Intersecting Hierarchical and Phrase-Based Models of Translation: Formal Aspects and Algorithms

Marc Dymetman **Nicola Cancedda**
Xerox Research Centre Europe
{marc.dymetman,nicola.cancedda}@xrce.xerox.com

Abstract

We address the problem of constructing hybrid translation systems by intersecting a Hiero-style hierarchical system with a phrase-based system and present formal techniques for doing so. We model the phrase-based component by introducing a variant of weighted finite-state automata, called σ-automata, provide a self-contained description of a general algorithm for intersecting weighted synchronous context-free grammars with finite-state automata, and extend these constructs to σ-automata. We end by briefly discussing complexity properties of the presented algorithms.

1 Introduction

Phrase-based (Och and Ney, 2004; Koehn et al., 2007) and Hierarchical (Hiero-style) (Chiang, 2007) models are two mainstream approaches for building Statistical Machine Translation systems, with different characteristics. While phrase-based systems allow a direct capture of correspondences between surface-level lexical patterns, but at the cost of a simplistic handling of re-ordering, hierarchical systems are better able to constrain re-ordering, especially for distant language pairs, but tend to produce sparser rules and often lag behind phrase-based systems for less distant language pairs. It might therefore make sense to capitalize on the complementary advantages of the two approaches by combining them in some way.

This paper attempts to lay out the formal prerequisites for doing so, by developing techniques for intersecting a hierarchical model and a phrase-based model. In order to do so, one first difficulty has to be overcome: while hierarchical systems are based on the mathematically well-understood formalism of weighted synchronous CFG's, phrase-based systems do not correspond to any classical formal model, although they are loosely connected to weighted finite state transducers, but crucially go beyond these by allowing phrase re-orderings.

One might try to address this issue by limiting *a priori* the amount of re-ordering, in the spirit of (Kumar and Byrne, 2005), which would allow to approximate a phrase-based model by a standard transducer, but this would introduce further issues. First, limiting the amount of reordering in the phrase-based model runs contrary to the underlying intuitions behind the intersection, namely that the hierarchical model should be mainly responsible for controlling re-ordering, and the phrase-based model mainly responsible for lexical choice. Second, the transducer resulting from the operation could be large. Third, even if we could represent the phrase-based model through a finite-state transducer, intersecting this transducer with the synchronous CFG would actually be intractable in the general case, as we indicate later.

We then take another route. For a fixed source sentence x, we show how to construct an automaton that represents all the (weighted) target sentences that can be produced by applying the phrase based model to x. However, this "σ-automaton" is non-standard in the sense that each transition is decorated with a set of source sentence tokens and that the only valid paths are

Proceedings of SSST-4, Fourth Workshop on Syntax and Structure in Statistical Translation, pages 1–9,
COLING 2010, Beijing, August 2010.

those that do not traverse two sets containing the same token (in other words, valid paths cannot "consume" the same source token twice).

The reason we are interested in σ-automata is the following. First, it is known that intersecting a synchronous grammar simultaneously with the source sentence x and a (standard) target automaton results in another synchronous grammar; we provide a self-contained description of an algorithm for performing this intersection, in the general weighted case, and where x is generalized to an arbitrary source automaton. Second, we extend this algorithm to σ-automata. The resulting weighted synchronous grammar represents, as in Hiero, the "parse forest" (or "hypergraph") of all weighted derivations (that is of all translations) that can be built over x, but where the weights incorporate knowledge of the phrase-based component; it can therefore form the basis of a variety of dynamic programming or sampling algorithms (Chiang, 2007; Blunsom and Osborne, 2008), as is the case with standard Hiero-type representations. While in the worst case the intersected grammar can contain an exponential number of nonterminals, we argue that such combinatorial explosion will not happen in practice, and we also briefly indicate formal conditions under which it will not be *allowed* to happen.

2 Intersecting weighted synchronous CFG's with weighted automata

We assume that the notions of weighted finite-state automaton [W-FSA] and weighted synchronous grammar [W-SCFG] are known (for short descriptions see (Mohri et al., 1996) and (Chiang, 2006)), and we consider:

1. A W-SCFG G, with associated source grammar G_s (resp. target grammar G_t); the terminals of G_s (resp. G_t) vary over the source vocabulary V_s (resp. target vocabulary V_t).
2. A W-FSA A_s over the source vocabulary V_s, with initial state $s_\#$ and final state $s_\$$.
3. A W-FSA A_t over the target vocabulary V_t, with initial state $t_\#$ and final state $t_\$$.

The grammar G defines a weighted synchronous language L_G over (V_s, V_t), the automaton A_s a weighted language L_s over V_s, and the automaton A_t a weighted language L_t over V_t. We then define the intersection language L' between these three languages as the synchronous language denoted $L' = L_s \Cap L_G \Cap L_t$ over (V_s, V_t) such that, for any pair (x, y) of a source and a target sentence, the weight $L'(x, y)$ is defined by $L'(x, y) \equiv L_s(x) \cdot L_G(x, y) \cdot L_t(y)$, where $L_s(x), L_G(x, y), L_t(y)$ are the weights associated to each of the component languages.

