Transforming Lexica as Trees

Mark-Jan Nederhof
University of St Andrews
North Haugh, St Andrews
KY16 9SX
Scotland

Abstract

We investigate the problem of structurally changing lexica, while preserving the information. We present a type of lexicon transformation that is complete on an interesting class of lexica. Our work is motivated by the problem of merging one or more lexica into one lexicon. Lexica, lexicon schemas, and lexicon transformations are all seen as particular kinds of trees.

1 Introduction

A standard for lexical resources, called Lexical Markup Framework (LMF), has been developed under the auspices of ISO (Francopoulo et al., 2006). At its core is the understanding that most information represented in a lexicon is hierarchical in nature, so that it can be represented as a tree. Although LMF also includes relations between nodes orthogonal to the tree structure, we will in this paper simplify the presentation by treating only purely tree-shaped lexica.

There is a high demand for tools supporting the merger of a number of lexica. A few examples of papers that express this demand are Chan Ka-Leung and Wu (1999), Jing et al. (2000), Monachini et al. (2004) and Ruimy (2006). A typical scenario is the following. The ultimate goal of a project is the creation of a single lexicon for a given language. In order to obtain the necessary data, several field linguists independently gather lexical resources. Despite efforts to come to agreements before the start of the field work, there will generally be overlap in the scope of the respective resources and there are frequently inconsistencies both in the lexical information itself and in the form in which information is represented. In the latter case, the information needs to be restructured as part of the process of creating a single lexicon.

We have developed a model of the merging process, and experiments with an implementation are underway. The actions performed by the tool are guided by a linguist, but portions of the work may also be done purely mechanically, if this is so specified by the user. The purpose of the present paper is to study one aspect of the adequacy of the model, namely the restructuring of information, with one input lexicon and one output lexicon. This corresponds to a special use of our tool, which may in general produce one output lexicon out of any number of input lexica.

As our lexica are trees, the use of well-established techniques such as term unification (Lloyd, 1984) and tree transduction (Fülöp and Vogler, 1998) seem obvious candidates for solutions to our problem. Also technologies such as XSL (Harold and Means, 2004) and XQuery (Walmsley, 2007) spring to mind. We have chosen a different approach however, which, as we will show, has favourable theoretical properties.

The structure of this paper is as follows. The type of lexicon that we consider is formalized in Section 2, and lexicon transformations are discussed in Section 3. Section 4 shows that the proposed type of lexicon transformation suffices to map all 'reasonable' lexica to one another, as long as they contain the same information. Conditions under which transformations preserve information are discussed in Section 5. A brief overview of an implementation is given in Section 6.

2 Lexica and their structures

In this section, we formalize the notions of lexica, lexicon structures, and their meanings, abstracting away from details that are irrelevant to the discussion that follows.

A *lexicon schema* S is a tuple (A, C, T), where A is a finite set of *attributes*, C is a finite set of *components* ($A \cap C = \emptyset$), and T is a labelled, unordered tree such that:

- each leaf node is labelled by an element from A,
- each non-leaf node is labelled by an element from C, and
- each element from $A \cup C$ occurs exactly once.

A *lexicon* L is a tuple (A, V, C, t), where A is as above, V is a set of *values*, C is as above, and t is a labelled, unordered tree such that:

- each leaf node is labelled by an element from $A \times V$,
- each non-leaf node is labelled by an element from C,
- if a leaf node with a label of the form (a, v_1) has a parent labelled c, then *each* leaf node with a label of the form (a, v_2) has a parent labelled c, and
- if a non-leaf node labelled c_1 has a parent labelled c_2, then *each* non-leaf node labelled c_1 has a parent labelled c_2.

Due to the last two constraints, we may compare lexica and lexicon schemata. In order to simplify this comparison, we will assume that in a lexicon, A and C only contain elements that occur in t. This is without loss of generality, as unused elements of A and C can be omitted. We will also assume that t contains at least two nodes, so that the root is not a leaf.

We say a lexicon $L = (A_L, V, C_L, t)$ is an *instance* of lexicon schema $S = (A_S, C_S, T)$ if $A_L \subseteq A_S$, $C_L \subseteq C_S$, and furthermore:

- the label of the root of t equals the label of the root of T,
- if a leaf node of t with a label of the form (a, v_1) has a parent labelled c, then the leaf node of T labelled a has a parent labelled c, and
- if a non-leaf node of t labelled c_1 has a parent labelled c_2, then the non-leaf node of T labelled c_1 has a parent labelled c_2.

