Range concatenation grammars for translation

Anders Søgaard
University of Potsdam
soegaard@ling.uni-potsdam.de

Abstract

Positive and bottom-up non-erasing binary range concatenation grammars (Boullier, 1998) with at most binary predicates ((2,2)-BRCGs) is a $\mathcal{O}(|G|n^6)$ time strict extension of inversion transduction grammars (Wu, 1997) (ITGs). It is shown that (2,2)-BRCGs induce inside-out alignments (Wu, 1997) and cross-serial discontinuous translation units (CDTUs); both phenomena can be shown to occur frequently in many hand-aligned parallel corpora. A CYK-style parsing algorithm is introduced, and induction from aligment structures is briefly discussed.

Range concatenation grammars (RCG) (Boullier, 1998) mainly attracted attention in the formal language community, since they recognize exactly the polynomial time recognizable languages, but recently they have been argued to be useful for data-driven parsing too (Maier and Søgaard, 2008). Bertsch and Nederhof (2001) present the only work to our knowledge on using RCGs for translation. Both Bertsch and Nederhof (2001) and Maier and Søgaard (2008), however, only make use of so-called *simple* RCGs, known to be equivalent to linear context-free rewrite systems (LCFRSs) (Weir, 1988; Boullier, 1998). Our strict extension of ITGs, on the other hand, makes use of the ability to copy substrings in RCG derivations; one of the things that makes RCGs strictly more expressive than LCFRSs. Copying enables us to recognize the intersection of any two translations that we can recognize and induce the union of any two alignment structures that we can induce. Our extension of ITGs in fact introduces two things: (i) A clause may introduce any number of terminals. This enables us to induce multiword translation units. (ii) A clause may copy a substring, i.e. a clause can associate two or more nonterminals $A_1, \ldots A_n$ with the same substring and thereby check if the substring is in the intersection of the languages of the subgrammars with start predicate names $A_1, \ldots A_n$.

The first point is motivated by studies such as Zens and Ney (2003) and simply reflects that in order to induce multiword translation units in this kind of synchronous grammars, it is useful to be able to introduce multiple terminals simultaneously. The second point gives us a handle on context-sensitivity. It means that (2,2)-BRCGs can define translations such as $\{\langle a^n b^m c^n d^m, a^n b^m d^m c^n \rangle \mid m, n \geq 0\}$, i.e. a translation of cross-serial dependencies into nested ones; but it also means that (2,2)-BRCGs induce a larger class of alignment structures. In fact the set of alignment structures that can be induced is closed under union, i.e. any alignment structure can be induced. The last point is of practical interest. It is shown below that phenomena such as inside-out alignments and CDTUs, which cannot be induced by ITGs, but by (2,2)-BRCGs, occur frequently in many hand-aligned parallel corpora.

1 (2,2)-BRCGs and ITGs

(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

Coling 2008: Companion volume – Posters and Demonstrations, pages 103–106
Manchester, August 2008

unary start predicate name S' simply by adding a clause $S'(X_1X_2) \rightarrow S(X_1, X_2)$.

Definition 1.1 (Positive RCGs)**.** 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: $N \rightarrow \mathbb{Z}^*$, T and V are finite sets of, resp., terminal and variables. P is a finite set of clauses of the form $\psi_0 \rightarrow \psi_1 \ldots \psi_m$, where and each of the $\psi_i, 0 \leq i \leq m$, is a predicate of the form $A(\alpha_1, \ldots, \alpha_{\rho(A)})$. 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 $\langle w_1 \ldots w_n, v_{n+2} \ldots v_{n+1+m}\rangle$ 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 $w_{i+1} \ldots w_j$ in the source string or a substring $v_{i+1} \ldots v_j$ 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 $A(\langle g \ldots h\rangle, \langle i \ldots j\rangle) \rightarrow B(\langle g \ldots h\rangle, \langle i+1 \ldots j-1\rangle)$ 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 $L(G) = \{\langle w_1 \ldots w_n, v_{n+2} \ldots v_{n+1+m}\rangle \mid S(\langle 0, n\rangle, \langle n+1, n+1+m\rangle) \overset{*}{\Longrightarrow} \epsilon\}$, i.e. an input string pair $\langle w_1 \ldots w_n, v_{n+2} \ldots v_{n+1+m}\rangle$ is recognized iff the empty string can be derived from $S(\langle 0, n\rangle, \langle n+1, n+1+m\rangle)$.

