Two monolingual parses are better than one (synchronous parse)*

Chris Dyer
UMIACS Laboratory for Computational Linguistics and Information Processing
Department of Linguistics
University of Maryland, College Park, MD 20742, USA
redpony AT umd.edu

Abstract

We describe a synchronous parsing algorithm that is based on two successive *monolingual* parses of an input sentence pair. Although the worst-case complexity of this algorithm is and must be $O(n^6)$ for binary SCFGs, its average-case run-time is far better. We demonstrate that for a number of common synchronous parsing problems, the two-parse algorithm substantially outperforms alternative synchronous parsing strategies, making it efficient enough to be utilized without resorting to a pruned search.

1 Introduction

Synchronous context free grammars (SCFGs) generalize monolingual context-free grammars to generate strings concurrently in pairs of languages (Lewis and Stearns, 1968) in much the same way that finite state transducers (FSTs) generalize finite state automata (FSAs).[1] *Synchronous parsing* is the problem of finding the best derivation, or forest of derivations, of a source and target sentence pair $\langle \mathbf{f}, \mathbf{e} \rangle$ under an SCFG, $\mathcal{G}$.[2] Solving this problem is necessary for several applications, for example, optimizing how well an SCFG translation model fits parallel training data. Wu (1997) describes a bottom-up $O(n^6)$ synchronous parsing algorithm for ITGs, a binary SCFG with a restricted form. For general grammars, the situation is even worse: the problem has been shown to be NP-hard (Satta and Peserico, 2005). Even if we restrict ourselves to binary ITGs, the $O(n^6)$ run-time makes large-scale learning applications infeasible. The usual solution is to use a heuristic search that avoids exploring edges that are likely (but not guaranteed) to be low probability (Zhang et al., 2008; Haghighi et al., 2009). In this paper, we derive an alternative synchronous parsing algorithm starting from a conception of parsing with SCFGs as a composition of binary relations. This enables us to factor the synchronous parsing problem into two successive *monolingual* parses. Our algorithm runs more efficiently than $O(n^6)$ with many grammars (including those that required using heuristic search with other parsers), making it possible to take advantage of synchronous parsing without developing search heuristics; and the SCFGs are not required to be in a normal form, making it possible to easily parse with more complex SCFG types.

*This work was supported in part by the GALE program of DARPA, Contract No. HR0011-06-2-001. The author wishes to thank Philip Rensik, Adam Lopez, Phil Blunsom, and Jason Eisner for helpful discussions.

[1]SCFGs have enjoyed a resurgence in popularity as the formal basis for a number of statistical translation systems, e.g. Chiang (2007). However, translation requires only the manipulation of SCFGs using monolingual parsing algorithms.

[2]It is assumed that $n = |\mathbf{f}| \approx |\mathbf{e}|$.

2 Synchronous parsing

Before presenting our algorithm, we review the $O(n^6)$ synchronous parser for binary ITGs.[3]

2.1 ITG synchronous parsing algorithm

Wu (1997) describes a bottom-up synchronous parsing algorithm that can be understood as a generalization of the CKY algorithm. CKY defines a table consisting of n^2 cells, with each cell corresponding to a span $[i, j]$ in the input sentence; and the synchronous variant defines a table in 4 dimensions, with cells corresponding to a source span $[s, t]$ *and* a target span $[u, v]$. The bottom of the chart is initialized first, and pairs of items are combined from bottom to top. Since combining items from the n^4 cells involves considering two split points (one source, one target), it is not hard to see that this algorithm runs in time $O(n^6)$.

[3]Generalizing the algorithm to higher rank grammars is possible (Wu, 1997), as is converting a grammar to a weakly equivalent binary form in some cases (Huang et al., 2009).

