On the complexity of alignment problems in two synchronous grammar formalisms

Anders Søgaard[*]
Center for Language Technology
University of Copenhagen
soegaard@hum.ku.dk

Abstract

The alignment problem for synchronous grammars in its unrestricted form, i.e. whether for a grammar and a string pair the grammar induces an alignment of the two strings, reduces to the universal recognition problem, but restrictions may be imposed on the alignment sought, e.g. alignments may be $1:1$, island-free or sure-possible sorted. The complexities of 15 restricted alignment problems in two very different synchronous grammar formalisms of syntax-based machine translation, inversion transduction grammars (ITGs) (Wu, 1997) and a restricted form of range concatenation grammars ((2,2)-BRCGs) (Søgaard, 2008), are investigated. The universal recognition problems, and therefore also the unrestricted alignment problems, of both formalisms can be solved in time $\mathcal{O}(n^6|G|)$. The complexities of the restricted alignment problems differ significantly, however.

1 Introduction

The synchronous grammar formalisms used in syntax-based machine translation typically induce alignments by aligning all words that are recognized simultaneously (Wu, 1997; Zhang and Gildea, 2004). On a par with weak and strong generative capacity, it is thus possible to talk about the alignment capacity of those formalisms. In this paper, two synchronous grammar formalisms are discussed, inversion transduction grammars (ITGs) (Wu, 1997) and two-variable binary bottom-up non-erasing range concatenation grammars ((2,2)-BRCGs) (Søgaard, 2008). It is known that ITGs do not induce the class of inside-out alignments discussed in Wu (1997). Another class that ITGs do not induce is that of alignments with discontinuous translation units (Søgaard, 2008). Søgaard (2008), on the other hand, shows that the alignments induced by (2,2)-BRCGs are closed under union, i.e. (2,2)-BRCGs induce all possible alignments.

The universal recognition problems of both ITGs and (2,2)-BRCGs can be solved in time $\mathcal{O}(n^6|G|)$. This may come as a surprise, as ITGs restrict the alignment search space considerably, while (2,2)-BRCGs do not. In the context of the NP-hardness of decoding in statistical machine translation (Knight, 1999; Udupa and Maji, 2006), it is natural to ask why the universal recognition problem of (2,2)-BRCGs isn't NP-hard? How can (2,2)-BRCGs induce all possible alignments and still avoid NP-hardness? This paper bridges the gap between these results and shows that when alignments are restricted to be $1:1$, island-free or sure-possible sorted (see below), or all combinations thereof, the alignment problem of (2,2)-BRCGs *is* NP-hard. (2,2)-BRCGs in a sense avoid NP-hardness by giving up control over global properties of alignments, e.g. any pair of words may be aligned multiple times in a derivation.

[*] This work was done while the first author was a Senior Researcher at the Dpt. of Linguistics, University of Potsdam, supported by the German Research Foundation in the Emmy Noether project *Ptolemaios* on grammar learning from parallel corpora; and while he was a Postdoctoral Researcher at the ISV Computational Linguistics Group, Copenhagen Business School, supported by the Danish Research Foundation in the project *Efficient syntax- and semantics-based machine translation*.

Proceedings of SSST-3, Third Workshop on Syntax and Structure in Statistical Translation, pages 60–68,
Boulder, Colorado, June 2009. ©2009 Association for Computational Linguistics

The alignment structures induced by synchronous grammars in syntax-based machine translation have the following property: If an alignment structure includes alignments $v|v'$, $v|w'$ and $w|w'$, it also includes the alignment $w|v'$, where w, w', v, v' are word instances.[1] This follows from the fact that only words that are recognized simultanously, are aligned. Otherwise alignment structures are just a binary symmetric relation on two strings, a source and a target string, such that two words in the source, resp. target string, cannot be aligned. Maximally connected subgraphs (ignoring precedence edges) are called translation units.

The alignment problem can be formulated this way (with s, s' source and target sentence, resp.):

INSTANCE: $G, \langle s, s' \rangle$.
QUESTION: Does G induce an alignment on $\langle s, s' \rangle$?

The alignment problem in its unrestricted form reduces to the universal recognition problem (Barton et al., 1987), i.e. whether for a grammar G and a string pair $\langle s, s' \rangle$ it holds that $\langle s, s' \rangle \in L(G)$? Of course the alignment may in this case be empty or partial. Both ITGs and (2,2)-BRCGs permit unaligned nodes.

This paper investigates the complexity of restricted versions of the alignment problem for ITGs and (2,2)-BRCGs. A simple example, which can be solved in linear time for both formalisms, is the alignment problem wrt. alignments that consist of a single translation unit including all source and target words. It may be formulated this way:

INSTANCE: $G, \langle s, s' \rangle$.
QUESTION: Does G induce an alignment that consists of a single translation unit with no unaligned words on $\langle s, s' \rangle$?

This can be solved for ITGs by checking if there is a production rule that introduces all the words in the right order such that:[2]

- The LHS nonterminal symbol (possibly suffixed by the empty string ϵ) can be derived from the start symbol.

- The empty string ϵ can be derived from all RHS nonterminal symbols.