It is natural to ask whether there exists a synchronous grammar G' generating the language L', which we will now show to be the case.[1] Our approach is inspired by the construction in (Bar-Hillel et al., 1961) for the intersection of a CFG and an FSA and the observation in (Lang, 1994) relating this construction to parse forests, and also partially from (Satta, 2008), although, by contrast to that work, our construction, (i) is done simultaneously rather than as the sequence of intersecting A_s with G, then the resulting grammar with A_t, (ii) handles weighted formalisms rather than non-weighted ones.

We will describe the construction of G' based on an example, from which the general construction follows easily. Consider a W-SCFG grammar G for translating between French and English, with initial nonterminal S, and containing among others the following rule:

$$\mathrm{N} \rightarrow \text{A manque à B / B misses A} : \theta, \qquad (1)$$

where the source and target right-hand sides are separated by a slash symbol, and where θ is a non-negative real weight (interpreted multiplicatively) associated with the rule.

Now let's consider the following "rule scheme":

$$\begin{aligned} {}^{t_0}_{s_0}\mathrm{N}^{t_3}_{s_4} \rightarrow {}&{}^{t_2}_{s_0}\mathrm{A}^{t_3}_{s_1}\ {}_{s_1}\text{manque}_{s_2}\ {}_{s_2}\text{à}_{s_3}\ {}^{t_0}_{s_3}\mathrm{B}^{t_1}_{s_4} \quad / \\ & {}^{t_0}_{s_3}\mathrm{B}^{t_1}_{s_4}\ {}^{t_1}\text{misses}^{t_2}\ {}^{t_2}_{s_0}\mathrm{A}^{t_3}_{s_1} \end{aligned} \qquad (2)$$

[1] We will actually only need the application of this result to the case where A_s is a "degenerate" automaton describing a single source sentence x, but the general construction is not harder to do than this special case and the resulting format for G' is well-suited to our needs below.

This scheme consists in an "indexed" version of the original rule, where the bottom indices s_i correspond to states of A_s ("source states"), and the top indices t_i to states of A_t ("target states"). The nonterminals are associated with two source and two target indices, and for the same nonterminal, these four indices have to match across the source and the target RHS's of the rule. As for the original terminals, they are replaced by "indexed terminals", where source (resp. target) terminals have two source (resp. target) indices. The source indices appear sequentially on the source RHS of the rule, in the pattern $s_0, s_1, s_1, s_2, s_2 \dots s_{m-1}, s_m$, with the nonterminal on the LHS receiving source indices s_0 and s_m, and similarly the target indices appear sequentially on the target RHS of the rule, in the pattern $t_0, t_1, t_1, t_2, t_2 \dots t_{n-1}, t_n$, with the nonterminal on the LHS receiving target indices t_0 and t_n. To clarify, the operation of associating indices to terminals and nonterminals can be decomposed into three steps:

$$
\begin{aligned}
{}_{s_0}\mathrm{N}_{s_4} &\rightarrow {}_{s_0}\mathrm{A}_{s_1}\ {}_{s_1}\text{manque}_{s_2}\ {}_{s_2}\text{à}\ {}_{s_3}\ {}_{s_3}\mathrm{B}_{s_4}\ / \\
&\qquad \text{B misses A} \\
{}^{t_0}\mathrm{N}^{t_3} &\rightarrow \text{A manque à B}\ / \\
&\qquad {}^{t_0}\mathrm{B}^{t_1}\ {}^{t_1}\text{misses}^{t_2}\ {}^{t_2}\mathrm{A}^{t_3} \\
{}^{t_0}_{s_0}\mathrm{N}^{t_3}_{s_4} &\rightarrow {}^{t_2}_{s_0}\mathrm{A}^{t_3}_{s_1}\ {}_{s_1}\text{manque}_{s_2}\ {}_{s_2}\text{à}\ {}_{s_3}\ {}^{t_0}_{s_3}\mathrm{B}^{t_1}_{s_4}\ / \\
&\qquad {}^{t_0}_{s_3}\mathrm{B}^{t_1}_{s_4}\ {}^{t_1}\text{misses}^{t_2}\ {}^{t_2}_{s_0}\mathrm{A}^{t_3}_{s_1}
\end{aligned}
$$

where the first two steps corresponds to handling the source and target indices separately, and the third step then assembles the indices in order to get the same four indices on the two copies of each RHS nonterminal. The rule scheme (2) now generates a family of rules, each of which corresponds to an *arbitrary* instantiation of the source and target indices to states of the source and target automata respectively. With every such rule instantiation, we associate a weight θ' which is defined as:

$$
\begin{aligned}
\theta' \equiv \theta \cdot & \prod_{{}_{s_i}\text{s-term}_{s_{i+1}}} \theta_{A_s}(s_i, \text{s-term}, s_{i+1}) \\
\cdot & \prod_{{}^{t_j}\text{t-term}^{t_{j+1}}} \theta_{A_t}(t_j, \text{t-term}, t_{j+1}), \quad (3)
\end{aligned}
$$

where the first product is over the indexed source terminals ${}_{s_i}\text{s-term}_{s_{i+1}}$, the second product over the indexed target terminals ${}^{t_j}\text{t-term}^{t_{j+1}}$; $\theta_{A_s}(s_i, \text{s-term}, s_{i+1})$ is the weight of the transition $(s_i, \text{s-term}, s_{i+1})$ according to A_s, and similarly for $\theta_{A_t}(t_j, \text{t-term}, t_{j+1})$. In these products, it may happen that $\theta_{A_s}(s_i, \text{s-term}, s_{i+1})$ is null (and similarly for A_t), and in such a case, the corresponding rule instantiation is considered not to be realized. Let us consider the multiset of all the weighted rule instantiations for (1) computed in this way, and for each rule in the collection, let us "forget" the indices associated to the terminals. In this way, we obtain a collection of weighted synchronous rules over the vocabularies V_s and V_t, but where each nonterminal is now indexed by four states.[2]