2.2 Parsing, intersection, and composition

We motivate an alternative conception of the synchronous parsing problem as follows. It has long been appreciated that monolingual parsing computes the intersection of an FSA and a CFG (Bar-Hillel et al., 1961; van Noord, 1995). That is, if S is an FSA encoding some sentence **s**, intersection of S with a CFG, $\mathcal{G}$, results in a parse forest which contains all and only derivations of **s**, that is $L(S) \cap L(\mathcal{G}) \in \{\{\mathbf{s}\}, \emptyset\}$.[4] Crucially for our purposes, the resulting parse forest *is also itself a CFG*.[5] Figure 1 illustrates, giving two equivalent representations of the forest $S \cap \mathcal{G}$, once as a directed hypergraph and once as a CFG. While $S \cap \mathcal{G}$ appears similar to $\mathcal{G}$, the non-terminals (NTs) of the resulting CFG are a cross product of pairs of states from S and NTs from $\mathcal{G}$.[6]

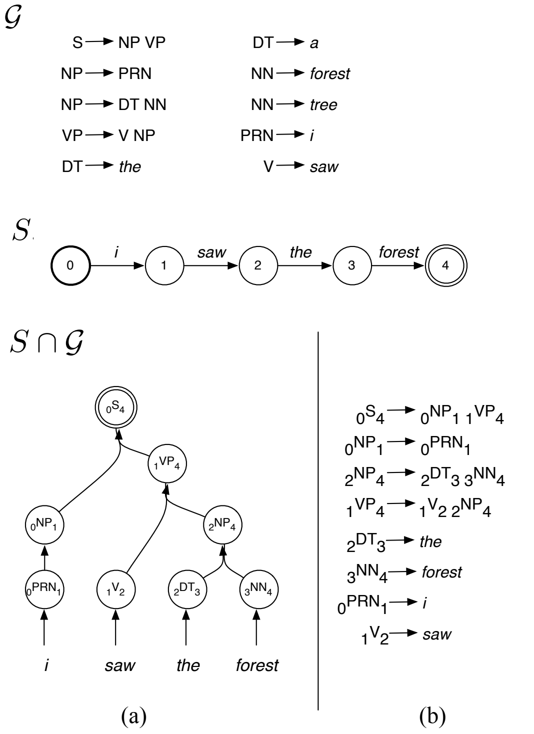
$\mathcal{G}$
S → NP VP
NP → PRN
NP → DT NN
VP → V NP
DT → *the*
DT → *a*
NN → *forest*
NN → *tree*
PRN → *i*
V → *saw*
S
0
i
1
saw
2
the
3
forest
4
$S \cap \mathcal{G}$
$_0S_4$
$_1VP_4$
$_0NP_1$
$_2NP_4$
$_0PRN_1$
$_1V_2$
$_2DT_3$
$_3NN_4$
i
saw
the
forest
$_0S_4 \rightarrow {_0NP_1}\ {_1VP_4}$
$_0NP_1 \rightarrow {_0PRN_1}$
$_2NP_4 \rightarrow {_2DT_3}\ {_3NN_4}$
$_1VP_4 \rightarrow {_1V_2}\ {_2NP_4}$
$_2DT_3 \rightarrow$ *the*
$_3NN_4 \rightarrow$ *forest*
$_0PRN_1 \rightarrow$ *i*
$_1V_2 \rightarrow$ *saw*
(a)
(b)

Figure 1: A CFG, $\mathcal{G}$, an FSA, S, encoding a sentence, and two equivalent representations of the parse forest $S \cap \mathcal{G}$, (a) as a directed hypergraph and (b) as a CFG.

When dealing with SCFGs, rather than intersection, parsing computes a related operation, *composition*.[7] The standard MT decoding-by-parsing task can be understood as computing the composition of an FST,[8] F, which encodes the source sentence **f** with the SCFG, $\mathcal{G}$, representing the translation model. The result is the translation forest, $F \circ \mathcal{G}$, which encodes all translations of **f** licensed by the translation model. While $\mathcal{G}$ can generate a potentially infinite set of strings in the source and target languages, $F \circ \mathcal{G}$ generates *only* **f** in the source language (albeit with possibly infinitely many derivations), but any number of different strings in the target language. It is not hard to see that a second composition operation of an FST, E, encoding the target string **e** with the e-side of $F \circ \mathcal{G}$ (again using a monolingual parsing algorithm), will result in a parse forest that exactly derives $\langle \mathbf{f}, \mathbf{e} \rangle$, which is the goal of synchronous composition. Figure 2 shows an example. In $F \circ \mathcal{G} \circ E$ the NTs (nodes) are the cross product of pairs of states from E, the NTs from $\mathcal{G}$, and pairs of states in F.

