An Efficient Generation Algorithm for Lexicalist MT

Victor Poznański, John L. Beaven & Pete Whitelock *
SHARP Laboratories of Europe Ltd.
Oxford Science Park, Oxford OX4 4GA
United Kingdom
{vp,jlb,pete}@sharp.co.uk

Abstract

The lexicalist approach to Machine Translation offers significant advantages in the development of linguistic descriptions. However, the Shake-and-Bake generation algorithm of (Whitelock, 1992) is NP-complete. We present a polynomial time algorithm for lexicalist MT generation provided that sufficient information can be transferred to ensure more determinism.

1 Introduction

Lexicalist approaches to MT, particularly those incorporating the technique of *Shake-and-Bake* generation (Beaven, 1992a; Beaven, 1992b; Whitelock, 1994), combine the linguistic advantages of transfer (Arnold et al., 1988; Allegranza et al., 1991) and interlingual (Nirenburg et al., 1992; Dorr, 1993) approaches. Unfortunately, the generation algorithms described to date have been intractable. In this paper, we describe an alternative generation component which has polynomial time complexity.

Shake-and-Bake translation assumes a source grammar, a target grammar and a bilingual dictionary which relates translationally equivalent sets of lexical signs, carrying across the semantic dependencies established by the source language analysis stage into the target language generation stage.

The translation process consists of three phases:

1. A *parsing phase*, which outputs a multiset, or *bag*, of source language signs instantiated with sufficiently rich linguistic information established by the parse to ensure adequate translations.
2. A *lexical-semantic transfer phase* which employs the bilingual dictionary to map the bag of instantiated source signs onto a bag of target language signs.
3. A *generation phase* which imposes an order on the bag of target signs which is guaranteed grammatical according to the monolingual target grammar. This ordering must respect the linguistic constraints which have been transferred into the target signs.

The *Shake-and-Bake* generation algorithm of (Whitelock, 1992) combines target language signs using the technique known as *generate-and-test*. In effect, an arbitrary permutation of signs is input to a shift-reduce parser which tests them for grammatical well-formedness. If they are well-formed, the system halts indicating success. If not, another permutation is tried and the process repeated. The complexity of this algorithm is $O(n!)$ because all permutations ($n!$ for an input of size n) may have to be explored to find the correct answer, and indeed *must* be explored in order to verify that there is no answer.

Proponents of the Shake-and-Bake approach have employed various techniques to improve generation efficiency. For example, (Beaven, 1992a) employs a chart to avoid recalculating the same combinations of signs more than once during testing, and (Popowich, 1994) proposes a more general technique for storing which rule applications have been attempted; (Brew, 1992) avoids certain pathological cases by employing global constraints on the solution space; researchers such as (Brown et al., 1990) and (Chen and Lee, 1994) provide a system for bag generation that is heuristically guided by probabilities. However, none of these approaches is guaranteed to avoid protracted search times if an exact answer is required, because bag generation is NP-complete (Brew, 1992).

Our novel generation algorithm has polynomial complexity ($O(n^4)$). The reduction in theoretical complexity is achieved by placing constraints on the power of the target grammar when operating on instantiated signs, and by using a more restrictive data structure than a bag, which we call a *target language normalised commutative bracketing*

*We wish to thank our colleagues Kerima Benkerimi, David Elworthy, Peter Gibbins, Ian Johnson, Andrew Kay and Antonio Sanfilippo at SLE, and our anonymous reviewers for useful feedback and discussions on the research reported here and on earlier drafts of this paper.

(TNCB). A TNCB records dominance information from derivations and is amenable to incremental updates. This allows us to employ a greedy algorithm to refine the structure progressively until either a target constituent is found and generation has succeeded or no more changes can be made and generation has failed.

In the following sections, we will sketch the basic algorithm, consider how to provide it with an initial guess, and provide an informal proof of its efficiency.

2 A Greedy Incremental Generation Algorithm

We begin by describing the fundamentals of a greedy incremental generation algorithm. The crucial data structure that it employs is the *TNCB*. We give some definitions, state some key assumptions about suitable TNCBs for generation, and then describe the algorithm itself.

2.1 TNCBs

We assume a sign-based grammar with binary rules, each of which may be used to *combine* two signs by unifying them with the daughter categories and returning the mother. Combination is the commutative equivalent of rule application; the linear ordering of the daughters that leads to successful rule application determines the orthography of the mother.