When we apply this procedure to all the rules of the grammar G, we obtain a new weighted synchronous CFG G', with start symbol ${}^{t_\#}_{s_\#}S^{t_\$}_{s_\$}$, for which we have the following **Fact**, of which we omit the proof for lack of space.

Fact 1. *The synchronous language $L_{G'}$ associated with G' is equal to $L' = L_s \Cap L_G \Cap L_t$.*

The grammar G' that we have just constructed does fulfill the goal of representing the bilateral intersection that we were looking for, but it has a serious defect: most of its nonterminals are *improductive*, that is, can never produce a bi-sentence. If a rule refers to such an improductive nonterminal, it can be eliminated from the grammar. This is the analogue for a SCFG of the classical operation of *reduction* for CFG's; while, conceptually, we could start from G' and perform the reduction by *deleting* the many rules containing improductive nonterminals, it is equivalent but much more efficient to do the reverse, namely to incrementally *add* the productive nonterminals and rules of G' starting from an initially empty set of rules, and by proceeding bottom-up starting from the terminals. We do not detail this process, which is relatively

[2] It is possible that the multiset obtained by this simplifying operation contains duplicates of certain rules (possibly with different weights), due to the non-determinism of the automata: for instance, two sequences such as '${}_{s_1}\text{manque}_{s_2}\ {}_{s_2}\text{à}_{s_3}$' and '${}_{s_1}\text{manque}_{s'_2}\ {}_{s'_2}\text{à}_{s_3}$' become indistinguishable after the operation. Rather than producing multiple instances of rules in this way, one can "conflate" them together and add their weights.

straightforward.[3]

A note on intersecting SCFGs with transducers Another way to write $L_s \Cap L_G \Cap L_t$ is as the intersection $(L_s \times L_t) \cap L_G$. $(L_s \times L_t)$ can be seen as a rational language (language generated by a finite state transducer) of an especially simple form over $V_s \times V_t$. It is then natural to ask whether our previous construction can be generalized to the intersection of G with an arbitrary finite-state transducer. However, this is not the case. Deciding the emptiness problem for the intersection between two finite state transducers is already undecidable, by reduction to Post's Correspondence problem (Berstel, 1979, p. 90) and we have extended the proof of this fact to show that intersection between a synchronous CFG and a finite state transducer also has an undecidable emptiness problem (the proof relies on the fact that a finite state transducer can be simulated by a synchronous grammar). *A fortiori*, this intersection cannot be represented through an (effectively constructed) synchronous CFG.

3 Phrase-based models and σ-automata

3.1 σ-automata: definition

Let V_s be a source vocabulary, V_t a target vocabulary. Let $x = x_1, \ldots, x_M$ be a *fixed* sequence of words over a certain source vocabulary V_s. Let us denote by z a *token* in the sequence x, and by Z the set of the M tokens in x. A σ-automaton over x has the general form of a standard weighted automaton over the target vocabulary, but where the edges are also decorated with elements of $\mathcal{P}(Z)$, the powerset of Z (see Fig. 1). An edge in the σ-automaton between two states q and q' then carries a label of the form (α, β), where $\alpha \in \mathcal{P}(Z)$ and $\beta \in V_t$ (note that here we do not allow β to be the empty string ϵ). A path from the initial state of the automaton to its final state is defined to be *valid* iff each token of x appears in exactly one label of the path, but not necessarily in the same order as in x. As usual, the output associated with the path is the ordered sequence of target labels on that path, and the weight of the path is the product of the weights on its edges.

σ-automata and phrase-based translation A mainstream phrase-based translation system such as Moses (Koehn et al., 2007) can be accounted for in terms of σ-automata in the following way. To simplify exposition, we assume that the language model used is a bigram model, but any n-gram model can be accommodated. Then, *given a source sentence* x, decoding works by attempting to construct a sequence of phrase-pairs of the form $(\tilde{x}_1, \tilde{y}_1), ..., (\tilde{x}_k, \tilde{y}_k)$ such that each $\tilde{x}_i$ corresponds to a contiguous subsequence of *tokens* of x, the $\tilde{x}_i$'s do not overlap and completely cover x, but may appear in a different order than that of x; the output associated with the sequence is simply the concatenation of all the $\tilde{y}_i$'s in that sequence.[4] The weight associated with the sequence of phrase-pairs is then the product (when we work with probabilities rather than log-probabilities) of the weight of each $(\tilde{x}_{i+1}, \tilde{y}_{i+1})$ in the context of the previous $(\tilde{x}_i, \tilde{y}_i)$, which consists in the product of several elements: (i) the "out-of-context" weight of the phrase-pair $(\tilde{x}_{i+1}, \tilde{y}_{i+1})$ as determined by its features in the phrase table, (ii) the language model probability of finding $\tilde{y}_{i+1}$ following $\tilde{y}_i$,[5] (iii) the contextual weight of $(\tilde{x}_{i+1}, \tilde{y}_{i+1})$ relative to $(\tilde{x}_i, \tilde{y}_i)$ corresponding to the distorsion cost of "jumping" from the token sequence $\tilde{x}_i$ to the token sequence $\tilde{x}_{i+1}$ when these two sequences may not be consecutive in x.[6]