The only difference for (2,2)-BRCGs is that production rules are typically referred to as clauses in the range concatenation grammar literature.

This paper considers some more complex examples; namely, the alignment problems wrt. $1:1$-alignments, (source-side and/or target-side) island-free alignments and sure-possible sorted alignments. The formal definitions of the three properties are as follows:

Definition 1.1. An alignment structure for a string pair $\langle w_1 \ldots w_n, v_1 \ldots v_m \rangle$ is a graph $D = \langle V, E \rangle$ where $V = V_s : \{w_1, \ldots, w_n\} \cup V_t : \{v_1, \ldots, v_m\}$ and $E = E_s : \{w_i \prec w_j \mid i < j\} \cup E_t : \{v_i \prec v_j \mid i < j\} \cup A$ where $A \subseteq V_s \times V_t$. If $(w_i, v_j) \in A$, also written $w_i|v_j$, w_i is said to be aligned to v_j, and vice versa. An alignment structure is said to be *wellformed* iff for all $w_i, w_j, v_{i'}, v_{j'}$ it holds that if $w_i|v_{i'}, w_i|v_{j'}$ and $w_j|v_{i'}$ are aligned then so are $w_j|v_{j'}$. An alignment structure is said to be $1:1$ iff no word occurs in two distinct tuples in A. An alignment structure is said to be *island-free* iff all words in V occur in some tuple in A; it is said to be source-side, resp. target-side, island-free if all words in V_s, resp. V_t, occur in some tuple in A. The set of alignments is divided into sure and possible alignments, i.e. $A = S \cup P$ (in most cases $P = \emptyset$). An alignment structure is said to be *sure-possible sorted* iff if it holds that $(w_i, v_{j'}) \in S$ then for all $w_j, v_{i'}$ neither $(w_i, v_{i'}) \in P$ nor $(w_j, v_{j'}) \in P$ holds; similarly, if it holds that $(w_i, v_{j'}) \in P$ then for all $w_j, v_{i'}$ neither $(w_i, v_{i'}) \in S$ nor $(w_j, v_{j'}) \in S$ holds.

The precedence relations in E are not important for any of our definitions, but are important for meaningful interpretation of alignment structures. Note that synchronous grammars are guaranteed to induce wellformed alignment structures. Some brief motivation for the properties singled out:

[1] $w|w'$ is our short hand notation for saying that w, a word in the source string, and w', a word in the target string, have been aligned. In the formal definition of alignments below, it is said that $w \in V_s$ (w is a word in the source string), $w' \in V_t$ (w' is a word in the target string) and $(w, w') \in A$, i.e. w is aligned to w', and vice versa. Alignments are bidirectional in what follows.

[2] In fact in normal form ITGs, we can simply check if there is a production rule with the start symbol in the LHS that introduces all the words in the right order, since all production rules with nonterminal symbols in the RHS are branching and contain no terminal symbols.

Result	1 : 1	IF(s)	IF(t)	SP	ITGs	(2,2)-BRCGs
(1)	✓				$\mathcal{O}(n^6\|G\|)$	NP-complete
(2)		✓			$\mathcal{O}(n^6\|G\|)$	NP-complete
(3)			✓		$\mathcal{O}(n^6\|G\|)$	NP-complete
(4)				✓	$\mathcal{O}(n^6\|G\|)$	NP-complete
(5)	✓	✓			$\mathcal{O}(n^6\|G\|)$	NP-complete
(6)		✓	✓		$\mathcal{O}(n^6\|G\|)$	NP-complete
(7)			✓	✓	$\mathcal{O}(n^6\|G\|)$	NP-complete
(8)	✓		✓		$\mathcal{O}(n^6\|G\|)$	NP-complete
(9)		✓		✓	$\mathcal{O}(n^6\|G\|)$	NP-complete
(10)	✓			✓	$\mathcal{O}(n^6\|G\|)$	NP-complete
(11)	✓	✓	✓		$\mathcal{O}(n^6\|G\|)$	NP-complete
(12)		✓	✓	✓	$\mathcal{O}(n^6\|G\|)$	NP-complete
(13)	✓		✓	✓	$\mathcal{O}(n^6\|G\|)$	NP-complete
(14)	✓	✓		✓	$\mathcal{O}(n^6\|G\|)$	NP-complete
(15)	✓	✓	✓	✓	$\mathcal{O}(n^6\|G\|)$	NP-complete

Figure 1: The complexity of restricted alignment problems for ITGs and (2,2)-BRCGs.

- $1:1$-alignments have been argued to be adequate by Melamed (1999) and elsewhere, and it may therefore be useful to know if a grammar extracted from a parallel corpus produces $1:1$-alignments for a finite set of sentence pairs.

- Island-free alignments are interesting to the extent that unaligned nodes increase the chance of translation errors. An island threshold may for instance be used to rule out risky translations.

- The notion of sure-possible sorted alignments is more unusual, but can, for instance, be used to check if the use of possible alignments is consistently triggered by words that are hard to align.