Whitelock's Shake-and-Bake generation algorithm attempts to arrange the bag of target signs until a grammatical ordering (an ordering which allows all of the signs to combine to yield a single sign) is found. However, the target *derivation* information itself is not used to assist the algorithm. Even in (Beaven, 1992a), the derivation information is used simply to cache previous results to avoid exact recomputation at a later stage, not to improve on previous guesses. The reason why we believe such improvement is possible is that, given adequate information from the previous stages, two target signs cannot combine by accident; they must do so because the underlying semantics within the signs licenses it.

If the linguistic data that two signs contain allows them to combine, it is because they are providing a semantics which might later become more specified. For example, consider the bag of signs that have been derived through the Shake-and-Bake process which represent the phrase:

(1) The big brown dog

Now, since the determiner and adjectives all modify the same noun, most grammars will allow us to construct the phrases:

(2) The dog
(3) The big dog
(4) The brown dog

as well as the 'correct' one. Generation will fail if all signs in the bag are not eventually incorporated in the final result, but in the naïve algorithm, the intervening computation may be intractable.

In the algorithm presented here, we start from observation that the phrases (2) to (4) are not incorrect semantically; they are simply under-specifications of (1). We take advantage of this by recording the constituents that have combined within the TNCB, which is designed to allow further constituents to be incorporated with minimal recomputation.

A TNCB is composed of a sign, and a history of how it was derived from its children. The structure is essentially a binary derivation tree whose children are unordered. Concretely, it is either NIL, or a triple:

$$\begin{aligned} \text{TNCB} &= \text{NIL} \mid \text{Value} \times \text{TNCB} \times \text{TNCB} \\ \text{Value} &= \text{Sign} \mid \\ &\quad \text{INCONSISTENT} \mid \\ &\quad \text{UNDETERMINED} \end{aligned}$$

The second and third items of the TNCB triple are the *child TNCBs*. The *value* of a TNCB is the sign that is formed from the combination of its children, or *INCONSISTENT*, representing the fact that they cannot grammatically combine, or *UNDETERMINED*, i.e. it has not yet been established whether the signs combine.

Undetermined TNCBs are commutative, e.g. they do not distinguish between the structures shown in Figure 1.

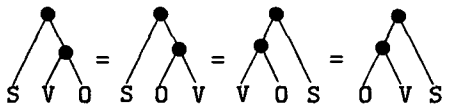

Figure 1: Equivalent TNCBs

In section 3 we will see that this property is important when starting up the generation process.

Let us introduce some terminology.

A TNCB is

- *well-formed* iff its value is a sign,
- *ill-formed* iff its value is INCONSISTENT,
- *undetermined* (and its value is UNDETERMINED) iff it has not been demonstrated whether it is well-formed or ill-formed.
- *maximal* iff it is well-formed and its parent (if it has one) is ill-formed. In other words, a maximal TNCB is a largest well-formed component of a TNCB.

Since TNCBs are tree-like structures, if a TNCB is undetermined or ill-formed then so are all of its ancestors (the TNCBs that contain it).

We define five operations on a TNCB. The first three are used to define the fourth transformation (*move*) which improves ill-formed TNCBs. The fifth is used to establish the well-formedness of undetermined nodes. In the diagrams, we use a cross to represent ill-formed nodes and a black circle to represent undetermined ones.

Deletion: A maximal TNCB can be deleted from its current position. The structure above it must be adjusted in order to maintain binary branching. In figure 2, we see that when node 4 is deleted, so is its parent node 3. The new node 6, representing the combination of 2 and 5, is marked undetermined.

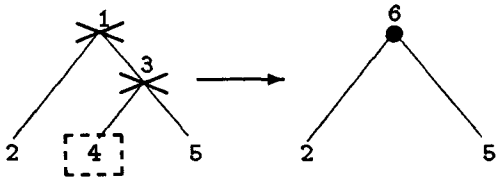

Figure 2: 4 is deleted, raising 5

Conjunction: A maximal TNCB can be conjoined with another maximal TNCB if they may be combined by rule. In figure 3, it can be seen how the maximal TNCB composed of nodes 1, 2, and 3 is conjoined with the maximal TNCB composed of nodes 4, 5 and 6 giving the TNCB made up of nodes 1 to 7. The new node, 7, is well-formed.