Such a model can be represented by a σ-automaton, where each phrase-pair $(\tilde{x}, \tilde{y})$ — for

[3] This bottom-up process is analogous to *chart-parsing*, but here we have decomposed the construction into first building a semantics-preserving grammar and then reducing it, which we think is formally neater.

[4] We assume here that the phrase-pairs $(\tilde{x}_i, \tilde{y}_i)$ are such that $\tilde{y}_i$ is not the empty string (this constraint could be removed by an adaptation of the ϵ-removal operation (Mohri, 2002) to σ-automata).

[5] This is where the bigram assumption is relevant: for a trigram model, we may need to encode in the automaton not only the immediately preceding phrase-pair, but also the previous one, and so on for higher-order models. An alternative is to keep the n-gram language model outside the σ-automaton and intersect it later with the grammar G' obtained in section 4, possibly using approximation techniques such as cube-pruning (Chiang, 2007).

[6] Any distorsion model — in particular "lexicalized reordering" — that only depends on comparing two consecutive phrase-pairs can be implemented in this way.

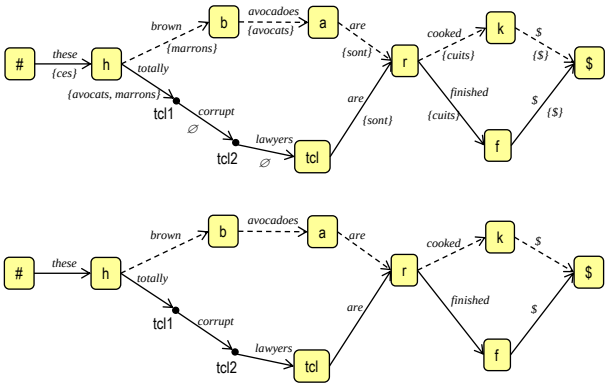


Figure 1: **On the top:** a σ-automaton with two valid paths shown. Each box denotes a state corresponding to a phrase pair, while states internal to a phrase pair (such as tcl1 and tcl2) are not boxed. Above each transition we have indicated the corresponding target word, and below it the corresponding set of source tokens. We use a terminal symbol \$ to denote the end of sentence both on the source and on the target. The solid path corresponds to the output *these totally corrupt lawyers are finished*, the dotted path to the output *these brown avocadoes are cooked*. Note that the source tokens are not necessarily consumed in the order given by the source, and that, for example, there exists a valid path generating *these are totally corrupt lawyers finished* and moving according to $h \rightarrow r \rightarrow tcl1 \rightarrow tcl2 \rightarrow tcl \rightarrow f$; Note, however, that this does not mean that if a biphrase such as `(marrons avocats, avocado chestnuts)` existed in the phrase table, it would be applicable to the source sentence here: because the source words in this biphrase would not match the order of the source tokens in the sentence, the biphrase would not be included in the σ-automaton at all. **On the bottom:** The target W-FSA automaton A_t associated with the σ-automaton, where we are ignoring the source tokens (but keeping the same weights).

$\tilde{x}$ a sequence of tokens in x and $(\tilde{x}, \tilde{y})$ an entry in the global phrase table — is identified with a state of the automaton and where the fact that the phrase-pair $(\tilde{x}', \tilde{y}') = ((x_1, ..., x_k), (y_1, ..., y_l))$ follows $(\tilde{x}, \tilde{y})$ in the decoding sequence is modeled by introducing l "internal" transitions with labels (σ, y_1), $(\emptyset, y_2)$, ..., $(\emptyset, y_l)$, where $\sigma = \{x_1, ..., x_k\}$, and where the first transition connects the state $(\tilde{x}, \tilde{y})$ to some unique "internal state" q_1, the second transition the state q_1 to some unique internal state q_2, and the last transition q_k to the state $(\tilde{x}', \tilde{y}')$.[7] Thus, a state $(\tilde{x}', \tilde{y}')$ essentially encodes the previous phrase-pair used during decoding, and it is easy to see that it is possible to account for the different weights associated with the phrase-based model by weights associated to the transitions of the σ-automaton.[8]

Example Let us consider the following French source sentence x: *ces avocats marrons sont cuits* (idiomatic expression for *these totally corrupt lawyers are finished*). Let's assume that the phrase table contains the following phrase pairs:

```
h: (ces, these)
a: (avocats, avocados)
b: (marrons, brown)
tcl: (avocats marrons,
            totally corrupt lawyers)
r: (sont, are)
k: (cuits, cooked)
f: (cuits, finished).
```

An illustration of the corresponding σ-automaton SA is shown at the top of Figure 1, with only a few transitions made explicit, and with no weights shown.[9]

[7] For simplicity, we have chosen to collect the set of all the source tokens $\{x_1, ..., x_k\}$ on the first transition, but we could distribute it on the l transitions arbitrarily (but keeping the subsets disjoint) without changing the semantics of what we do. This is because once we have entered one of the l internal transitions, we will always have to traverse the remaining internal transitions and collect the full set of source tokens.