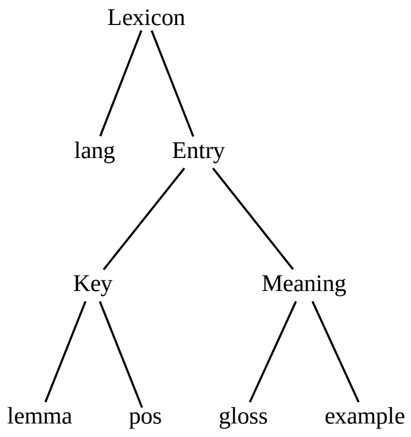


Figure 1: A lexicon schema S.

Examples of a lexicon schema and a lexicon are given in Figures 1 and 2. For the sake of succinctness, an attribute-value pair such as $($example, 'Er ist mit dem Zug gefahren'$)$ is commonly separated by $=$, and where it is required for graphical reasons, the value may be drawn beneath the attribute, stretched out vertically.

On a number of occasions in the constructions and proofs that follow, it is convenient to assume that the root node of a lexicon schema has exactly one child. If this does not hold, as in the running example, we may introduce an artificial root node labelled by an artificial component, denoted by '\$', which has the conceptual root node as only child. We will refer to the lexicon schema that results as an *extended* lexicon schema. (Cf. the theory of context-free grammars, which are often extended with a new start symbol.) As a consequence, a lexicon that is an instance of an extended lexicon schema may, in pathological cases, have several nodes that are labelled by the conceptual root component of the schema.

The components in lexicon schemata and lexica provide a means of structuring sets of attributes, or sets of attribute-value pairs respectively, into tree-shaped forms. The discussion that follows will treat components and structure as secondary, and will take attributes and attribute-value pairs as the primary carriers of information.

A *lexicon base* B is a tuple (A, V, I), where A and V are as above, and I is a finite non-empty set of *items*, each of which is a partial function from A to V, defined on at least one attribute. Such partial functions will also be represented as non-empty sets of attribute-value pairs, in which each attribute occurs at most once.

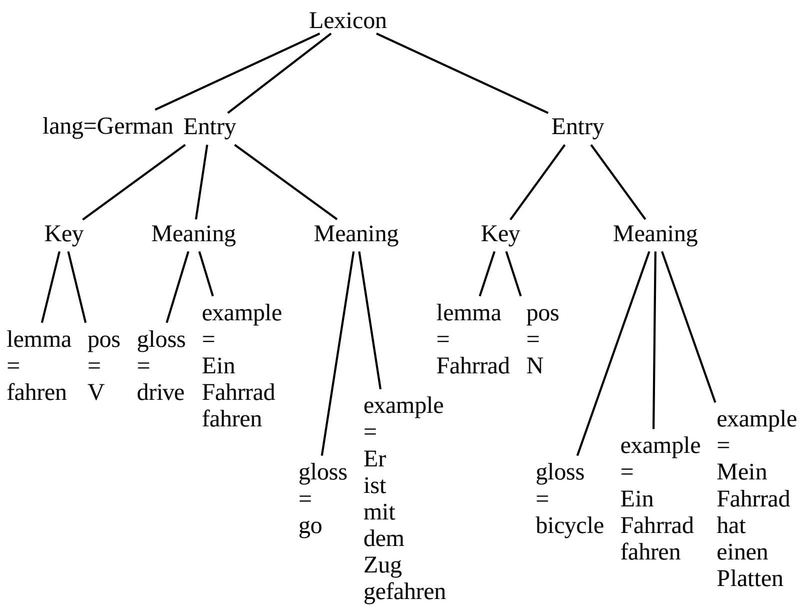


Figure 2: A lexicon L that is an instance of S from Figure 1.

Let $L = (A, V, C, t)$ be a lexicon, where r is the root of t. Its base, denoted by $B(L)$, is (A, V, I) with $I = I(r)$, where the function I on nodes n of the lexicon is defined as follows.

- For a leaf node n labelled by the attribute-value pair (a, v), $I(n) = \{\{(a, v)\}\}$. In words, the set $I(n)$ contains only one item, which is a partial function mapping attribute a to value v.

