CONSTRUCTING LEXICAL TRANSDUCERS

Lauri Karttunen

Rank Xerox Research Centre
Grenoble

0. INTRODUCTION

A **lexical transducer**, first discussed in Karttunen, Kaplan and Zaenen 1992, is a specialised finite-state automaton that maps inflected surface forms to lexical forms, and vice versa. The lexical form consists of a canonical representation of the word and a sequence of tags that show the morphological characteristics of the form in question and its syntactic category. For example, a lexical transducer for French might relate the surface form *veut* to the lexical form *vouloir+IndPr+SG+P3*. In order to map between these two forms, the transducer may contain a path like the one shown in Fig. 1.

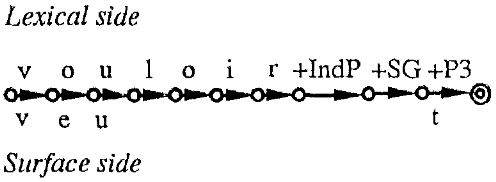

Fig. 1 Transducer path

The circles in Fig. 1 represent states, the arcs (state transitions) are labelled by a pair of symbols: a lexical symbol and a surface symbol. Sometimes they are the same (*v:v*), sometimes different (*o:e*), sometimes one or the other side is empty (= EPSILON). The exact alignment of the symbols on the two sides is not crucial: the path would express the *veut* ↔ *vouloir+IndP+SG+P3* mapping even if the last *t* on the lower side was moved to match the *l* of *vouloir*.

Because finite-state transducers are **bidirectional**, the same transducer can be used for analysis (*veut* → *vouloir +IndP+SG+P3*) as well as generation (*vouloir+IndP+SG+P3* → *veut*). Analysis and generation differ only with respect to the choice of the input side (surface or lexical). The transducer and the analysis/generation algorithm are the same in both directions.

Other advantages that lexical transducers have over other methods of morphological processing are **compactness** and **speed**. The logical size of a transducer for French is around 50K states and 100K arcs but the network can be physically compacted to a few hundred kilobytes. The speed of analysis varies from a few thousand words per second to 10K w/s or more depending on hardware and the degree of compaction.

At this time there exist full-size lexical transducers for at least five languages: English and French (Xerox DDS), German (Lingsoft), Korean (Hyuk-Chul Kwon, see Kwon and Karttunen 1994), and Turkish (Kemal Oflazer). It is expected that by the time of the Coling conference several languages will have been added to the list.

The standard way of constructing lexical transducers, as described in Karttunen, Kaplan and Zaenen 1994, consists of (1) a finite-state source lexicon that defines the set of valid lexical forms of the language (possibly infinite), and (2) a set of finite-state rules that assign the proper surface realisation to all lexical forms and morphological categories of the language. The rules are compiled to transducers and merged with the source lexicon using intersection and composition. The use of intersection is based on the special interpretation of two-level rules as denoting equal

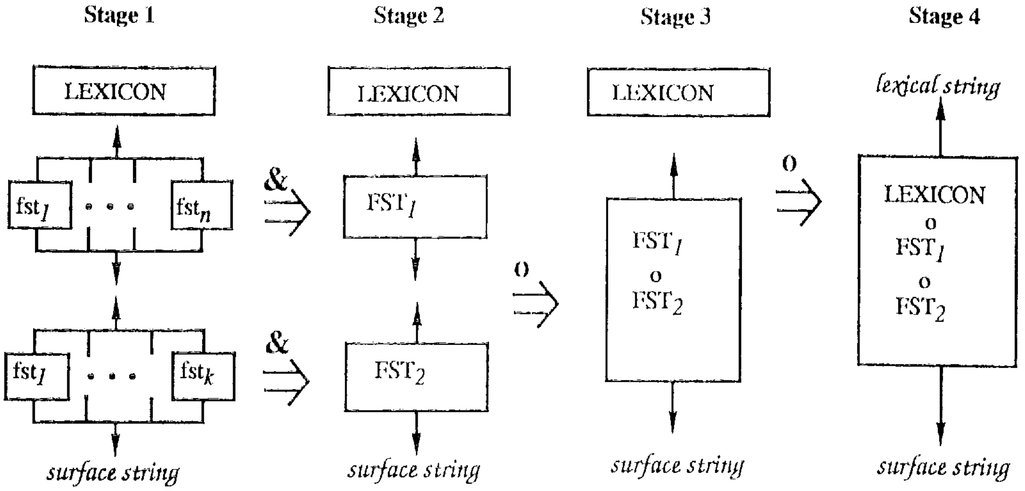

Fig. 2 Construction of a lexical transducer with intersection and composition (Karttunen, Kaplan and Zaenen 1992)

length relations (see Kaplan and Kay 1994 for a detailed discussion).