The results for all cross-classifications of the four properties – $1 : 1$, source-side island-free (IF(s)), target-side island-free (IF(t)) and sure-possible sorted (SP) – are presented in the table in Figure 1.[3] Note that all ($2^4 - 1 = 15$) combinations of the four properties lead to NP-hard alignment problems for (2,2)-BRCGs. Consequently, while the unrestricted alignment problem for (2,2)-BRCGs can be solved in $\mathcal{O}(n^6|G|)$, the alignment problem turns NP-hard as soon as restrictions are put on the alignments sought. So the extra expressivity of (2,2)-BRCGs in a way comes at the expense of control over the kind of alignments obtained.

On the structure of the paper: Sect. 2 and 3 briefly introduce, resp., ITGs and (2,2)-BRCGs. Sect. 4 presents three NP-hardness proofs from which the 15 results in Figure 1 can be derived. The three proofs are based on reconstructions of the Hamilton circuit problem, the 3SAT problem and the vertex cover problem (Garey and Johnson, 1979).

2 Inversion transduction grammars

Inversion transduction grammars (ITGs) (Wu, 1997) are a notational variant of binary syntax-directed translation schemas (Aho and Ullman, 1972) and are usually presented with a normal form:

$$\begin{array}{lcl} A & \rightarrow & [BC] \\ A & \rightarrow & \langle BC \rangle \\ A & \rightarrow & e \mid f \\ A & \rightarrow & e \mid \epsilon \\ A & \rightarrow & \epsilon \mid f \end{array}$$

where $A, B, C \in N$ and $e, f \in T$. The first production rule, intuitively, says that the subtree $[[]_B[]_C]_A$ in the source language translates into

[3] One of our reviewers remarks that the Figure 1 is 'artificially blown up', since all combinations have the same complexity. It cannot really be left out, however. The numbers in the figure's left-most column serves as a reference in the proofs below. Since the 15 results derive from only four proofs, it is convenient to have a short-hand notation for the decision problems.

a subtree $[[]_B[]_C]_A$, whereas the second production rule inverts the order in the target language, i.e. $[[]_C[]_B]_A$. The universal recognition problem of ITGs can be solved in time $\mathcal{O}(n^6|G|)$ by a CYK-style parsing algorithm with two charts.

Figure 1 tells us that all the restricted alignment problems listed can be solved in time $\mathcal{O}(n^6|G|)$. The explanation is simple. It can be read off from the syntactic form of the production rules in ITGs whether they introduce $1:1$-alignments, island-free alignments or sure-possible sorted alignments. Note that normal form ITGs only induce $1:1$-alignments.

Consider, for example, the following grammar, not in normal form for brevity:

$$\begin{array}{llll} (1) & S & \rightarrow & \langle ASB \rangle \mid \langle AB \rangle \\ (2) & A & \rightarrow & a \mid a \\ (3) & A & \rightarrow & a \mid \epsilon \\ (4) & B & \rightarrow & b \mid b \end{array}$$

Note that this grammar recognizes the translation $\{\langle a^n b^n, b^n a^m \rangle \mid n \geq m\}$. To check if for a string pair $\langle w_1 \dots w_n, v_1 \dots v_m \rangle$ this grammar induces an island-free alignment, simply remove production rule (3). It holds that only strings in the sublanguage $\{\langle a^n b^n, b^n a^n \rangle \mid n \geq 1\}$ induce island-free alignments. Similarly, to check if the grammar induces source-side island-free alignments for string pairs, no production rules will have to be removed.

3 Two-variable binary bottom-up non-erasing range concatenation grammars

(2,2)-BRCGs are *positive* RCGs (Boullier, 1998) with binary start predicate names, i.e. $\rho(S) = 2$. In RCG, predicates can be negated (for complementation), and the start predicate name is typically unary. The definition is changed only for aesthetic reasons; a positive RCG with a binary start predicate name S is turned into a positive RCG with a unary start predicate name S' simply by adding a clause $S'(X_1X_2) \rightarrow S(X_1, X_2)$.

A positive RCG is a 5-tuple $G = \langle N, T, V, P, S \rangle$. N is a finite set of predicate names with an arity function ρ: $N \rightarrow \mathbb{N}$, T and V are finite sets of, resp., terminal and variables. P is a finite set of clauses of the form $\psi_0 \rightarrow \psi_1 \dots \psi_m$, where each of the $\psi_i, 0 \leq i \leq m$, is a predicate of the form $A(\alpha_1, \dots, \alpha_{\rho(A)})$. Each $\alpha_j \in (T \cup V)^*, 1 \leq j \leq \rho(A)$, is an argument. $S \in N$ is the start predicate name with $\rho(S) = 2$.

Note that the order of RHS predicates in a clause is of no importance. Three subclasses of RCGs are introduced for further reference: An RCG $G = \langle N, T, V, P, S \rangle$ is *simple* iff for all $c \in P$, it holds that no variable X occurs more than once in the LHS of c, and if X occurs in the LHS then it occurs exactly once in the RHS, and each argument in the RHS of c contains exactly one variable. An RCG $G = \langle N, T, V, P, S \rangle$ is a *k-RCG* iff for all $A \in N, \rho(A) \leq k$. Finally, an RCG $G = \langle N, T, V, P, S \rangle$ is said to be *bottom-up non-erasing* iff for all $c \in P$ all variables that occur in the RHS of c also occur in its LHS.