Thus, synchronous parsing is the task of computing $F \circ \mathcal{G} \circ E$. Since composition is associative, we can compute this quantity either as $(F \circ \mathcal{G}) \circ E$ or $F \circ (\mathcal{G} \circ E)$. Alternatively, we can use an algorithm that performs 3-way composition directly.

2.3 The two-parse algorithm[9]

The *two-parse algorithm* refers to performing a synchronous parse by computing either $(F \circ \mathcal{G}) \circ E$ or $F \circ (\mathcal{G} \circ E)$. Each composition operation is carried out using a standard monolingual parsing algorithm, such as Earley's or CKY. In the experiments below, since we use ϵ-free grammars, we use a variant of CKY for unrestricted CFGs (Chiang, 2007).

Once the first composition is done, the resulting parse forest must be converted into a CFG representation that the second parser can utilize. This is straightforward to do: each node becomes a unique non-terminal symbol, with its incoming edges corresponding to different ways of rewriting it. Tails of edges are non-terminal variables in the RHS of these rewrites. A single bottom-up traversal of the forest is sufficient to perform the conversion. Since

[4]$L(x)$ denotes the set of strings generated by the grammar/automaton x. In future mentions of intersection and composition operations, this will be implicit.

[5]The forest grammar derives only **s**, but using possibly many derivations.

[6]Each pair of states from the FSA corresponds to a span $[i, j]$ in a CKY table.

[7]Intersection is a special case of composition where the input and output labels on the transducers are identical (Mohri, 2009).

[8]FSTs used to represent the source and target sentences have identical input and output labels on every transition.

[9]Satta (submitted) has independently derived this algorithm.

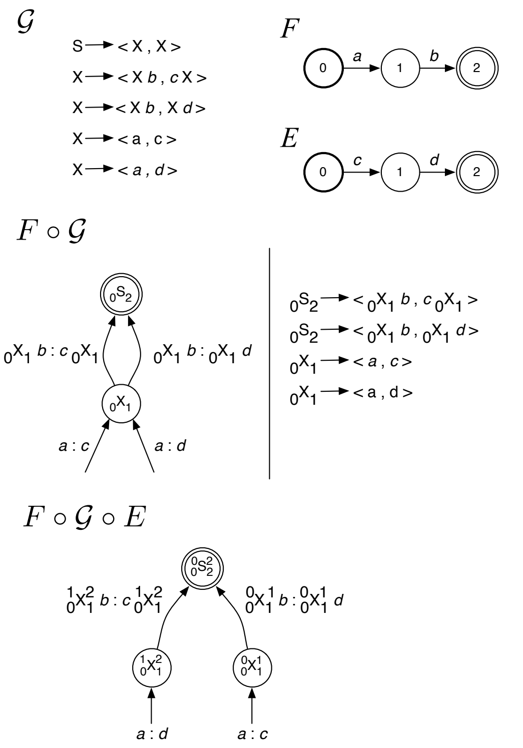

Figure 2: An SCFG, $\mathcal{G}$, two FSAs, E and F, and two equivalent representations of $F \circ \mathcal{G}$. The synchronous parse forest of the pair $\langle ab, cd \rangle$ with $\mathcal{G}$ is given under $F \circ \mathcal{G} \circ E$.

our parser operates more efficiently with a determinized grammar, we left-factor the grammar during this traversal as well.

Analysis. Monolingual parsing runs in worst case $O(|\mathcal{G}| \cdot n^3)$ time, where n is the length of the input being parsed and $|\mathcal{G}|$ is a measure of the size of the grammar (Graham et al., 1980). Since the grammar term is constant for most typical parsing applications, it is generally not considered carefully; however, in the two-parse algorithm, the size of the grammar term for the second parse is not $|\mathcal{G}|$ but $|F \circ \mathcal{G}|$, which clearly depends on the size of the input F; and so understanding the impact of this term is key to understanding the algorithm's run-time.