For practical as well as linguistic reasons it may be desirable to use more than one set of rules to describe the mapping. Fig. 2 illustrates the construction of a lexical transducer for French by two sets of rules using intersection (&) and composition (o) operations.

Stage 1 shows two parallel rule systems arranged in a cascade. In Stage 2, the rules on each level have been intersected to a single transducer. Stage 3 shows the composition of the rule system to a single transducer. Stage 4 shows the final result that is derived by composing the rule system with the lexicon.

Although the conceptual picture is quite straightforward, the actual computations are very resource intensive because of the size of the intermediate structures at Stages 2 and 3. Individual rule transducers are generally quite small but the result of intersecting and composing sets of rule transducers tends to result in very large structures.

This paper describes two recent advances in the construction of lexical transducers that circumvent these problems: (1) moving the description of irregular mappings from the rule system to the source lexicon; (2) performing intersection and composition in single operation. Both of these features have been implemented in the Xerox authoring tools (Karttunen and Beesley 1992, Karttunen 1993) for lexical transducers.

1. LEXICON AS A SET OF RELATIONS

1.1 Stem Alternations

The differences between lexical forms and surface forms may be divided to regular and irregular alternations. Regular variation, such as the *al~aux* pattern in the declension of French nouns (*cheval~chevaux*), affects a large class of similar lexical items; irregular variation is limited to a handful of words like the *i~a* alternation in some English verbs (*swim~swam*). Both types of alternations can be described equally well by means of two-level rules. The only difference is that irregular alternations generally must be constrained with more care, perhaps using diacritic markers to flag the appropriate words, to prevent them from applying outside their proper domain. In standard two-level systems (Antworth 1990), it is not possible to distinguish idiosyncratic mappings from regular alternations.

From a practical point of view, the treatment of irregular alternations by individual two-level rules is not optimal if the number of rules becomes very large. For example, the description of the irregular verb morphology in English requires more than a hundred rules, a large number of which deal with the idiosyncratic behaviour of just one word, such as the rule in (1).

```
(1) "From 'be' to 'is' - Part 1"
 b:i <=> #: _ e:  Irregular:
                  +Pres: +Sg: +P3: ;
```

Here # (word boundary) and `Irregular` (a diacritic assigned to strong verbs in the source lexicon) guarantee that the rule applies just where it is supposed to apply: to derive the first letter of *is* in the present tense 3rd person form of *be*. Another rule is needed to delete the *e* because two-level rules are restricted to apply to just a pair of symbols. This is an artefact of the formalism but even if the *be~is* alternation were to be described by a single rule, the construction of dozens of rules of such limited applicability is a tedious, error prone process.

A natural solution to this problem is to allow idiosyncratic variation to be described without any rules at all, by a simple lexical *fiat*, and to use rules only for the more general alternations. This is one of the novel features of the Xerox lexicon compiler (Karttunen 1993). Technically, it means that the lexicon is not necessarily a collection of simple lexical forms as in classical Kimmo systems but a relation that may include pairs such as <*be+Pres+Sg+P3, is*>.

This can be achieved by changing the lexical format so that the lexicon interpreter compiles an entry like (2) to the transducer shown in Fig. 3.

```
(2)  be+Pres+Sg+P3+Verb:is  Neg? ;
```

Here the colon marks the juncture between the lexical form and its surface realisation, `Neg?` is the continuation class for forms that may be have an attached *n't* clitic.

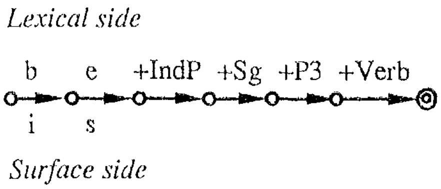

Fig. 3 Entry (2) in compiled form.

The convention used by the Xerox compiler is to interpret paired entries so that the lexical form gets paired with the second form from left to right adding epsilons to the end of the shorter form to make up the difference in length. The location of the epsilons can also be marked explicitly by zeros in the lexical form. For example, if it is important to regard the *s* of *is* as the regular present singular 3rd person ending, the entry could be written as in (3).

```
(3) be+Pres+Sg+P3+Verb:i000s # ;
```

This has the effect of moving the s in Fig. 3 to the right to give a `+P3:s` pair. The mapping from *vouloir+IndP+Sg+P3* to *veut* in Fig. 1 can be achieved in a similar way.

1. 2 Inflectional and Derivational Affixes

Moving all idiosyncratic realisations from the rules to the source lexicon also gives a natural way to describe the realisation of inflectional and derivational suffixes. Although it is possible to write a set of two-level rules that realise the English progressive suffix as *ing*, the rules that make up this mapping have no other function in the English morphology system. A more straight-forward solution is just to include the entry (4) is the appropriate sublexicon.

```
(4)   +Prog:ing  # ;
```

The forms on the lower side of a source lexicon need not be identical to the surface forms of the language; they can still be modified by the rules. This allows a good separation between idiosyncratic and regular aspects of inflectional and derivational morphology. The application of vowel harmony to the case marking suffixes in Finnish illustrates this point.