A positive RCG is a (2,2)-BRCG iff it is a 2-RCG, if an argument of the LHS predicate contains at most two variables, and if it is bottom-up non-erasing.

The language of a (2,2)-BRCG is based on the notion of *range*. For a string pair $\langle w_1 \dots w_n, v_{n+2} \dots v_{n+1+m} \rangle$ a range is a pair of indices $\langle i, j \rangle$ with $0 \leq i \leq j \leq n$ or $n < i \leq j \leq n+1+m$, i.e. a string span, which denotes a substring $w_{i+1} \dots w_j$ in the source string or a substring $v_{i+1} \dots v_j$ in the target string. Only consequtive ranges can be concatenated into new ranges. Terminals, variables and arguments in a clause are bound to ranges by a substitution mechanism. An *instantiated* clause is a clause in which variables and arguments are consistently replaced by ranges; its components are *instantiated predicates*. For example $A(\langle g \dots h \rangle, \langle i \dots j \rangle) \rightarrow B(\langle g \dots h \rangle, \langle i+1 \dots j-1 \rangle)$ is an instantiation of the clause $A(X_1, aY_1b) \rightarrow B(X_1, Y_1)$ if the target string is such that $v_{i+1} = a$ and $v_j = b$. A *derive* relation $\Longrightarrow$ is defined on strings of instantiated predicates. If an instantiated predicate is the LHS of some instantiated clause, it can be replaced by the RHS of that instantiated clause. The language of a (2,2)-BRCG $G = \langle N, T, V, P, S \rangle$ is the set $L(G) = \{\langle w_1 \dots w_n, v_{n+2} \dots v_{n+1+m} \rangle \mid S(\langle 0, n \rangle, \langle n+1, n+1+m \rangle) \overset{*}{\Longrightarrow} \epsilon\}$, i.e. an input string pair $\langle w_1 \dots w_n, v_{n+2} \dots v_{n+1+m} \rangle$ is recognized iff the empty string can be derived from $S(\langle 0, n \rangle, \langle n+1, n+1+m \rangle)$.

It is not difficult to see that ITGs are also (2,2)-BRCGs. The left column is ITG production rules;

the right column their translations in simple (2,2)-BRCGs.

$$\begin{array}{r|l}
A \rightarrow [BC] & A(X_1X_2, Y_1Y_2) \rightarrow B(X_1, Y_1)C(X_2, Y_2) \\
A \rightarrow \langle BC \rangle & A(X_1X_2, Y_1Y_2) \rightarrow B(X_1, Y_2)C(X_2, Y_1) \\
A \rightarrow e \mid f & A(e, f) \rightarrow \epsilon \\
A \rightarrow e \mid \epsilon & A(e, \epsilon) \rightarrow \epsilon \\
A \rightarrow \epsilon \mid f & A(\epsilon, f) \rightarrow \epsilon
\end{array}$$

Consequently, (2,2)-BRCGs recognize all translations recognized by ITGs. In fact the inclusion is strict, as shown in Søgaard (2008). The universal recognition problem of (2,2)-BRCGs can be solved in time $\mathcal{O}(n^6|G|)$ by the CYK-style parsing algorithm presented in Søgaard (2008).

Example 3.1. Consider the (2,2)-BRCG $G = \langle\{S_s, S_0, S_0', S_1, S_1', A, B, C, D\}, \{a, b, c, d\}, \{X_1, X_2, Y_1, Y_2\}, P, S_s\rangle$ with P the following set of clauses:

$$\begin{array}{lrcl}
\text{(1)} & S_s(X_1, Y_1) & \rightarrow & S_0(X_1, Y_1)S_0'(X_1, Y_1) \\
\text{(2)} & S_0(X_1X_2, Y_1) & \rightarrow & S_1(X_1, Y_1)D(X_2) \\
\text{(3)} & S_1(aX_1c, abY_1) & \rightarrow & S_1(X_1, Y_1) \\
\text{(4)} & S_1(X_1, Y_1Y_2) & \rightarrow & B(X_1)C(Y_1)D(Y_2) \\
\text{(5)} & S_0'(X_1X_2, Y_1) & \rightarrow & S_1'(X_2, Y_1)A(X_1) \\
\text{(6)} & S_1'(bX_1d, Y_1cd) & \rightarrow & S_1'(X_1, Y_1) \\
\text{(7)} & S_1'(X_1, Y_1Y_2) & \rightarrow & C(X_1)A(Y_1)B(Y_2) \\
\text{(8)} & A(aX_1) & \rightarrow & A(X_1) \\
\text{(9)} & A(\epsilon) & \rightarrow & \epsilon \\
\text{(10)} & B(bX_1) & \rightarrow & B(X_1) \\
\text{(11)} & B(\epsilon) & \rightarrow & \epsilon \\
\text{(12)} & C(cX_1) & \rightarrow & C(X_1) \\
\text{(13)} & C(\epsilon) & \rightarrow & \epsilon \\
\text{(14)} & D(dX_1) & \rightarrow & D(X_1) \\
\text{(15)} & D(\epsilon) & \rightarrow & \epsilon
\end{array}$$