If $\mathcal{G}$ is an ϵ-free SCFG with non-terminals N and maximally two NTs in a rule's right hand side, and n is the number of states in F (corresponding to the number of words in the $\mathbf{f}$ in a sentence pair $\langle \mathbf{f}, \mathbf{e} \rangle$), then the number of nodes in the parse forest $F \circ \mathcal{G}$ will be $O(|N| \cdot n^2)$. This can be shown easily since by stipulation, we are able to use CKY+ to perform the parse, and there will be maximally as many nodes in the forest as there are cells in the CKY chart times the number of NTs. The number of edges will be $O(|N| \cdot n^3)$, which occurs when every node can be derived from all possible splits. This bound on the number of edges implies that $|F \circ \mathcal{G}| \in O(n^3)$.[10] Therefore, the worst case run-time of the two-parse algorithm is $O(|N| \cdot n^3 \cdot n^3 + |\mathcal{G}| \cdot n^3) = O(|N| \cdot n^6)$, the same as the bound on the ITG algorithm. We note that while the ITG algorithm requires that the SCFGs be rank-2 and in a normal form, the two-parse algorithm analysis holds as long as the grammars are rank-2 and ϵ-free.[11]

3 Experiments

We now describe two different synchronous parsing applications, with different classes of SCFGs, and compare the performance of the two-parse algorithm with that of previously used algorithms.

Phrasal ITGs. Here we compare performance of the two-parse algorithm and the $O(n^6)$ ITG parsing algorithm on an Arabic-English phrasal ITG alignment task. We used a variant of the phrasal ITG described by Zhang et al. (2008).[12] Figure 3 plots the average run-time of the two algorithms as a function of the Arabic sentence length. The two-parse approach is far more efficient. In total, aligning the 80k sentence pairs in the corpus completed in less than 4 hours with the two-parse algorithm but required more than 1 week with the baseline algorithm.[13]

"Hiero" grammars. An alternative approach to computing a synchronous parse forest is based on *cube pruning* (Huang and Chiang, 2007). While more commonly used to integrate a target m-gram LM during decoding, Blunsom et al. (2008), who required synchronous parses to discriminatively train

[10]How tight these bounds are depends on the ambiguity in the grammar w.r.t. the input: to generate n^3 edges, every item in every cell must be derivable by every combination of its subspans. Most grammars are substantially less ambiguous.

[11]Since many widely used SCFGs meet these criteria, including hierarchical phrase-based translation grammars (Chiang, 2007), SAMT grammars (Zollmann and Venugopal, 2006), and phrasal ITGs (Zhang et al., 2008), a detailed analysis of ϵ-containing and higher rank grammars is left to future work.

[12]The restriction that phrases contain exactly a single alignment point was relaxed, resulting in much larger and more ambiguous grammars than those used in the original work.

[13]A note on implementation: our ITG aligner was minimal; it only computed the probability of the sentence pair using the inside algorithm. With the two-parse aligner, we stored the complete forest during both the first and second parses.

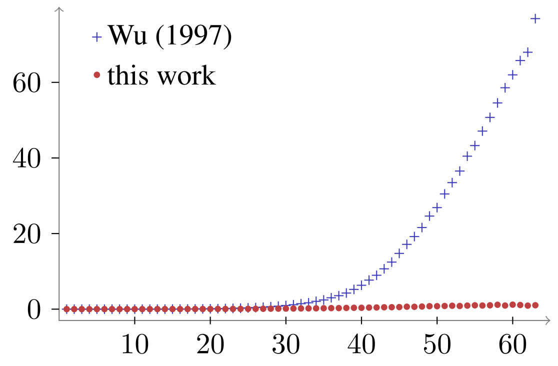

Figure 3: Average synchronous parser run-time (in seconds) as a function of Arabic sentence length (in words).

an SCFG translation model, repurposed this algorithm to discard partial derivations during translation of **f** if the derivation yielded a target m-gram not found in **e** (p.c.). We replicated their BTEC Chinese-English baseline system and compared the speed of their 'cube-parsing' technique and our two-parse algorithm.[14] The SCFG used here was extracted from a word-aligned corpus, as described in Chiang (2007).[15] The following table compares the average per sentence synchronous parse time.