[8] By creating states such as $((\tilde{x}, \tilde{y}), (\tilde{x}', \tilde{y}'))$ that encode the two previous phrase-pairs used during decoding, it is possible in principle to account for a trigram language model, and similarly for higher-order LMs. This is similar to implementing n-gram language models by automata whose states encode the $n-1$ words previously generated.

[9] Only two (valid) paths are shown. If we had shown the full σ-automaton, then the graph would have been "complete" in the sense that for any two box states B, B', we would have shown a connection $B \rightarrow B'_1 ... \rightarrow B'_{k-1} \rightarrow B'$, where the B'_i are internal states, and k is the length of the target side of the biphrase B'.

4 Intersecting a synchronous grammar with a σ-automaton

Intersection of a W-SCFG with a σ-automaton If SA is a σ-automaton over input x, with each valid path in SA we associate a weight in the same way as we do for a weighted automaton. For any target word sequence in V_t^* we can then associate the sum of the weights of all valid paths outputting that sequence. The weighted language $L_{SA,x}$ over V_t obtained in this way is called the language associated with SA. Let G be a W-SCFG over V_s, V_t, and let us denote by $L_{G,x}$ the weighted language over V_s, V_t corresponding to the intersection $\{x\} \Cap G \Cap V_t^*$, where $\{x\}$ denotes the language giving weight 1 to x and weight 0 to other sequences in V_s^*, and V_t^* denotes the language giving weight 1 to all sequences in V_t^*. Note that non-null bi-sentences in $L_{G,x}$ have their source projection equal to x and therefore $L_{G,x}$ can be identified with a weighted language over V_t. The intersection of the languages $L_{SA,x}$ and $L_{G,x}$ is denoted by $L_{SA,x} \Cap L_{G,x}$.

Example Let us consider the following W-SCFG (where again, weights are not explicitly shown, and where we use a terminal symbol \$ to denote the end of a sentence, a technicality needed for making the grammar compatible with the SA automaton of Figure 1):

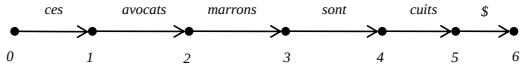
```
S → NP VP $ / NP VP $
NP → ces N A / these A N
VP → sont A / are A
A → marrons / brown
A → marrons / totally corrupt
A → cuits / cooked
A → cuits / finished
N → avocats / avocadoes
N → avocats / lawyers
```

It is easy to see that, for instance, the sentences: *these brown avocadoes are cooked \$*, *these brown avocadoes are finished \$*, and *these totally corrupt lawyers are finished \$* all belong to the intersection $L_{SA,x} \Cap L_{G,x}$, while the sentences *these avocadoes brown are cooked \$*, *totally corrupt lawyers are finished these \$* belong only to $L_{SA,x}$.

Building the intersection We now describe how to build a W-SCFG that represents the intersection $L_{SA,x} \Cap L_{G,x}$. We base our explanations on the example just given.

A Relaxation of the Intersection At the bottom of Figure 1, we show how we can associate an automaton A_t with the σ-automaton SA: we simply "forget" the source-sides of the labels carried by the transitions, and retain all the weights. As before, note that we are only showing a subset of the transitions here.

All valid paths for SA map into valid paths for A_t (with the same weights), but the reverse is not true because some valid A_t paths can correspond to traversals of SA that either consume several time the same source token or do not consume all source tokens. For instance, the sentence *these brown avocadoes brown are \$* belongs to the language of A_t, but cannot be produced by SA. Let's however consider the intersection $\{x\} \Cap G \Cap A_t$, where, with a slight abuse of notation, we have notated $\{x\}$ the "degenerate" automaton representing the sentence x, namely the automaton (with weights on all transitions equal to 1):



This is a *relaxation* of the true intersection, but one that we can represent through a W-SCFG, as we know from section 2.[10]

This being noted, we now move to the construction of the full intersection.

The full intersection We discussed in section 2 how to modify a synchronous grammar rule in order to produce the indexed rule scheme (2) in order to represent the bilateral intersection of the grammar with two automata. Let us redo that construction here, in the case of our example

[10] Note that, in the case of our very simple example, any target string that belongs to this relaxed intersection (which consists of the eight sentences *these* {*brown* | *totally corrupt*} {*avocadoes* | *lawyers*} *are* {*cooked* | *finished*}) actually belongs to the full intersection, as none of these sentences corresponds to a path in SA that violates the token-consumption constraint. More generally, it may often be the case in practice that the W-SCFG, by itself, provides enough "control" of the possible target sentences to prevent generation of sentences that would violate the token-consumption constraints, so that there may be little difference in practice between performing the relaxed intersection $\{x\} \Cap G \Cap A_t$ and performing the full intersection $\{x\} \Cap G \Cap L_{SA,x}$.

W-SCFG, of the target automaton represented on the bottom of Figure 1, and of the source automaton $\{x\}$.