Theorem 1.2 ((Boullier, 2000))**.** *The recognition problem of bottom-up non-erasing k-RCG can be solved in time $\mathcal{O}(|G|n^d)$ where $d = max_{c_j \in P}(k_j + v_j)$ where c_j is the jth clause in P, k_j is the arity of its LHS predicate, and v_j is the number of different variables in that LHS predicate.*

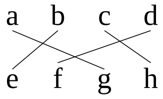
It follows immediately that the recognition problem of (2,2)-BRCG can be solved in time $\mathcal{O}(|G|n^6)$, since k_j can be at most 2, and v_j can be at most 4.

Example 1.3. Consider the (2,2)-BRCG $G = \langle\{S_0, S_1, S_2\}, \{a, b, c, d, e, f, g, h\}, \{X_1, X_2, Y_1, Y_2\}, P, S_0\rangle$ with P the following set of clauses:

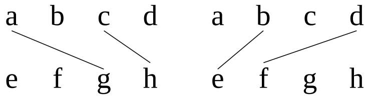
$$\begin{array}{rcl}
S_0(X_1, Y_1) & \rightarrow & S_1(X_1, Y_1)S_2(X_1, Y_1) \\
S_1(X_1d, Y_1Y_2) & \rightarrow & A_0(X_1, Y_2)E(Y_1) \\
A_0(X_1c, Y_1h) & \rightarrow & A_1(X_1, Y_1) \\
A_1(aX_1, g) & \rightarrow & B(X_1) \\
S_2(aX_1, Y_1Y_2) & \rightarrow & T_0(X_1, Y_1)G(Y_2) \\
T_0(X_1d, Y_1f) & \rightarrow & T_1(X_1, Y_1) \\
T_1(bX_1, e) & \rightarrow & C(X_1)
\end{array}$$

$$\begin{array}{rcl|rcl}
B(b) & \rightarrow & \epsilon & C(c) & \rightarrow & \epsilon \\
E(ef) & \rightarrow & \epsilon & G(gh) & \rightarrow & \epsilon
\end{array}$$

which when words that are recognized simultaneously are aligned, induces the alignment:



by inducing the alignments in the, resp., S_1 and S_2 derivations:



Example 1.4. 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:

$$\begin{array}{rcl}
S_s(X_1, Y_1) & \rightarrow & S_0(X_1, Y_1)S_0'(X_1, Y_1) \\
S_0(X_1X_2, Y_1) & \rightarrow & S_1(X_1, Y_1)D(X_2) \\
S_1(aX_1c, abY_1) & \rightarrow & S_1(X_1, Y_1) \\
S_1(X_1, Y_1Y_2) & \rightarrow & B(X_1)C(Y_1)D(Y_2)
\end{array}$$

$$\begin{array}{rcl}
S_0'(X_1X_2, Y_1) & \rightarrow & S_1'(X_2, Y_1)A(X_1) \\
S_1'(bX_1d, Y_1cd) & \rightarrow & S_1'(X_1, Y_1) \\
S_1'(X_1, Y_1Y_2) & \rightarrow & C(X_1)A(Y_1)B(Y_2)
\end{array}$$

$$\begin{array}{rcl|rcl}
A(aX_1) & \rightarrow & A(X_1) & A(\epsilon) & \rightarrow & \epsilon \\
B(bX_1) & \rightarrow & B(X_1) & B(\epsilon) & \rightarrow & \epsilon \\
C(cX_1) & \rightarrow & C(X_1) & C(\epsilon) & \rightarrow & \epsilon \\
D(dX_1) & \rightarrow & D(X_1) & D(\epsilon) & \rightarrow & \epsilon
\end{array}$$

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

Since the component grammars in ITGs are context-free, Example 1.4 shows that there is at least one translation not recognizable by ITGs that is recognized by a (2,2)-BRCG; $\{a^n b^m c^n d^m \mid m, n \geq 0\}$ is known to be non-context-free. ITGs translate into simple (2,2)-BRCGs in the following way; see Wu (1997) for a definition of ITGs. The left column is ITG production rules; the right column their translations in simple (2,2)-BRCGs.

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

It follows immediately that

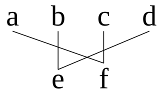
Theorem 1.5. *(2,2)-BRCGs are strictly more expressive than ITGs.*

2 Alignment capacity

Zens and Ney (2003) identify a class of alignment structures that cannot be induced by ITGs, but that can be induced by a number of similar synchronous grammar formalisms, e.g. synchronous tree substitution grammar (STSG) (Eisner, 2003).