Algorithm	avg. run-time (sec)
Blunsom et al. (2008)	7.31
this work	0.20

4 Discussion

Thinking of synchronous parsing as two composition operations has both conceptual and practical benefits. The two-parse strategy can outperform both the ITG parsing algorithm (Wu, 1997), as well as the 'cube-parsing' technique (Blunsom et al., 2008). The latter result points to a connection with recent work showing that determinization of edges before LM integration leads to fewer search errors during decoding (Iglesias et al., 2009).

Our results are somewhat surprising in light of work showing that 3-way composition algorithms for FSTs operate far more efficiently than performing successive pairwise compositions (Allauzen and Mohri, 2009). This is certainly because the 3-way algorithm used here (the ITG algorithm) does an exhaustive search over all n^4 span pairs without awareness of any top-down constraints. This suggests that faster composition algorithms that incorporate top-down filtering may still be discovered.

[14]To the extent possible, the two experiments were carried out using the exact same code base, which was a C++ implementation of an SCFG-based decoder.

[15]Because of the mix of terminal and non-terminal symbols, such grammars cannot be used by the ITG synchronous parsing algorithm.

References

C. Allauzen and M. Mohri. 2009. N-way composition of weighted finite-state transducers. *International Journal of Foundations of Comp. Sci.*, 20(4):613–627.

Y. Bar-Hillel, M. Perles, and E. Shamir. 1961. On formal properties of simple phrase structure grammars. *Zeitschrift für Phonetik, Sprachwissenschaft und Kommunikationsforschung*, 14:143–172.

P. Blunsom, T. Cohn, and M. Osborne. 2008. Probalistic inference for machine translation. In *EMNLP*.

D. Chiang. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, 33(2):201–228.

S. L. Graham, W. L. Ruzzo, and M. Harrison. 1980. An improved context-free recognizer. *ACM Trans. Program. Lang. Syst.*, 2(3):415–462.

A. Haghighi, J. Blitzer, J. DeNero, and D. Klein. 2009. Better word alignments with supervised ITG models. In *Proc. of ACL/IJCNLP*, pages 923–931.

L. Huang and D. Chiang. 2007. Forest rescoring: Faster decoding with integrated language models. In *ACL*.

L. Huang, H. Zhang, D. Gildea, and K. Knight. 2009. Binarization of synchronous context-free grammars. *Computational Linguistics*, 35(4).

G. Iglesias, A. de Gispert, E. R. Banga, and W. Byrne. 2009. Hierarchical phrase-based translation with weighted finite state transducers. In *Proc. NAACL*.

P. M. Lewis, II and R. E. Stearns. 1968. Syntax-directed transduction. *J. ACM*, 15(3):465–488.

M. Mohri. 2009. Weighted automata algorithms. In M. Droste, W. Kuich, and H. Vogler, editors, *Handbook of Weighted Automata*, Monographs in Theoretical Computer Science, pages 213–254. Springer.

G. Satta and E. Peserico. 2005. Some computational complexity results for synchronous context-free grammars. In *Proceedings of NAACL*.

G. Satta. submitted. Translation algorithms by means of language intersection.

G. van Noord. 1995. The intersection of finite state automata and definite clause grammars. In *Proc. of ACL*.

D. Wu. 1997. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*, 23(3):377–404.

H. Zhang, C. Quirk, R. C. Moore, and D. Gildea. 2008. Bayesian learning of non-compositional phrases with synchronous parsing. In *Proceedings of ACL*.

A. Zollmann and A. Venugopal. 2006. Syntax augmented machine translation via chart parsing. In *Proc. of the Workshop on SMT*.