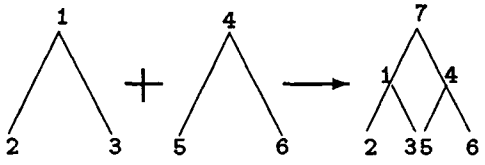

Figure 3: 1 is conjoined with 4 giving 7

Adjunction: A maximal TNCB can be inserted inside a maximal TNCB, i.e. conjoined with a non-maximal TNCB, where the combination is licensed by rule. In figure 4, the TNCB composed of nodes 1, 2, and 3 is inserted inside the TNCB composed of nodes 4, 5 and 6. All nodes (only 8 in figure 4) which dominate the node corresponding to the new combination (node 7) must be marked undetermined — such nodes are said to be disrupted.

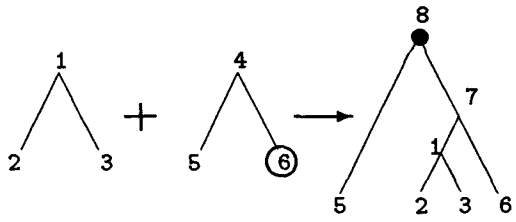

Figure 4: 1 is adjoined next to 6 inside 4

Movement: This is a combination of a deletion with a subsequent conjunction or adjunction. In figure 5, we illustrate a move via conjunction. In the left-hand figure, we assume we wish to move the maximal TNCB 4 next to the maximal TNCB 7. This first involves deleting TNCB 4 (noting it), and raising node 3 to replace node 2. We then introduce node 8 above node 7, and make both nodes 7 and 4 its children. Note that during deletion, we remove a surplus node (node 2 in this case) and during conjunction or adjunction we introduce a new one (node 8 in this case) thus maintaining the same number of nodes in the tree.

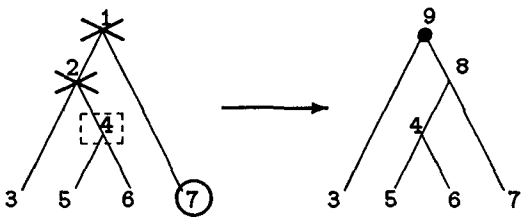

Figure 5: A conjoining move from 4 to 7

Evaluation: After a movement, the TNCB is undetermined as demonstrated in figure 5. The signs of the affected parts must be recalculated by combining the recursively evaluated child TNCBs.

2.2 Suitable Grammars

The Shake-and-Bake system of (Whitelock, 1992) employs a bag generation algorithm because it is assumed that the input to the generator is no more than a collection of instantiated signs. Full-scale bag generation is not necessary because sufficient information can be transferred from the source language to severely constrain the subsequent search during generation.

The two properties required of TNCBs (and hence the target grammars with instantiated lexical signs) are:

1. **Precedence Monotonicity.** The order of the

orthographies of two combining signs in the orthography of the result must be determinate — it must not depend on any subsequent combination that the result may undergo. This constraint says that if one constituent fails to combine with another, no permutation of the elements making up either would render the combination possible. This allows bottom-up evaluation to occur in linear time. In practice, this restriction requires that sufficiently rich information be transferred from the previous translation stages to ensure that sign combination is deterministic.

2. **Dominance Monotonicity.** If a maximal TNCB is adjoined at the highest possible place inside another TNCB, the result will be well-formed after it is re-evaluated. Adjunction is only attempted if conjunction fails (in fact conjunction is merely a special case of adjunction in which no nodes are disrupted); an adjunction which disrupts i nodes is attempted before one which disrupts $i+1$ nodes. Dominance monotonicity merely requires all nodes that are disrupted under this top-down control regime to be well-formed when re-evaluated. We will see that this will ensure the termination of the generation algorithm within $n-1$ steps, where n is the number of lexical signs input to the process.

We are currently investigating the mathematical characterisation of grammars and instantiated signs that obey these constraints. So far, we have not found these restrictions particularly problematic.

2.3 The Generation Algorithm

The generator cycles through two phases: a *test* phase and a *rewrite* phase. Imagine a bag of signs, corresponding to "the big brown dog barked", has been passed to the generation phase. The first step in the generation process is to convert it into some arbitrary TNCB structure, say the one in figure 6. In order to verify whether this structure is valid, we evaluate the TNCB. This is the test phase. If the TNCB evaluates successfully, the orthography of its value is the desired result. If not, we enter the rewrite phase.

If we were continuing in the spirit of the original Shake-and-Bake generation process, we would now form some arbitrary mutation of the TNCB and retest, repeating this test-rewrite cycle until we either found a well-formed TNCB or failed. However, this would also be intractable due to the undirectedness of the search through the vast number of possibilities. Given the added derivation information contained within TNCBs and the properties mentioned above, we can direct this search by incrementally improving on previously evaluated results.