The string pair $\langle abbcdd, abcdcd \rangle$ is derived:

$$\begin{array}{lll}
 & S_s(\langle 0,6\rangle, \langle 0,6\rangle) & \\
\Longrightarrow & S_0(\langle 0,6\rangle, \langle 0,6\rangle)S_0'(\langle 0,6\rangle, \langle 0,6\rangle) & \text{(1)} \\
\Longrightarrow & S_1(\langle 0,4\rangle, \langle 0,6\rangle)D(\langle 4,6\rangle) & \text{(2)} \\
 & S_0'(\langle 0,6\rangle, \langle 0,6\rangle) & \\
\Longrightarrow & S_1(\langle 0,4\rangle, \langle 0,6\rangle)S_0'(\langle 0,6\rangle, \langle 0,6\rangle) & \text{(14–15)} \\
\Longrightarrow & S_1(\langle 1,3\rangle, \langle 2,6\rangle)S_0'(\langle 0,6\rangle, \langle 0,6\rangle) & \text{(3)} \\
\Longrightarrow & B(\langle 1,3\rangle)C(\langle 2,4\rangle)D(\langle 4,6\rangle) & \text{(4)} \\
 & S_0'(\langle 0,6\rangle, \langle 0,6\rangle) & \\
\Longrightarrow & S_0'(\langle 0,6\rangle, \langle 0,6\rangle) & \text{(10–15)} \\
\Longrightarrow & S_1'(\langle 1,6\rangle, \langle 0,6\rangle)A(\langle 0,1\rangle) & \text{(5)} \\
\Longrightarrow & S_1'(\langle 1,6\rangle, \langle 0,6\rangle) & \text{(8–9)} \\
\Longrightarrow & S_1'(\langle 2,5\rangle, \langle 0,4\rangle) & \text{(6)} \\
\Longrightarrow & S_1'(\langle 3,4\rangle, \langle 0,2\rangle) & \text{(6)} \\
\Longrightarrow & C(\langle 3,4\rangle)A(\langle 0,1\rangle)B(\langle 1,2\rangle) & \text{(7)} \\
\Longrightarrow & \epsilon & \text{(8–13)}
\end{array}$$

Note that $L(G) = \{\langle a^n b^m c^n d^m, (ab)^n (cd)^m \rangle \mid m, n \geq 0\}$.

4 Results

4.1 Checking island-freeness and sure-possible sortedness

One possible way to check for island-freeness and sure-possible sortedness in the context of (2,2)-BRCGs is to augment the CYK-style algorithm with feature structures (Boolean vectors); all there is needed, e.g. to check sure-possible sortedness, is to pair up the nonterminals inserted in the cells of the chart with a flat feature structure of the form:

$$\begin{bmatrix} \text{SURE}_1 \; val_1 \\ \vdots \\ \text{SURE}_n \; val_n \end{bmatrix}$$

where n is the length of the source, resp. target, string in the source, resp. target, chart, and $1 \leq i \leq n : val_i \in \{+, -\}$. When a clause applies that induces a sure alignment between a word w_i and some word in the target, resp. source, string, the attribute SURE_i is assigned the value +; if a possible alignment is induced between w_i and another word, the attribute is assigned the value -. This can all be done in constant time. A copying clause now checks if the appropriate nonterminals have been inserted in the cells in question, but also that the associated feature structures unify. This can be done in linear time. Feature structures can be used the same way to record what words have been aligned to check island-freeness. Unfortunately, this technique does not guarantee polynomial runtime. Note that there can be 2^n many distinct feature structures for each nonterminal symbol in a chart. Consequently, whereas the size of a cell in the standard CYK algorithm is bounded by $|N|$, and in synchronous parsing by $|N| \times (2n-1)$,[4] the cells are now of exponential size in the worst case.

The following three sections provide three NP-hardness proofs: The first shows that the alignment

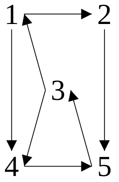
[4] The indices used to check that two nonterminals are derived simultaneously (Søgaard, 2008) mean that it may be necessary within a cell in the source, resp. target, chart to keep track of multiple tuples with the same nonterminals. In the worst case, there is a nonterminal for each span in the target, resp. source, chart, i.e. $2n - 1$ many.

problem wrt. $1:1$-alignments is NP-hard for (2,2)-BRCGs and goes by reduction of the Hamilton circuit problem for directed connected graphs. The second shows that the alignment problem wrt. source- or target-side island-free and sure-possible sorted alignments is NP-hard for (2,2)-BRCGs and goes by 3SAT reduction. The third proof is more general and goes by reduction of the vertex cover problem. All three formal decision problems are discussed in detail in Garey and Johnson (1979). All 15 results in Figure 1 are derived from modifications of these proofs.