Inside-out alignments (Wu, 1997), such as the one in Example 1.3, cannot be induced by *any* of these theories; in fact, there seems to be no useful synchronous grammar formalisms available that handle inside-out alignments, with the possible exceptions of synchronous tree-adjoining grammars (Shieber and Schabes, 1990), Bertsch and Nederhof (2001) and generalized multitext grammars (Melamed et al., 2004), which are all way more complex than ITG, STSG and (2,2)-BRCG. Nevertheless, Wellington et al. (2006) report that 5% of the sentence pairs in an aligned parallel Chinese–English corpus contained inside-out alignments. Example 1.3 shows that (2,2)-BRCGs induce inside-out alignments.

An even stronger motivation for using (2,2)-BRCG for translation is the existence of cross-serial DTUs (CDTUs). Informally, a CDTU is a DTU such that there is a part of another DTU in its gap. Here's a simple example:



Neither ITGs nor STSGs can induce CDTUs; ITGs cannot induce DTUs with multiple gaps (MDTUs) either. Our experiments are summarized in Figure 1. Overall the results show that handling CDTUs is important for alignment error rates.

3 Parsing and induction from alignments

A CYK-style algorithm is presented for (2,2)-BRCG in Figure 2; it is assumed, w.l.o.g, that if the same variable occurs twice in the LHS of a clause, the clause is of the form $A_0(X_1, Y_1) \rightarrow A_1(X_1, Y_1)A_2(X_1, Y_1)$. It modifies the original CYK algorithm (Younger, 1967) in four ways: (i) It uses two charts; one for the source string (s) and one for the target string (t). (ii) Pairs of nonterminals and integers (A, ι), rather than just nonterminals, are stored in the cells of the chart (l. 2,4,6,7). Integers represent derivation steps at which nonterminals are inserted. (iii) Multiple terminals are allowed (l. 2,6,7). (iv) If a clause is copying, the same two cells in the chart are visited twice (l. 4). Note that the variable ι in insertion, e.g. in l. 4/1, is the *current* derivation step, but ι_i in look-up, e.g. in l. 4/2, is the derivation step in which the associated nonterminal was added to the chart.

The overall runtime of this algorithm is in $\mathcal{O}(|G|n^6)$, since it has, for branching clauses, six embedded loops that iterate over the string, i.e. the four **for** loops and the two $\exists$s in Figure 2.

The induction problem from alignments can be reduced to the induction problem for ITGs by simply unravelling the alignment structures. The simplest algorithm for doing this assumes that alignments are sequences of translation units, and considers each at a time. If a gap is found, the translation unit is a DTU and is moved to a new alignment structure. The complexity of the algorithm is quadratic in the length of the input sentences, i.e. linear in the size of the alignment structure, and for a sentence pair $\langle w_1 \ldots w_n, v_1 \ldots v_m \rangle$ the ITG induction algorithm has to consider at most $\frac{\mathbf{min}(n+m)}{2}$ aligment structures.

4 Conclusion

A new class of grammars for syntax-based machine translation was presented; while its recognition problem remains solvable in time $\mathcal{O}(|G|n^6)$, the grammars induce frequently occurring alignment configurations that cannot be induced by comparable classes of grammars in the literature. A parsing and an induction algorithm were presented.

	Sent.	TUs	DTUs	CDTUs	MDTUs	CDTUs/Sent.
English–French:	100	937	95	36	11	36%
English-Portuguese:	100	939	100	52	3	52%
English–Spanish:	100	950	90	26	7	26%
Portuguese–French:	100	915	77	19	3	19%
Portuguese–Spanish:	100	991	80	40	3	40%
Spanish–French:	100	975	74	24	8	24%