- For a non-leaf node n, assume that m different components or attributes $d_1, \ldots, d_m$ occur among the children. (Each element d is either a component or an attribute.) Let N_j $(1 \leq j \leq m)$ be the set of children of n labelled by d_j if d_j is a component or by (d_j, v), some value v, if d_j is an attribute. Then:

$$I(n) = \{\iota_1 \cup \cdots \cup \iota_m \mid n_1 \in N_1, \ldots, n_m \in N_m, \iota_1 \in I(n_1), \ldots, \iota_m \in I(n_m)\}.$$

Note that by the definition of lexica and of $N_1, \ldots,$ N_m, no attribute may occur both in ι_i and in ι_j if $i \neq j$. This means that $\iota_1 \cup \cdots \cup \iota_m$ is a partial function as required.

For the lexicon of the running example, the base is:

{ {lang=German, lemma=fahren, pos=V,
gloss=drive,
example=Ein Fahrrad fahren},
{lang=German, lemma=fahren, pos=V,
gloss=go,
example=Er ist mit dem Zug gefahren},
{lang=German, lemma=Fahrrad, pos=N,
gloss=bicycle,
example=Ein Fahrrad fahren},
{lang=German, lemma=Fahrrad, pos=N,
gloss=bicycle,
example=Mein Fahrrad hat einen Platten} }.

There are many different lexica however that share the same base. This is illustrated by Figure 3. We see that the information is presented in an entirely different fashion, with a focus on the examples.

In a lexicon such as that in Figure 2, there may be nodes labelled 'Meaning' without any children corresponding to attribute 'example'. This means that there would be items ι in $B(L)$ such that $\iota(\text{example})$ is undefined. For some of the constructions and proofs below, it is convenient to circumvent this complication, by assuming special 'null' values for absent leaf nodes for attributes. As a result, we may treat an item as a complete function rather than as a partial function on the domain A.

There is a certain resemblance between the base of a lexicon and the disjunctive normal form of a logical expression, the attribute-value pairs taking the place of propositional variables, and the items

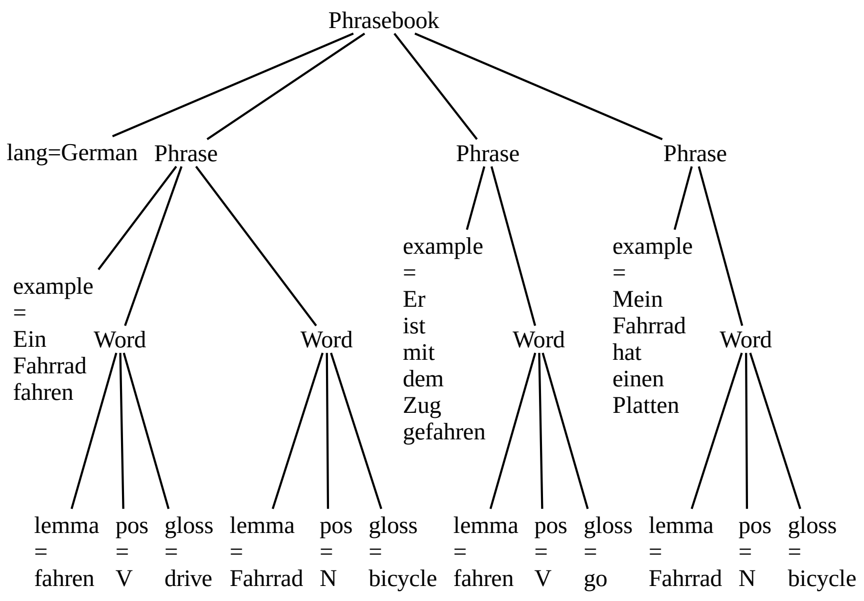


Figure 3: A lexicon L' with the same base as the one in Figure 2.

taking the place of conjunctions. Thus our semantic interpretation of lexica is such that two siblings in the tree are regarded as alternatives (disjunction) if their labels contain the same attribute or component, and they are regarded as joint information (conjunction) if their labels contain distinct attributes or components.