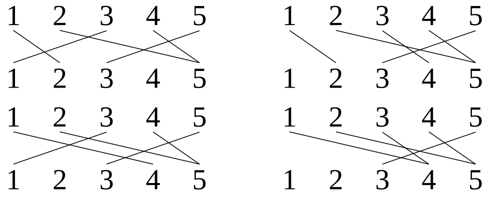
4.2 NP-hardness of the $1:1$ restriction for (2,2)-BRCGs

Theorem 4.1. *The alignment problem wrt.* $1:1$-*alignments is NP-hard for (2,2)-BRCGs.*

Proof. An instance of the Hamilton circuit problem for directed connected graphs is simply a directed connected graph $G = \langle V, E\rangle$ and the problem is whether there is a path that visits each vertex exactly once and returns to its starting point? Consider, for instance, the directed connected graph:



It is easy to see that there is no path in this case that visits each vertex exactly once and returns to its starting point. The intuition behind our reconstruction of the Hamilton circuit problem for directed connected graphs is to check this via alignments between a sequence of all the vertices in the graph and itself. The grammar permits an alignment between two words $w|v$ if there is a directed edge between the corresponding nodes in the graph, e.g. $(w, v) \in E$. The alignment structures below depict the possible alignments induced by the grammar obtained by the translation described below for our example graph:



Since no alignment above is $1:1$, there is no solution to the corresponding circuit problem. The translation goes as follows:

- Add a rule $S(X_1, Y_1) \rightarrow \{S_{v_i}(X_1, Y_1) \mid \forall v_i.\exists v_j.(v_i, v_j) \in E\}$.
- For each edge $(v_i, v_j) \in E$ add a rule $S_{v_i}(X_1 v_i X_2, Y_1 v_j Y_2) \rightarrow \top(X_1)\top(X_2)\top(X_3)\top(X_4)$.[5]
- For all $v_i \in V$ add a rule $\top(v_i X_1) \rightarrow \top(X_1)$.
- Add a rule $\top(\epsilon) \rightarrow \epsilon$.

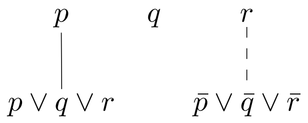
The grammar ensures source-side island-freeness, and therefore if there exists a $1:1$-alignment of any linearization of V and itself, by connectivity of the input graph, there is a solution to the Hamilton circuit problem for directed connected graphs.

□

4.3 NP-hardness of island-freeness and sure-possible sortedness for (2,2)-BRCGs

Theorem 4.2. *The alignment problem wrt. target-side island-free and sure-possible sorted alignments is NP-hard for (2,2)-BRCGs.*

Proof. An instance of the 3SAT problem is a propositional logic formula ϕ that is a conjunction of clauses of three literals connected by disjunctions, and the problem whether this formula is satisfiable, i.e. has a model? Say $\phi = p \vee q \vee r \wedge \bar{p} \vee \bar{q} \vee \bar{r}$. For our reconstruction, we use the propositional variables in ϕ as source string, and ϕ itself with $\wedge$'s omitted and conjuncts as words as the target string. One of the representations of a solution constructed by the translation described below is the following alignment structure:



Solid lines are sure alignments; dotted lines are possible alignments. The intuition is to use sure alignments to encode true assignments, and possible alignments as false assignments. The alignment

[5] $\top$ is an arbitrary predicate name chosen to reflect the fact that all possible substrings over the vocabulary are recognized by the $\top$ predicates.

above thus corresponds to the model $\{p, \bar{r}\}$, which clearly satisfies ϕ.

For the translation, assume that each 3SAT instance, over a set of propositional variables $\mathbb{PROP}$, consists of a set of clauses $c_1 \ldots c_m$ that are sets of literals of size 3. For any literal l_j, if $l_j = \bar{p_j}$ then $pos(l_j) = p_j$ and $lit(l_j) = -$; and if $l_j = p_j$ then $pos(l_j) = p_j$ and $lit(l_j) = +$. If l_j is a literal in c_i, we write $l_j \in c_i$. First add the following four clauses:

$$\begin{array}{rcl} S_s(X_1, Y_1) & \rightarrow & S_s(X_1, Y_1) \mid S_p(X_1, Y_1) \\ S_p(X_1, Y_1) & \rightarrow & S_s(X_1, Y_1) \mid S_p(X_1, Y_1) \end{array}$$

- If $l_j \in c_i$ and $lit(l_j) = -$, add $S_p(X_1 pos(l_j) X_2, Y_1 c_i Y_2) \rightarrow \top(X_1)\top(X_2)$ $\top(Y_1)\top(Y_2)$.
- If $l_j \in c_i$ and $lit(l_j) = +$, add $S_s(X_1 pos(l_j) X_2, Y_1 c_i Y_2) \rightarrow \top(X_1)\top(X_2)$ $\top(Y_1)\top(Y_2)$.
- For all p_j, add $\top(p_j X_1) \rightarrow \top(X_1)$.
- For all c_i, add $\top(c_i X_1) \rightarrow \top(X_1)$.
- Add a rule $\top(\epsilon) \rightarrow \epsilon$.