Figure 1: Statistics for six 100-sentence hand-aligned Europarl bitexts (Graca et al., 2008).

```
BUILD$(s, [w_1 \ldots w_n]), (t, [v_1 \ldots v_m])$

for $j \leftarrow 1$ to $n$, for $j' \leftarrow 1$ to $m$
do $s(i-1, j), t(i'-1, j') \leftarrow \{(A, \iota) \mid A(w_i \ldots w_j, v_{i'} \ldots v_{j'}) \rightarrow \epsilon \in P\}$
    for $k \leftarrow (j-1)$ to $0$, for $k' \leftarrow (j'-1)$ to $0$
    do $s(k, j), t(k', j') \leftarrow \{(A, \iota) \mid A(X_1, Y_1) \rightarrow B(X_1, Y_1)C(X_1, Y_1) \in P,$
    $(B, \iota_1), (C, \iota_2) \in s(k, j), (B, \iota_1), (C, \iota_2) \in t(k', j')\}$
    for $l \leftarrow (j-2)$ to $0$, for $l' \leftarrow (j'-2)$ to $0$
    do $s(l, j), t(l', j') \leftarrow \{(A, \iota) \mid A(\phi_1 X_1 \phi_2 X_2 \phi_3, \psi_1 Y_1 \psi_2 Y_2 \psi_3) \rightarrow B(X_1, Y_1)C(X_2, Y_2) \in P,$
    $\exists i.(B, \iota_1) \in s(l + |\phi_1|, i), (C, \iota_2) \in s(i + |\phi_2|, j - |\phi_3|), \phi_1 = w_{l+1} \ldots w_{l+|\phi_1|},$
    $\phi_2 = w_{i+1} \ldots w_{i+|\phi_2|}, \phi_3 = w_{j-|\phi_3|} \ldots w_j,$
    $\exists i'.(B, \iota_1) \in t(l' + |\psi_1|, i'), (C, \iota_2) \in t(i' + |\psi_2|, j' - |\psi_3|), \psi_1 = v_{l'+1} \ldots v_{l'+|\psi_1|},$
    $\psi_2 = v_{i'+1} \ldots v_{i'+|\psi_2|}, \psi_3 = v_{j'-|\psi_3|} \ldots v_{j'}\}$
    do $s(l, j), t(l', j') \leftarrow \{(A, \iota) \mid A(\phi_1 X_1 \phi_2 X_2 \phi_3, \psi_1 Y_1 \psi_2 Y_2 \psi_3) \rightarrow B(X_1, Y_1)C(X_2, Y_2) \in P,$
    $\exists i.(B, \iota_1) \in s(l + |\phi_1|, i), (C, \iota_2) \in s(i + |\phi_2|, j - |\phi_3|), \phi_1 = w_{l+1} \ldots w_{l+|\phi_1|},$
    $\phi_2 = w_{i+1} \ldots w_{i+|\phi_2|}, \phi_3 = w_{j-|\phi_3|} \ldots w_j,$
    $\exists i'.(C, \iota_2) \in t(l' + |\psi_1|, i'), (B, \iota_1) \in t(i' + |\psi_2|, j' - |\psi_3|), \psi_1 = v_{l'+1} \ldots v_{l'+|\psi_1|},$
    $\psi_2 = v_{i'+1} \ldots v_{i'+|\psi_2|}, \psi_3 = v_{j'-|\psi_3|} \ldots v_{j'}\}$
if $(S, \iota_1) \in s(0, n), (S, \iota_1) \in t(0, m)$ then return *success* else *failure*
```

Figure 2: CYK-style parsing algorithm for (2,2)-BRCG.

References

Bertsch, Eberhard and Mark-Jan Nederhof. 2001. On the complexity of some extensions of RCG parsing. In *Proceedings of the 7th International Workshop on Parsing Technologies*, pages 66–77, Beijing, China.

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

Boullier, Pierre. 2000. A cubic time extension of context-free grammars. *Grammars*, 3(2–3):111–131.

Eisner, Jason. 2003. Learning non-isomorphic tree mappings for machine translation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 205–208, Sapporo, Japan.

Graca, Joao, Joana Pardal, Luísa Coheur, and Diamantino Caseiro. 2008. Building a golden collection of parallel multi-language word alignments. In *Proceedings of the 6th International Conference on Language Resources and Evaluation*, Marrakech, Morocco.

Maier, Wolfgang and Anders Søgaard. 2008. Treebanks and mild context-sensitivity. In *Proceedings of the 13th Conference on Formal Grammar*, Hamburg, Germany.

Melamed, Dan, Giorgio Satta, and Benjamin Wellington. 2004. Generalized multitext grammars. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*, pages 661–668, Barcelona, Spain.

Shieber, Stuart and Yves Schabes. 1990. Synchronous tree-adjoining grammars. In *Proceedings of the 13th Conference on Computational Linguistics*, pages 253–258, Helsinki, Finland.

Weir, David. 1988. *Characterizing mildly context-sensitive grammar formalisms*. Ph.D. thesis, University of Pennsylvania, Philadelphia, Pennsylvania.

Wellington, Benjamin, Sonjia Waxmonsky, and Dan Melamed. 2006. Empirical lower bounds on the complexity of translational equivalence. In *Proceedings of the 44th Annual Conference of the Association for Computational Linguistics*, pages 977–984, Sydney, Australia.

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

Younger, Daniel. 1967. Recognition and parsing of context-free languages in time n^3. *Information and Control*, 10(2):189–208.

Zens, Richard and Hermann Ney. 2003. A comparative study on reordering constraints in statistical machine translation. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics*, pages 144–151, Sapporo, Japan.