Theorem 1 *For each lexicon base $B = (A_B, V, I)$ and for each lexicon schema $S = (A_S, C, T)$ with $A_B \subseteq A_S$, there is a lexicon L that is an instance of S and whose base is B.*

Proof Assume the root r of T has only one child r'. (Otherwise, make S extended first.) Let T' be the subtree of T at r'. For each item $\iota \in I$, create a copy of T', denoted by t_ι. At each leaf node of t_ι, supplement the label a with the corresponding value from ι if any; if a does not occur in ι, then remove the leaf node from t_ι. (If the parent of a removed leaf node has no other children, then also remove the parent, etc.) Create a root node, with the same label as r, the children of which are the roots of the respective t_ι. Let the resulting tree be called t. The requirements of the theorem are now satisfied by $L = (A_B, V, C, t)$. ∎

3 Lexicon transformations

As we have seen, the information contained in one lexicon base may be rendered in different structural forms, in terms of lexica. The structure of a lexicon is isolated from its content by means of a lexicon schema. In this section we will address the question how we may formalize transformations from one lexicon schema S_1 to lexicon schema S_2, or more precisely, from one class of lexica that are instances of S_1 to another class of lexica that are instances of S_2. In fact, for the sake of the definitions below, we assume that the input to a transformation is not a lexicon but its base, which contains all the necessary information. (That the actual implementation mentioned in Section 1 may often avoid expansion to the base need not concern us here.)

A *lexicon transformation* R is a tuple (A, C, τ), where A is a finite set of attributes as before, C is a finite set of components as before, and τ is a labelled, unordered tree such that:

- each leaf node is labelled by an element from A,
- the root node is labelled by an element from C,
- each internal node is either labelled by an element from C, or by a subset of A,
- each element from $A \cup C$ occurs exactly once as a label by itself,
- each element from A occurs exactly once in a label that is a subset of A, and
- each node ν labelled by a set $\{a_1, \ldots, a_k\} \subseteq A$ has exactly one child, which is labelled

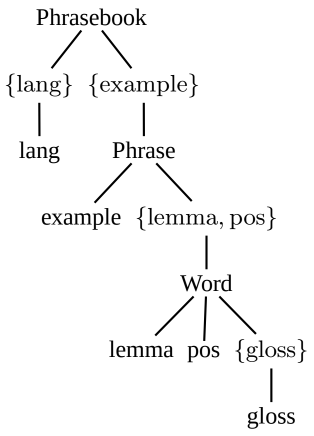
by an element from $A \cup C$, and the leaves labelled $a_1, \ldots, a_k$ are each descendants of ν.

A lexicon transformation is very similar to a lexicon schema, except for the extra nodes labelled by sets $A' \subseteq A$ of attributes, which we refer to as *restrictors*. Such a node indicates that for the purpose of the subtree, one should commit to particular subsets of the input lexicon base. Each such subset is determined by a choice of a fixed value for each attribute in A'.

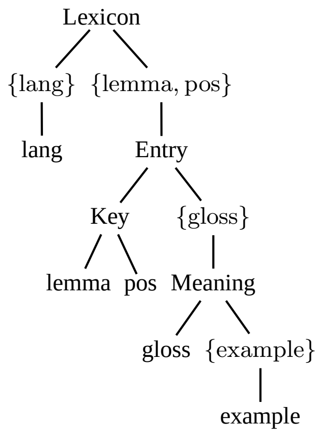
As an example, consider the lexicon transformations in Figure 4(a) and (b). If we omit the nodes labelled by restrictors, then we obtain a lexicon schema. In the case of (b), this is the lexicon schema in Figure 1. In Figure 4(a), the node labelled $\{\mathrm{example}\}$ means that the transformation takes one non-empty subset of the base for each possible value of attribute 'example'. For each subset, one node labelled 'Phrase' is generated in the target lexicon. At the node labelled $\{\mathrm{lemma}, \mathrm{pos}\}$, the subset of the base is further restricted, and for each combination of a value of 'lemma' and a value of 'pos' in the current subset of items, a node labelled 'Word' is generated. If the base contains several glosses for one choice of 'example', 'lemma' and 'pos', each such gloss leads to a separate leaf node.

The meaning of a lexicon transformation is formally expressed in Figure 5. A call $lexicon(\nu, I')$, where ν is a node of τ and I' is a subset of I from the input lexicon base $B = (A, V, I)$, returns a set of nodes. The function is recursive, in that the value of $lexicon(\nu, I')$ is expressed in terms of values of $lexicon(\nu', I'')$ for child nodes ν' of ν and subsets I'' of I'. The main purpose is the computation of $lexicon(\rho, I)$, where ρ is the root of τ. As ρ is labelled by an element from C, $lexicon(\rho, I)$ is by definition a singleton set $\{r\}$, with r becoming the root of the resulting lexicon.