It is easy to see that the first rule adds at most $7m$ clauses, which for the largest non-redundant formulas equals $7((2|\mathbb{PROP}|)^3)$. The second rule adds at most $2|\mathbb{PROP}|$ clauses; and the third at most $m \leq (2|\mathbb{PROP}|)^3$ clauses. It is also easy to see that the grammar induces a target-side island-free, sure-possible sorted alignment if and only if the 3SAT instance is satisfiable. Note that the grammar does not guarantee that all induced alignments are target-side island-free. Nothing, in other words, corresponds to conjunctions in our reconstruction. This is not necessary as long as there is at least one target-side island-free alignment that is induced. □

Note that the proof also applies in the case where it is the source side that is required to be island-free. All needed is to make the source string the target string, and vice versa. Note also that the proof can be modified for the case where both sides are island-free: Just add a dummy symbol to the clause side and allow (or force) all propositional variables to be aligned to this dummy symbol. Consequently, if there is a target-side (clause-side) island-free alignment there is also an island-free alignment. Reversely, if there is an island-free alignment there is also a target-side island-free alignment of the string pair in question.

Note also that a more general proof can be obtained by introducing a clause, similar to the clause introduced in the first bullet point of the Hamilton circuit reduction in the proof of Theorem 4.1: $S(X_1, Y_1) \rightarrow \{S_{c_i}(X_1, Y_1) \mid 1 \leq i \leq m\}$. The four rules used to change between sure and possible alignments then of course need to be copied out for all S_{c_i} predicates, and the LHS predicates, except $\top$, of the other clauses must be properly subscripted. Now the grammar enforces target-side island-freeness, and sure-possible sortedness is the only restriction needed on alignments. Consequently, this reduction proves (4) that the alignment problem wrt. sure-possible sortedness is NP-hard for (2,2)-BRCGs.

4.4 NP-hardness of island-freeness for (2,2)-BRCGs

Theorem 4.3. *The alignment problem wrt. island-free alignments is NP-hard for (2,2)-BRCGs.*

Proof. An instance of the vertex problem is a graph $D = \langle V, E \rangle$ and an integer k, and the problem whether there exists a vertex cover of D of size k? Say $D = \langle V = \{a, b, c, d\}, E = \{(a, c), (b, c), (b, d), (c, d)\}\rangle$ and $k = 2$. The translation described below constructs a sentence pair

$$\langle \rho_1\rho_2\rho_3\rho_4 uu\delta\delta\delta\delta, aaaabbbbccccdddd\rangle$$

for this instance, and a (2,2)-BRCG with the clauses in Figure 2. Note that there are four kinds of clauses:

- A clause with an S predicate in the LHS. In general, there will be one such clause in the grammar constructed for any instance of the vertex cover problem.
- 8 clauses with ρ_i predicates in the LHS. In general, there will be $2|E|$ many clauses of this form in the grammars.
- 8 clauses with U^i predicates in the LHS. In general, there will be $|V| \times (|V| - k)$ many clauses of this form in the grammars.

- 16 clauses with δ^1 predicates in the LHS. In general, there will be $(|E| \times |V| - |E| - |E| \times (|V| - k)) \times |V|$ many clauses of this form in the grammars.

For an instance $\langle D = \langle V, E \rangle, k \rangle$, the translation function in general constructs the following clauses:

$$\begin{array}{lll} S(X_1, Y_1) & \rightarrow & \{\rho_i(X_1, Y_1) \mid 1 \leq i \leq |E|\} \cup \\ & & \{U^{|V|-k}(X_1, Y_1)\} \cup \\ & & \{\delta^{|E|\times|V|-|E|-|E|\times(|V|-k)}(X_1, Y_1)\} \end{array}$$

and for all $1 \leq i \leq |E|$ iff $e_i \in E = (e, e')$:

$$\begin{array}{lll} \rho_i(X_1 \rho_i X_2, Y_1 e Y_2) & \rightarrow & \top(X_1)\top(X_2)\top(Y_1)\top(Y_2) \\ \rho_i(X_1 \rho_i X_2, Y_1 e' Y_2) & \rightarrow & \top(X_1)\top(X_2)\top(Y_1)\top(Y_2) \end{array}$$

For all $2 \leq i \leq |V| - k$ and for all $v \in V$:

$$\begin{array}{lll} U^i(X_1 U X_2, Y_1 v \ldots v Y_2) & \rightarrow & U^{i-1}(X_1, Y_1) \\ & & \top(X_2)\top(Y_2) \end{array}$$

where $|v \ldots v| = |E|$. For the case U^1, add the clauses for all $v \in V$:

$$\begin{array}{lll} U^1(X_1 U X_2, Y_1 v \ldots v Y_2) & \rightarrow & \top(X_1)\top(Y_1) \\ & & \top(X_2)\top(Y_2) \end{array}$$

The string pair is constructed this way:

$$\begin{array}{c} \langle \rho_1 \ldots \rho_{|E|} U_1 \ldots U_{|V|-k} \\ \delta_1 \ldots \delta_{|E|\times|V|-|E|-|E|\times(|V|-k)}, \sigma \rangle \end{array}$$

Finally, for all words w in this string pair, add:

$$\top(w X_1) \quad \rightarrow \quad \top(X_1)$$

Since this translation is obviously polynomial, it follows that the alignment problem wrt. island-free alignments for (2,2)-BRCGs is NP-hard. □

Note that the proof also applies if only the source, resp. target, side is required to be island-free, since the grammar restricts the alignments in a way such that if one side is island-free then so is the other side. This gives us results (2) and (3).