We enter the rewrite phase, then, with an ill-formed TNCB. Each move operation must improve

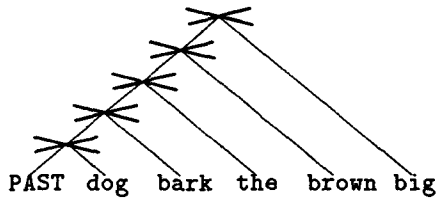

Figure 6: An arbitrary right-branching TNCB structure

it. Let us see why this is so.

The *move* operation maintains the same number of nodes in the tree. The deletion of a maximal TNCB removes two ill-formed nodes (figure 2). At the deletion site, a new undetermined node is created, which may or may not be ill-formed. At the destination site of the movement (whether conjunction or adjunction), a new well-formed node is created.

The ancestors of the new well-formed node will be at least as well-formed as they were prior to the movement. We can verify this by case:

1. When two maximal TNCBs are conjoined, nodes dominating the new node, which were previously ill-formed, become undetermined. When re-evaluated, they may remain ill-formed or some may now become well-formed.
2. When we adjoin a maximal TNCB within another TNCB, nodes dominating the new well-formed node are disrupted. By dominance monotonicity, all nodes which were disrupted by the adjunction must become well-formed after re-evaluation. And nodes dominating the maximal disrupted node, which were previously ill-formed, may become well-formed after re-evaluation.

We thus see that rewriting and re-evaluating must improve the TNCB.

Let us further consider the contrived worst-case starting point provided in figure 6. After the test phase, we discover that every single interior node is ill-formed. We then scan the TNCB, say top-down from left to right, looking for a maximal TNCB to move. In this case, the first move will be *PAST* to *bark*, by conjunction (figure 7).

Once again, the test phase fails to provide a well-formed TNCB, so we repeat the rewrite phase, this time finding *dog* to conjoin with *the* (figure 8 shows the state just after the second pass through the test phase).

After further testing, we again re-enter the rewrite phase and this time note that *brown* can be inserted in the maximal TNCB *the dog barked* adjoined with *dog* (figure 9). Note how, after combining *dog* and *the*, the parent sign reflects the correct orthography

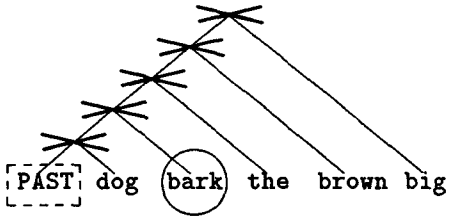

Figure 7: The initial guess

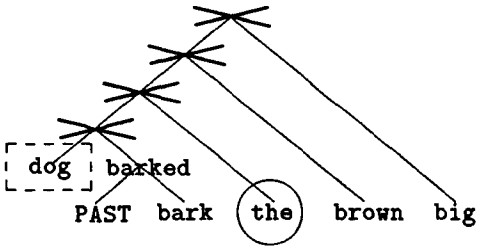

Figure 8: The TNCB after "PAST" is moved to "bark"

even though they did not have the correct linear precedence.

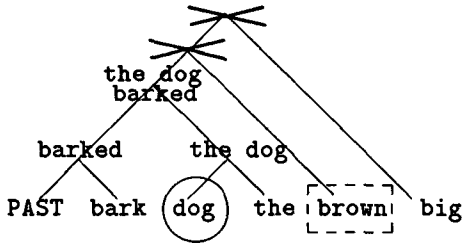

Figure 9: The TNCB after "dog" is moved to "the"

After finding that *big* may not be conjoined with *the brown dog*, we try to adjoin it within the latter. Since it will combine with *brown dog*, no adjunction to a lower TNCB is attempted.

The final result is the TNCB in figure 11, whose orthography is "the big brown dog barked".

We thus see that during generation, we formed a basic constituent, *the dog*, and incrementally refined it by adjoining the modifiers in place. At the heart of this approach is that, once well-formed, constituents can only grow; they can never be dismantled.

Even if generation ultimately fails, maximal well-formed fragments will have been built; the latter may be presented to the user, allowing graceful degradation of output quality.