Finnish case endings are subject to vowel harmony. For example, the abessive ("without") case in Finnish is realised as either *tta* or *ttä* depending on the stem; the latter form occurs in words without back vowels. The general shape of the abessive, *ttA*, is not predictable from anything but the specific realisation of the final vowel (represented by the "archiphoneme" *A*) is just an instance of the general vowel harmony rule in Finnish. This gives us the three-level structure in Fig. 4 for forms like *syyttä* "without reason."

```
s y y +Sg +Abe          lexical form
                        ↕   LEXICON
s y y     t t A   intermediate form
                        ↕   RULES
s y y     t t ä        surface form
```

Fig. 4 Three level analysis of Finnish Abessive.

The mapping from the lexical form to the intermediate form is given by the lexicon; the rules are responsible for the mapping from the lower side of the lexicon to the surface. The intermediate representation disappears in the composition with the rules so that the resulting transducer maps the lexical form directly to the surface form, and vice versa, as in Fig. 5.

```
s  y  y +Sg +Abe          lexical form

s  y  y     t   t   ä     surface form
```

Fig. 5 Final structure after composition.

This computational account coincides nicely with traditional descriptions of Finnish noun inflection (Laaksonen and Lieko 1988).

2. INTERSECTING COMPOSITION

The transfer of irregular alternations from the rules to the lexicon has a significant effect on reducing the number of rules that have to be intersected in order to produce lexical transducers in accordance with the scheme in Fig. 2. Nevertheless, in practice the intersection of even a modest number of rules tends to be quite large. The intermediate rule transducers in Fig. 2 (Stages 2 and 3) may actually be larger than the final result that emerges from the composition with the lexicon in Stage 4. This has been observed in some versions of our French and English lexicons. Fig. 6 illustrates this curious phenomenon.

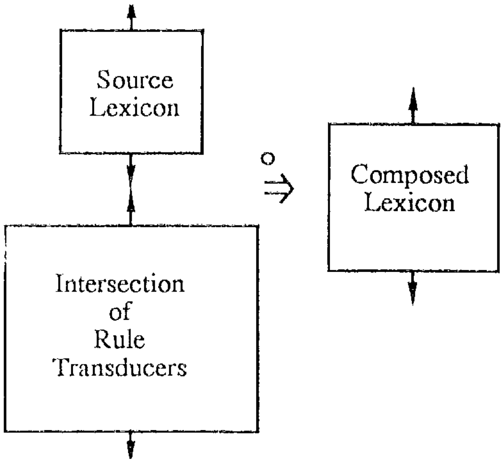

Fig. 6. Composition of a lexicon with rule transducers

As the relative sizes of the three components in Fig. 6 indicate, the composition with the lexicon results in a network that is smaller than the rule network by itself. This is an interesting discovery because it does not follow from the mathematics of the operation. One might expect composition to have the opposite effect. In the worst case, it could yield a result whose size is the product of the sizes of the input machines.

Instead of the expected blowup, the composition of a lexicon with a rule system tends to produce a transducer that is not significantly larger than the source lexicon. The reason appears to be that the rule intersection involves computing the combined effect of the rules for many types complex situations that never occur in the lexicon. Thus the composition with the actual lexicon can be a simplification from the point of the rule system.

This observation suggests that it is advantageous not to do rule intersections as a separate operation, as in Fig. 2. We can avoid the large intermediate results by performing the intersection of the rules and the composition with the lexicon in one combined operation. We call this new operation **intersecting composition.** It takes as input a lexicon (either a simple automaton or a transducer) and any number of rule transducers. The result is what we would get by composing the lexicon with the intersection of the rules.

The basic idea of intersecting composition is to build an output transducer where each state corresponds to a state in the source lexicon and a configuration of rule states. The initial output state corresponds to the initial state of the lexicon and its configuration consists of the initial states of the rule transducers. At every step in the construction, we try to add transitions to our current output state by matching transitions in the current lexicon state against the transitions in all the rule states of the current configuration. The core of the algorithm is given in (5).

(5) <u>Intersecting Composition</u>

> For each transition *x*:*y* in the current lexicon state, find a pair *y*:*z* such that all rule states in the current configuration have a transition for the pair. For each such *y*:*z* pair, get its destination configuration and the corresponding output state. Then build an *x*:*z* transition to that state from the current output state. When all the arcs in the current lexicon state have been processed, move on to the next unprocessed output state. Iterate until finished.