The construction is then done in three steps:

$$
\begin{aligned}
{}_{s_0}\mathrm{NP}_{s_3} &\rightarrow {}_{s_0}\mathrm{ces}_{s_1}\ \ {}_{s_1}\mathrm{N}_{s_2}\ \ {}_{s_2}\mathrm{A}_{s_3}\ \ / \\
&\qquad \text{these}\ \ \mathrm{A}\ \ \mathrm{N} \\
{}^{t_0}\mathrm{NP}^{t_3} &\rightarrow \text{ces}\ \ \mathrm{N}\ \ \mathrm{A}\ \ / \\
&\qquad {}^{t_0}\text{these}^{t_1}\ \ {}^{t_1}\mathrm{A}^{t_2}\ \ {}^{t_2}\mathrm{N}^{t_3} \\
{}^{t_0}_{s_0}\mathrm{NP}^{t_3}_{s_3} &\rightarrow {}_{s_0}\mathrm{ces}_{s_1}\ \ {}^{t_2}_{s_1}\mathrm{N}^{t_3}_{s_2}\ \ {}^{t_1}_{s_2}\mathrm{A}^{t_2}_{s_3}\ \ / \\
&\qquad {}^{t_0}\text{these}^{t_1}\ \ {}^{t_1}_{s_2}\mathrm{A}^{t_2}_{s_3}\ \ {}^{t_2}_{s_1}\mathrm{N}^{t_3}_{s_2}
\end{aligned}
$$

In order to adapt that construction to the case where we want the intersection to be with a σ-automaton, what we need to do is to further specialize the nonterminals. Rather than specializing a nonterminal X in the form ${}^{t}_{s}X^{t'}_{s'}$, we specialize it in the form: ${}^{t}_{s}X^{t',\sigma}_{s'}$, where σ represents a set of source tokens that correspond to "collecting" the source tokens in the σ-automaton along a path connecting the states t and t'.[11]

We then proceed to define a new rule scheme associated to our rule, which is obtained as before in three steps, as follows.

$$
\begin{aligned}
{}_{s_0}\mathrm{NP}_{s_3} &\rightarrow {}_{s_0}\mathrm{ces}_{s_1}\ \ {}_{s_1}\mathrm{N}_{s_2}\ \ {}_{s_2}\mathrm{A}_{s_3}\ \ / \\
&\qquad \text{these}\ \ \mathrm{A}\ \ \mathrm{N} \\
{}^{t_0}\mathrm{NP}^{t_3,\sigma_{03}} &\rightarrow \text{ces}\ \ \mathrm{N}\ \ \mathrm{A}\ \ / \\
&\qquad {}^{t_0}\text{these}^{t_1,\sigma_{01}}\ \ {}^{t_1}\mathrm{A}^{t_2,\sigma_{12}}\ \ {}^{t_2}\mathrm{N}^{t_3,\sigma_{23}} \\
{}^{t_0}_{s_0}\mathrm{NP}^{t_3,\sigma_{03}}_{s_3} &\rightarrow {}_{s_0}\mathrm{ces}_{s_1}\ \ {}^{t_2}_{s_1}\mathrm{N}^{t_3,\sigma_{23}}_{s_2}\ \ {}^{t_1}_{s_2}\mathrm{A}^{t_2,\sigma_{12}}_{s_3}\ \ / \\
&\qquad {}^{t_0}\text{these}^{t_1,\sigma_{01}}\ \ {}^{t_1}_{s_2}\mathrm{A}^{t_2,\sigma_{12}}_{s_3}\ \ {}^{t_2}_{s_1}\mathrm{N}^{t_3,\sigma_{23}}_{s_2}
\end{aligned}
$$

The only difference with our previous technique is in the addition of the σ's to the top indices. Let us focus on the second step of the annotation process:

$$
\begin{aligned}
{}^{t_0}\mathrm{NP}^{t_3,\sigma_{03}} &\rightarrow \text{ces}\ \ \mathrm{N}\ \ \mathrm{A}\ \ / \\
&\qquad {}^{t_0}\text{these}^{t_1,\sigma_{01}}\ \ {}^{t_1}\mathrm{A}^{t_2,\sigma_{12}}\ \ {}^{t_2}\mathrm{N}^{t_3,\sigma_{23}}
\end{aligned}
$$

[11] To avoid a possible confusion, it is important to note right away that *σ is not necessarily related* to the tokens appearing between the positions s and s' in the source sentence (that is, between these states in the associated source automaton), but is defined solely in terms of the source tokens along the t, t' path. See example with "persons" and "people" below.

Conceptually, when instanciating this scheme, the t_i's may range over all possible states of the σ-automaton, and the σ_{ij} over all subsets of the source tokens, but under the following constraints: the RHS σ's (here $\sigma_{01}, \sigma_{12}, \sigma_{23}$) must be disjoint and their union must be equal to the σ on the LHS (here σ_{03}). Additionally, a σ associated with a target terminal (as σ_{01} here) must be equal to the token set associated to the transition that this terminal realizes between σ-automaton states (here, this means that σ_{01} must be equal to the token set $\{ces\}$ associated with the transition between t_0, t_1 labelled with 'these'). If we perform all these instantiations, compute their weights according to equation (3), and finally remove the indices associated with terminals in the rules (by adding the weights of the rules only differing by the indices of terminals, as done previously), we obtain a very large "raw" grammar, but one for which one can prove direct counterpart of **Fact 1**. Let us call, as before G' the raw W-SCFG obtained, its start symbol being ${}^{t_\#}_{s_\#}S^{t_\$,\sigma_{all}}_{s_\$}$, with σ_{all} the set of all source tokens in x.