It is not difficult to see either that it is possible to convert the grammar into a grammar that induces $1:1$-alignments. This gives us results (5), (8) and (11). Of course by the observation that all the grammars only use sure alignments, it follows that the alignment problems in (7), (9–10) and (12–15) are also NP-hard.

5 Conclusion

The universal recognition problems of both ITGs and (2,2)-BRCGs can be solved in time $\mathcal{O}(n^6|G|)$. This may come as a surprise, as ITGs restrict the alignment space considerably, while (2,2)-BRCGs induce all possible alignments. In the context of the NP-hardness of decoding in statistical machine translation (Knight, 1999; Udupa and Maji, 2006), it is natural to ask why the universal recognition problem of (2,2)-BRCGs isn't NP-hard? This paper bridges the gap between these results and shows that when alignments are restricted to be $1 : 1$, island-free or sure-possible sorted, or all combinations thereof, the alignment problem of (2,2)-BRCGs *is* NP-hard. Consequently, while the unrestricted alignment problem for (2,2)-BRCGs can be solved in $\mathcal{O}(n^6|G|)$, the alignment problem turns NP-hard as soon as restrictions are put on the alignments sought. So the extra expressivity in a way comes at the expense of control over the kind of alignments obtained. Note also that an alignment of two words may be enforced multiple times in a (2,2)-BRCGs parse, since two derivation trees that share leaves on both sides can align the same two words.

Our results are not intended to be qualifications of the usefulness of (2,2)-BRCGs (Søgaard, 2008), but rather they are attempts to bridge a gap in our understanding of the synchronous grammar formalisms at hand to us in syntax-based machine translation.

$$
\begin{array}{rcl}
S(X_1, Y_1) & \rightarrow & \rho_1(X_1, Y_1)\rho_2(X_1, Y_1) \\
 & & \rho_3(X_1, Y_1)\rho_4(X_1, Y_1) \\
 & & U^2(X_1, Y_1)\delta^4(X_1, Y_1) \\
\rho_1(X_1\rho_1X_2, Y_1aY_2) & \rightarrow & \top(X_1)\top(X_2)\top(Y_1)\top(Y_2) \\
\rho_1(X_1\rho_1X_2, Y_1cY_2) & \rightarrow & \top(X_1)\top(X_2)\top(Y_1)\top(Y_2) \\
 & \ldots & \\
U^2(X_1UX_2, aaaaY_1) & \rightarrow & U^1(X_1, Y_1)\top(X_2) \\
U^1(X_1UX_2, Y_1bbbbY_2) & \rightarrow & \top(X_1)\top(Y_1)\top(X_2)\top(Y_2) \\
U^2(X_1UX_2, Y_1bbbbY_2) & \rightarrow & U^1(X_1, Y_1)\top(X_2)\top(Y_2) \\
 & \ldots & \\
\delta^4(X_1\delta X_2, Y_1aY_2) & \rightarrow & \delta^3(X_1, Y_1)\top(X_2)\top(Y_2) \\
\delta^4(X_1\delta X_2, Y_1bY_2) & \rightarrow & \delta^3(X_1, Y_1)\top(X_2)\top(Y_2) \\
 & \ldots &
\end{array}
$$

Figure 2: A (2,2)-BRCG for the instance of the vertex cover problem $\langle\langle\{a, b, c, d\}, \{(a, c), (b, c), (b, d), (c, d)\}\rangle, 2\rangle$.

References

Alfred Aho and Jeffrey Ullman. 1972. *The theory of parsing, translation and compiling*. Prentice-Hall, London, England.

Edward Barton, Robert Berwick, and Erik Ristad. 1987. *Computational complexity and natural language*. MIT Press, Cambridge, Massachusetts.

Pierre Boullier. 1998. Proposal for a natural language processing syntactic backbone. Technical report, INRIA, Le Chesnay, France.

Michael Garey and David Johnson. 1979. *Computers and intractability*. W. H. Freeman & Co., New York, New York.

Kevin Knight. 1999. Decoding complexity in word-replacement translation models. *Computational Linguistics*, 25(4):607–615.

Dan Melamed. 1999. Bitext maps and alignment via pattern recognition. *Computational Linguistics*, 25(1):107–130.

Anders Søgaard. 2008. Range concatenation grammars for translation. In *Proceedings of the 22nd International Conference on Computational Linguistics, Companion Volume*, pages 103–106, Manchester, England.

Raghavendra Udupa and Hemanta Maji. 2006. Computational complexity of statistical machine translation. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics*, pages 25–32, Trento, Italy.

Dekai Wu. 1997. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*, 23(3):377–403.

Hao Zhang and Daniel Gildea. 2004. Syntax-based alignment: supervised or unsupervised? In *Proceedings of the 20th International Conference on Computational Linguistics*, pages 418–424, Geneva, Switzerland.