Special provisions have to be made for *x*:*y* arcs in the lexicon state and for *y*:*z* arcs in the rules when *y* is EPSILON. In the former case, we build an *x*:EPSILON transition to the output state associated with the destination and the unchanged configuration; in the latter case we build an EPSILON:*z* arc to the output state associated with the current lexicon state and the new destination configuration.

If there is only one rule transducer, intersecting composition is equivalent to ordinary composition. If all the networks are simple finite-state networks rather than transducers, the operation reduces to *n*-way intersection in which one of the networks (the lexicon in our case) is used to drive the process. This special case of the algorithm was discovered independently by Pasi Tapanainen (Tapanainen, 1991; Koskenniemi, Tapanainen and Voutilainen, 1992) as an efficient way to apply finite-state constraints for syntactic disambiguation. The distinguished network in his application represents alternative interpretations of the sentence. The method allows the sentence to be disambiguated without computing the large (possibly uncomputable) intersection of the rule automata.

Fig. 7 shows a logically equivalent but a more efficient way to perform the computations in Fig. 2 using intersecting composition (designated by **&o**).

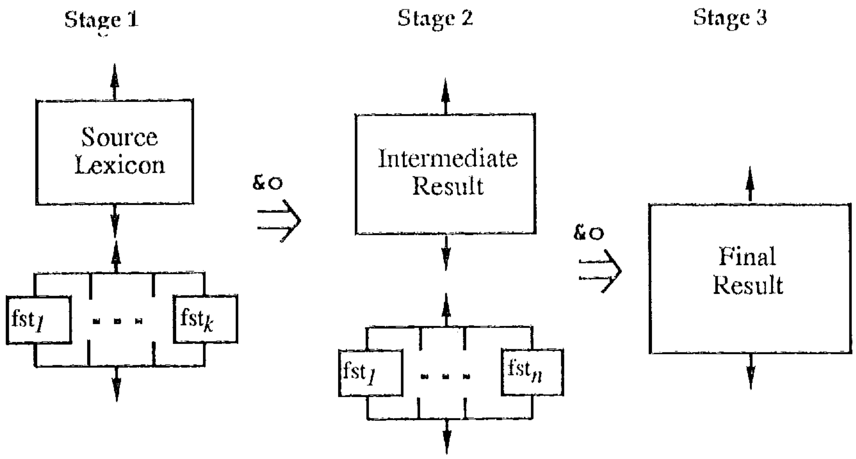

Fig. 7. Construction of a lexical transducer with intersecting composition.

In Fig. 7, the first set of rules applies to the source lexicon, the second set of rules modifies the outcome of the first stage to yield the final outcome. The rules are not intersected separately. The Xerox lexicon compiler (Karttunen 1993) is designed to carry out the computations in this optimal way.

REFERENCES

Antworth, E. L (1990). *PC-KIMMO: a two-level processor for morphological analysis.* Summer Institute of Linguistics, Dallas, Texas.

Kaplan, Ronald M. and Martin Kay (1994). Regular Models of Phonological Rule Systems. To appear in *Computational Linguistics.*

Karttunen, Lauri, Ronald M. Kaplan and Annie Zaenen (1992). Two-Level Morphology with Composition. In the *Proceedings of the fifteenth International Conference on Computational Linguistics. Coling-92.* Nantes, 23-28/8/1992. Vol. I 141-48. ICCL.

Karttunen, Lauri and Kenneth R. Beesley (1992).*Two-Level Rule Compiler* Technical Report. ISTL-1992-2. Xerox Palo Alto Research Center. Palo Alto, California.

Karttunen, Lauri (1993).*Finite-State Lexicon Compiler* Technical Report. ISTL-NLTT-1993-04-02. Xerox Palo Alto Research Center. Palo Alto, California.

Koskenniemi, Kimmo, Pasi Tapanainen and Atro Voutilainen (1992). Compiling and Using Finite-State Syntactic Rules. In the *Proceedings of the fifteenth International Conference on Computational Linguistics. Coling-92.* Nantes, 23-28/8/1992. Vol. I 156-62. ICCL. 1992.

Kwon, Hyuk-Chul and Lauri Karttunen (1994). Incremental Construction of a Lexical Transducer for Korean. In the *Proceedings of the 15th International Conference on Computational Linguistics.* Coling-94. Kyoto, Aug. 5-9, 1994.

Laaksonen, Kaino and Anneli Lieko (1988). *Suomen kielen äänne- ja muoto-oppi* (Phonology and Morphology of Finnish). Oy Finn Lectura Ab. Loimaa.

Tapanainen, Pasi (1991). *Äärellisinä automaatteina esitettyjen kielioppisääntöjen soveltaminen luonnollisen kielen jäsentäjässä* (Applying finite-state grammar rules in natural language parsing). Master's Thesis. Department of Computer Science, University of Helsinki.