Fact 2. *The synchronous language $L_{G'}$ associated with G' is equal to* $(\{x\}, L_{SA,x} \Cap L_{G,x})$.

The grammar that is obtained this way, despite correctly representing the intersection, contains a lot of useless rules, this being due to the fact that many nonterminals can not produce any output. The situation is wholly similar to the case of section 2, and the same bottom-up techniques can be used for activating nonterminals and rules bottom-up.

The algorithm is illustrated in Figure 2, where we have shown the result of the process of activating in turn the nonterminals (abbreviated by) N1, A1, A2, NP1, VP1, S1. As a consequence of these activations, the original grammar rule $NP \rightarrow ces\ N\ A\ /these\ A\ N$ (for instance) becomes instantiated as the rule:

$$
\begin{aligned}
&{}^{\#}_{0}\mathrm{NP}^{tcl,\{ces,avocats,marrons\}}_{3} \rightarrow \\
&{}_{0}\mathrm{ces}_{1}\ \ {}^{tcl2}_{1}\mathrm{N}^{tcl,\emptyset}_{2}\ \ {}^{h}_{2}\mathrm{A}^{tcl2,\{avocats,marrons\}}_{3}\ \ / \\
&{}^{\#}\text{these}^{h,\{ces\}}\ \ {}^{h}_{2}\mathrm{A}^{tcl2,\{avocats,marrons\}}_{3}\ \ {}^{tcl2}_{1}\mathrm{N}^{tcl,\emptyset}_{2}
\end{aligned}
$$

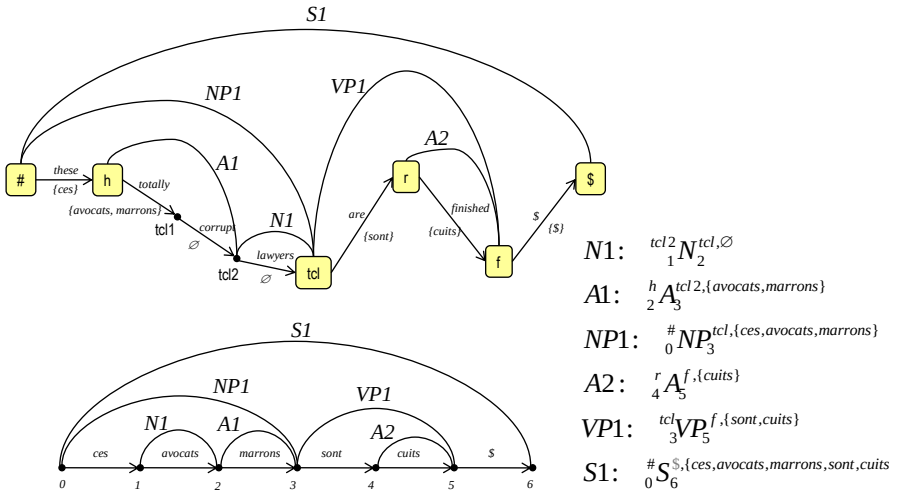


Figure 2: Building the intersection. The bottom of the figure shows some active non-terminals associated with the source sequence, at the top these same non-terminals associated with a sequence of transitions in the σ-automaton, corresponding to the target sequence *these totally corrupt lawyers are finished $.* To avoid cluttering the drawing, we have used the abbreviations shown on the right. Note that while A1 only spans *marrons* in the bottom chart, it is actually decorated with the source token set $\{avocats, marrons\}$: such a "disconnect" between the views that the W-SCFG and the σ-automaton have of the source tokens is not ruled out.

that is, after removal of the indices on terminals:

$$
\begin{aligned}
&{}^{\#}_{0}\mathrm{NP}^{tcl,\{ces,avocats,marrons\}}_{3} \rightarrow \\
&\text{ces} \quad {}^{tcl2}_{1}\mathrm{N}^{tcl,\emptyset}_{2} \quad {}^{h}_{2}\mathrm{A}^{tcl2,\{avocats,marrons\}}_{3} \quad / \\
&\text{these} \quad {}^{h}_{2}\mathrm{A}^{tcl2,\{avocats,marrons\}}_{3} \quad {}^{tcl2}_{1}\mathrm{N}^{tcl,\emptyset}_{2}
\end{aligned}
$$

Note that while the nonterminal ${}^{tcl2}_{1}\mathrm{N}^{tcl,\emptyset}_{2}$ by itself consumes no source token (it is associated with the empty token set), any actual use of this nonterminal (in this specific rule or possibly in some other rule using it) does require traversing the internal node $tcl2$ and therefore all the internal nodes "belonging" to the biphrase tcl (because otherwise the path from # to \$ would be disconnected); in particular this involves consuming all the tokens on the source side of tcl, including 'avocats'.[12]