Note that the placement of restrictors is critical. For example, if we were to move up the restrictor $\{\mathrm{gloss}\}$ in Figure 4(b) to merge with the restrictor $\{\mathrm{lemma}, \mathrm{pos}\}$, this would result in one entry for each combination of 'lemma', 'pos' and 'gloss', and in each entry there would be at most one meaning. It is not apparent that such a choice would be less appropriate than the choice we made in Figures 2 and 4(b). However, if we were to move down the node labelled $\{\mathrm{gloss}\}$ to become a child of the node labelled 'Meaning' and a parent of the leaf node labelled 'gloss', then we would lose the coupling between glosses and examples, which seems undesirable. This observation underlies much of the development in Section 5.



(a)



(b)

Figure 4: Two lexicon transformations: (a) is capable of mapping the base of lexicon L (Figure 2) to lexicon L' (Figure 3), and (b) is capable of the reverse mapping.

4 Completeness

Next, we investigate whether the lexicon transformations as we defined them are powerful enough to produce 'reasonable' lexica starting from a lexicon base. As unreasonable, we reject those lexica that contain information that cannot be expressed in terms of a base. This concerns siblings in the tree with the same component label. How many siblings with the same component should be generated can be deduced from the base, provided we may assume that there is a combination of attribute values that distinguishes one sibling from another.

```
$lexicon(\nu, I')$ :
    if the label of $\nu$ is $a \in A$
        let $v$ be the (only) value such that $\exists \iota \in I'[\iota(a) = v]$
        create a new node $n$ with label $(a, v)$
        return $\{n\}$
    else if the label of $\nu$ is $c \in C$
        let the children of $\nu$ be $\nu_1, \ldots, \nu_m$
        create a new node $n$ with label $c$ and children $\bigcup_{1 \leq i \leq m} lexicon(\nu_i, I')$
        return $\{n\}$
    else if the label of $\nu$ is $A' = \{a_1, \ldots, a_k\} \subseteq A$
        let the only child of $\nu$ be $\nu'$
        let $\mathbf{I}$ be the set of all $I''$ such that there is a combination of
            $v_1, \ldots, v_k \in V$ with $I'' = \{\iota \in I' \mid \iota(a_1) = v_1, \ldots, \iota(a_k) = v_k\} \neq \emptyset$
        return $\bigcup_{I'' \in \mathbf{I}} lexicon(\nu', I'')$
```

Figure 5: The meaning of a lexicon transformation, as a recursive function. The return value is a set of nodes that are created. The main application is $lexicon(\rho, I)$, where ρ is the root of τ and I is taken from the input lexicon base.

We call such a combination of attributes a *key*.

Formally, a *key mapping* for a lexicon schema (A, C, T) is a function f that maps each component from C to a subset of A, subject to the following restrictions. Let c be a component and let n be the node of T that is labelled by c. Then for each attribute a in key $f(c)$, the leaf node of T that is labelled by a should be a descendant of n. The component that occurs as label of the root of T is always mapped to the empty set of attributes, and may be ignored in the following discussion.

Let lexicon $L = (A_L, V, C_L, t)$ be an instance of schema $S = (A_S, C_S, T)$. We say that L *satisfies* the key mapping f for S if:

1. among the leaves, there is no pair of distinct siblings in t with identical labels, and
2. for each maximal set $\{n_1, \ldots, n_m\}$ of siblings in t labelled by the same component c, with $f(c) = \{a_1, \ldots, a_k\}$, we have that for each i ($1 \leq i \leq m$), there is a distinct combination of values $v_1, \ldots, v_k \in V$ such that:

$$I(n_i) = \{\iota \in \bigcup_{1 \leq j \leq m} I(n_j) \mid \begin{array}{l} \iota(a_1) = v_1, \ldots, \\ \iota(a_k) = v_k\}. \end{array}$$

The second condition states that the total set of items coming from all siblings with the same label c is partitioned on the basis of distinct combinations of values for attributes from the key, and the subsets of the partition come from the respective siblings.

Returning to the running example, the lexicon L in Figure 2 satisfies the key mapping f given by:

$$\begin{array}{lcl} f(\text{Lexicon}) & = & \emptyset \\ f(\text{Entry}) & = & \{\text{lemma}, \text{pos}\} \\ f(\text{Key}) & = & \emptyset \\ f(\text{Meaning}) & = & \{\text{gloss}\} \end{array}$$

A different key mapping exists for the lexicon L' in Figure 3.