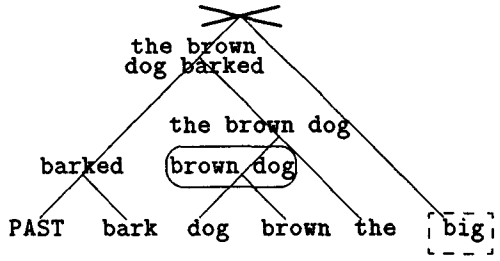

Figure 10: The TNCB after "brown" is moved to "dog"

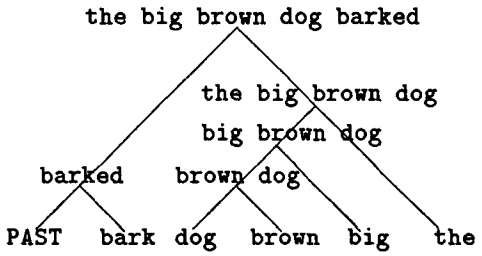

Figure 11: The final TNCB after "big" is moved to "brown dog"

3 Initialising the Generator

Considering the algorithm described above, we note that the number of rewrites necessary to repair the initial guess is no more than the number of ill-formed TNCBs. This can never exceed the number of interior nodes of the TNCB formed from n lexical signs (i.e. $n-2$). Consequently, the better formed the initial TNCB used by the generator, the fewer the number of rewrites required to complete generation. In the last section, we deliberately illustrated an initial guess which was as bad as possible. In this section, we consider a heuristic for producing a motivated guess for the initial TNCB.

Consider the TNCBs in figure 1. If we interpret the S, O and V as Subject, Object and Verb we can observe an equivalence between the structures with the bracketings: (S (V O)), (S (O V)), ((V O) S), and ((O V) S). The implication of this equivalence is that if, say, we are translating into a (S (V O)) language from a head-final language and have isomorphic dominance structures between the source and target parses, then simply mirroring the source parse structure in the initial target TNCB will provide a correct initial guess. For example, the English sentence (5):

(5) the book is red

has a corresponding Japanese equivalent (6):

(6) ((hon wa) (akai desu))
((book TOP) (red is))

If we mirror the Japanese bracketing structure in English to form the initial TNCB, we obtain: ((book the) (red is)). This will produce the correct answer in the test phase of generation without the need to rewrite at all.

Even if there is not an exact isomorphism between the source and target commutative bracketings, the first guess is still reasonable as long as the majority of child commutative bracketings in the target language are isomorphic with their equivalents in the source language. Consider the French sentence:

(7) ((le ((grand chien) brun)) aboya)
(8) ((the ((big dog) brown)) barked)

The TNCB implied by the bracketing in (8) is equivalent to that in figure 10 and requires just one rewrite in order to make it well-formed. We thus see how the TNCBs can mirror the dominance information in the source language parse in order to furnish the generator with a good initial guess. On the other hand, no matter how the SL and TL structures differ, the algorithm will still operate correctly with polynomial complexity. Structural transfer can be incorporated to improve the efficiency of generation, but it is never necessary for correctness or even tractability.

4 The Complexity of the Generator

The theoretical complexity of the generator is $O(n^4)$, where n is the size of the input. We give an informal argument for this. The complexity of the test phase is the number of evaluations that have to be made. Each node must be tested no more than twice in the worst case (due to precedence monotonicity), as one might have to try to combine its children in either direction according to the grammar rules. There are always exactly $n-1$ non-leaf nodes, so the complexity of the test phase is $O(n)$. The complexity of the rewrite phase is that of locating the two TNCBs to be combined. In the worst case, we can imagine picking an arbitrary child TNCB ($O(n)$) and then trying to find another one with which it combines ($O(n)$). The complexity of this phase is therefore the product of the picking and combining complexities, i.e. $O(n^2)$. The combined complexity of the test-rewrite cycle is thus $O(n^3)$. Now, in section 3, we argued that no more than $n-1$ rewrites would ever be necessary, thus the overall complexity of generation (even when no solution is found) is $O(n^4)$.

Average case complexity is dependent on the quality of the first guess, how rapidly the TNCB structure is actually improved, and to what extent the TNCB must be re-evaluated after rewriting. In the SLEMaT system (Poznański et al., 1993), we have tried to form a good initial guess by mirroring the source structure in the target TNCB, and allowing some local structural modifications in the bilingual equivalences.

Structural transfer operations only affect the efficiency and not the functionality of generation. Transfer specifications may be incrementally refined and empirically tested for efficiency. Since complete specification of transfer operations is not required for correct generation of grammatical target text, the version of Shake-and-Bake translation presented here maintains its advantage over traditional transfer models, in this respect.