Complexity considerations The bilateral intersection that we defined between a W-SCFG and two W-FSA's in section 2 can be shown to be of polynomial complexity in the sense that it takes polynomial time and space relative to the sum of the sizes of the two automata and of the grammar to construct the (reduced) intersected grammar G', *under the condition that the grammar right-hand sides have length bounded by a constant.*[13]

The situation here is different, because the construction of the intersection can in principle introduce nonterminals indexed not only by states of the automata, but also by arbitrary subsets of source tokens, and this may lead in extreme cases to an exponential number of rules. Such problems however can only happen in situations where, in a nonterminal ${}^{t}_{s}X^{t',\sigma}_{s'}$, the set σ is allowed to contain tokens that are "unrelated" to the token set $\{personnes\}$ appearing between s and s' in the source automaton. An illustration of such a situation is given by the following example. Suppose that the source sen-

[12] In particular there is no risk that a derivation relative to the intersected grammar generates a target containing two instances of 'lawyers', one associated to the expansion of ${}^{tcl2}_{1}\mathrm{N}^{tcl,\emptyset}_{2}$ and consuming no source token, and another one associated with a different nonterminal and consuming the source token 'avocats': this second instance would involve not traversing $tcl1$, which is impossible as soon as ${}^{tcl2}_{1}\mathrm{N}^{tcl,\emptyset}_{2}$ is used.

[13] If this condition is removed, and for the simpler case where the source (resp. target) automaton encodes a single sentence x (resp. y), (Satta and Peserico, 2005) have shown that the problem of deciding whether (x, y) is recognized by G is NP-hard relative to the sum of the sizes. A consequence is then that the grammar G' cannot be constructed in polynomial time unless $P = NP$.

tence contains the two tokens $personnes$ and $gens$ between positions $i, i+1$ and $j, j+1$ respectively, with i and j far from each other, that the phrase table contains the two phrase pairs $(personnes, persons)$ and $(gens, people)$, but that the synchronous grammar only contains the two rules $X \rightarrow personnes/people$ and $Y \rightarrow gens/persons$, with these phrases and rules exhausting the possibilities for translating $gens$ and $personnes$; Then the intersected grammar will contain such nonterminals as ${}^{t}_{i}X^{t',\{gens\}}_{i+1}$ and ${}^{r}_{j}Y^{r',\{personnes\}}_{j+1}$, where in the first case the token set $\{gens\}$ in the first nonterminal is unrelated to the tokens appearing between $i, i+1$, and similarly in the second case.

Without experimentation on real cases, it is impossible to say whether such phenomena would empirically lead to combinatorial explosion or whether the synchronous grammar would sufficiently constrain the phrase-base component (whose re-ordering capabilities are responsible *in fine* for the potential NP-hardness of the translation process) to avoid it. Another possible approach is to prevent *a priori* a possible combinatorial explosion by adding formal constraints to the intersection mechanism. One such constraint is the following: disallow introduction of ${}^{t}_{i}X^{t',\sigma}_{j}$ when the symmetric difference between σ and the set of tokens between positions i and j in the source sentence has cardinality larger than a small constant. Such a constraint could be interpreted as keeping the SCFG and phrase-base components "in sync", and would be better adapted to the spirit of our approach than limiting the amount of re-ordering permitted to the phrase-based component, which would contradict the reason for using a hierarchical component in the first place.

5 Conclusion

Intersecting hierarchical and phrase-based models of translation could allow to capitalize on complementarities between the two approaches. Thus, one might train the hierarchical component on corpora represented at the part-of-speech level (or at a level where lexical units are abstracted into some kind of classes) while the phrase-based component would focus on translation of lexical material. The present paper does not have the ambition to demonstrate that such an approach would improve translation performance, but only to provide some formal means for advancing towards that goal.

References

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

Berstel, Jean. 1979. *Transductions and Context-Free Languages*. Teubner, Stuttgart.

Blunsom, P. and M. Osborne. 2008. Probabilistic inference for machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 215–223. Association for Computational Linguistics. Slides downloaded.

Chiang, David. 2006. An introduction to synchronous grammars. `www.isi.edu/~chiang/papers/synchtut.pdf`, June.

Chiang, David. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, 33:201–228.

Koehn, Philipp, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *ACL*. The Association for Computer Linguistics.

Kumar, Shankar and William Byrne. 2005. Local phrase reordering models for statistical machine translation. In *Proc. HLT/EMNLP*.

Lang, Bernard. 1994. Recognition can be harder than parsing. *Computational Intelligence*, 10:486–494.

Mohri, Mehryar, Fernando Pereira, and Michael Riley. 1996. Weighted automata in text and speech processing. In *ECAI-96 Workshop on Extended Finite State Models of Language*.

Mohri, Mehryar. 2002. Generic epsilon-removal and input epsilon-normalization algorithms for weighted transducers. *International Journal of Foundations of Computer Science*, 13:129–143.

Och, Franz Josef and Hermann Ney. 2004. The alignment template approach to statistical machine translation. *Comput. Linguist.*, 30(4):417–449.

Satta, Giorgio and Enoch Peserico. 2005. Some computational complexity results for synchronous context-free grammars. In *HLT '05: Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 803–810, Morristown, NJ, USA. Association for Computational Linguistics.

Satta, Giorgio. 2008. Translation algorithms by means of language intersection. Submitted. `www.dei.unipd.it/~satta/publ/paper/inters.pdf`.