.aclweb.org/anthology/N06-1022.pdf.

John Chen and K. Vijay-Shanker. 2004. Automatic extraction of TAGs from the Penn Treebank. In *New developments in parsing technology*, Springer, pages 73–89.

David Chiang, Jacob Andreas, Daniel Bauer, Karl Moritz Hermann, Bevan Jones, and Kevin Knight. 2013. Parsing graphs with hyperedge replacement grammars. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*. https://www.aclweb.org/anthology/P/P13/P13-1091.pdf.

Hubert Comon, Max Dauchet, Rémi Gilleron, Florent Jacquemard, Denis Lugiez, Sophie Tison, Marc Tommasi, and Christof Löding. 2007. *Tree Automata techniques and applications*. published online - `http://tata.gforge.inria.fr/`. http://tata.gforge.inria.fr/.

Jonas Groschwitz, Alexander Koller, and Mark Johnson. 2016. Efficient techniques for parsing with tree automata. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*. http://aclweb.org/anthology/P16-1192.

Jonas Groschwitz, Alexander Koller, and Christoph Teichmann. 2015. Graph parsing with S-graph Grammars. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*. http://www.aclweb.org/anthology/P15-1143.

Aravind K. Joshi and Yves Schabes. 1997. Tree-Adjoining Grammars. In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Languages*, Springer-Verlag, volume 3.

Dan Klein and Christopher D. Manning. 2001. Parsing with treebank grammars: Empirical bounds, theoretical models, and the structure of the Penn Treebank. In *Proceedings of 39th Annual Meeting of the Association for Computational Linguistics*. http://aclweb.org/anthology/P/P01/P01-1044.pdf.

Dan Klein and Christopher D. Manning. 2003. A* parsing: fast exact viterbi parse selection. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*. https://doi.org/10.3115/1073445.1073461.

Alexander Koller. 2015. Semantic construction with graph grammars. In *Proceedings of the 11th International Conference on Computational Semantics*. pages 228–238. http://anthology.aclweb.org/W/W15/W15-0127.pdf.

Alexander Koller and Marco Kuhlmann. 2011. A generalized view on parsing and translation. In *Proceedings of the 12th International Conference on Parsing Technologies*. http://www.aclweb.org/anthology/W11-2902.

Alexander Koller and Marco Kuhlmann. 2012. Decomposing TAG algorithms using simple algebraizations. In *Proceedings of the 11th International Workshop on Tree Adjoining Grammars and Related Formalisms (TAG+11)*. http://aclweb.org/anthology/W/W12/W12-4616.pdf.

Slav Petrov and Dan Klein. 2007. Improved inference for unlexicalized parsing. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics*. http://www.aclweb.org/anthology/N/N07/N07-1051.

Andreas van Cranenburgh. 2012. Efficient parsing with linear context-free rewriting systems. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*. http://aclweb.org/anthology/E12-1047.

Yi Zhang and Hans-Ulrich Krieger. 2011. Large-scale corpus-driven pcfg approximation of an hpsg. In *Proceedings of the 12th International Conference on Parsing Technologies*. http://www.aclweb.org/anthology/W11-2923.