The monotonicity constraints, on the other hand, might constitute a dilution of the Shake-and-Bake ideal of independent grammars. For instance, precedence monotonicity requires that the status of a clause (strictly, its lexical head) as main or subordinate has to be transferred into German. It is not that the transfer of information *per se* compromises the ideal — such information must often appear in transfer entries to avoid grammatical but incorrect translation (e.g. *a great man* translated as *un homme grand*). The problem is justifying the main/subordinate distinction in every language that we might wish to translate into German. This distinction can be justified monolingually for the other languages that we treat (English, French, and Japanese). Whether the constraints will ultimately require monolingual grammars to be enriched with entirely unmotivated features will only become clear as translation coverage is extended and new language pairs are added.

5 Conclusion

We have presented a polynomial complexity generation algorithm which can form part of any Shake-and-Bake style MT system with suitable grammars and information transfer. The transfer module is free to attempt structural transfer in order to produce the best possible first guess. We tested a TNCB-based generator in the SLEMaT MT system with the pathological cases described in (Brew, 1992) against Whitelock's original generation algorithm, and have obtained speed improvements of several orders of magnitude. Somewhat more surprisingly, even for short sentences which were not problematic for Whitelock's system, the generation component has performed consistently better.

References

V. Allegranza, P. Bennett, J. Durand, F. van Eynde, L. Humphreys, P. Schmidt, and E. Steiner. 1991. Linguistics for Machine Translation: The Eurotra Linguistic Specifications. In C. Copeland, J. Durand, S. Krauwer, and B. Maegaard, editors, *The Eurotra Formal Specifications. Studies in Machine*

Translation and Natural Language Processing 2, pages 15–124. Office for Official Publications of the European Communities.

D. Arnold, S. Krauwer, L. des Tombe, and L. Sadler. 1988. 'Relaxed' Compositionality in Machine Translation. In *Second International Conference on Theoretical and Methodological Issues in Machine Translation of Natural Languages*, Carnegie Mellon Univ, Pittsburgh.

John L. Beaven. 1992a. *Lexicalist Unification-based Machine Translation.* Ph.D. thesis, University of Edinburgh, Edinburgh.

John L. Beaven. 1992b. Shake-and-Bake Machine Translation. In *Proceedings of COLING 92*, pages 602–609, Nantes, France.

Chris Brew. 1992. Letting the Cat out of the Bag: Generation for Shake-and-Bake MT. In *Proceedings of COLING 92*, pages 29–34, Nantes, France.

Peter F. Brown, John Cocke, A Della Pietra, Vincent J. Della Pietra, Fredrick Jelinek, John D. Lafferty, Robert L. Mercer, and Paul S. Roossin. 1990. A Statistical Approach to Machine Translation. *Computational Linguistics*, 16(2):79–85, June.

Hsin-Hsi Chen and Yue-Shi Lee. 1994. A Corrective Training Algorithm for Adaptive Learning in Bag Generation. In *International Conference on New Methods in Language Processing (NeMLaP)*, pages 248–254, Manchester, UK. UMIST.

Bonnie Jean Dorr. 1993. *Machine Translation: A View from the Lexicon.* Artificial Intelligence Series. The MIT Press, Cambridge, Mass.

Sergei Nirenburg, Jaime Carbonell, Masaru Tomita, and Kenneth Goodman. 1992. *Machine Translation: A Knowledge-Based Approach.* Morgan Kaaufmann, San Mateo, CA.

Fred Popowich. 1994. Improving the Efficiency of a Generation Algorithm for Shake and Bake Machine Translation using Head-Driven Phrase Structure Grammar. Technical Report CMPT-TR 94-07, School of Computing Science, Simon Fraser University, Burnaby, British Columbia, CANADA V5A 1S6.

V. Poznański, John L. Beaven, and P. Whitelock. 1993. The Design of SLEMaT Mk II. Technical Report IT-1993-19, Sharp Laboratories of Europe, LTD, Edmund Halley Road, Oxford Science Park, Oxford OX4 4GA, July.

P. Whitelock. 1992. Shake and Bake Translation. In *Proceedings of COLING 92*, pages 610–616, Nantes, France.

P. Whitelock. 1994. Shake-and-Bake Translation. In C. J. Rupp, M. A. Rosner, and R. L. Johnson, editors, *Constraints, Language and Computation*, pages 339–359. Academic Press, London.