If n_1 and n_2 are two distinct nodes in the tree T of schema S, with labels c_1 and c_2, respectively, then we may assume that $f(c_1)$ and $f(c_2)$ are disjoint, for the following reason. Suppose that the intersection of $f(c_1)$ and $f(c_2)$ includes an attribute a, then n_1 must be a descendant of n_2 or vice versa, because the leaf labelled a must be a descendant of both n_1 and n_2. Assume that n_1 is a descendant of n_2. As the base is already restricted at n_1 to items ι with $\iota(a) = v$, for certain v, a may be omitted from $f(c_2)$ without changing the semantics of the key mapping. This observation is used in the construction in the proof of the following.

Theorem 2 *Let lexicon $L = (A_L, V, C_L, t)$ be an instance of schema $S = (A_S, C_S, T)$, satisfying key mapping f. Then there is a lexicon transformation that maps $B(L)$ to L.*

Proof The required lexicon transformation is constructed out of T and f. We insert an additional restrictor node just above each non-leaf node labelled c, and as the restrictor we take $f(c)$.

(If $f(c) = \emptyset$, we may abstain from adding a restrictor node.) If an attribute a does not occur in $f(c)$, for any $c \in C_S$, then we add a restrictor node with set $\{a\}$ just above the leaf node labelled a. The result is the tree τ of a lexicon transformation $R = (A_S, C_S, \tau)$.

It is now straightforward to prove that R maps $B(L)$ to L, by induction on the height of T, on the basis of the close similarity between the structure of T and the structure of τ, and the close link between the chosen restrictors and the keys from which they were constructed. ∎

For the running example, the construction in the proof above leads to the transformation in Figure 4(b).

Theorem 2 reveals the conditions under which the structure of a lexicon can be retrieved from its base, by means of a transformation. Simultaneously, it shows the completeness of the type of lexicon transformation that we proposed. If a lexicon L is given, and if an alternative lexicon L' with $B(L') = B(L)$ exists that is an instance of some schema S and that is 'reasonable' in the sense that it satisfies a key mapping for S, then L' can be effectively constructed from L by the derived transformation.

5 Consistency

We now investigate the conditions under which a lexicon transformation preserves the base. The starting point is the observation at the end of Section 3, where we argued that if a restrictor is chosen too low in the tree τ relative to other restrictors, then some necessary dependence between attribute values is lost. Note that the proof of Theorem 1 suggests that having only one restrictor with all attributes at the root of the tree always preserves the base, but the result would be unsatisfactory in practice.

For a set A of attributes, we define an *independence system* D as a set of triples (A_1, A_2, A_3) where A_1, A_2, $A_3 \subseteq A$ and $A_1 \cap A_2 = \emptyset$. We pronounce $(A_1, A_2, A_3) \in D$ as 'A_1 and A_2 are independent under A_3'. It should be noted that A_3 may overlap with A_1 and with A_2.

We say a lexicon base (A, V, I) *satisfies* D if for each $(A_1, A_2, A_3) \in D$ with $A_1 = \{a_{1,1}, \ldots a_{1,k_1}\}$, $A_2 = \{a_{2,1}, \ldots a_{2,k_2}\}$, $A_3 = \{a_{3,1}, \ldots a_{3,k_3}\}$, and for each combination of values $v_{1,1}, \ldots, v_{1,k_1}, v_{2,1}, \ldots, v_{2,k_2}, v_{3,1}, \ldots, v_{3,k_3}$, we have:

$$\begin{aligned}
&\exists \iota \in I[\iota(a_{1,1}) = v_{1,1} \wedge \ldots \wedge \iota(a_{1,k_1}) = v_{1,k_1} \wedge \\
&\qquad \iota(a_{3,1}) = v_{3,1} \wedge \ldots \wedge \iota(a_{3,k_3}) = v_{3,k_3}] \wedge \\
&\exists \iota \in I[\iota(a_{2,1}) = v_{2,1} \wedge \ldots \wedge \iota(a_{2,k_2}) = v_{2,k_2} \wedge \\
&\qquad \iota(a_{3,1}) = v_{3,1} \wedge \ldots \wedge \iota(a_{3,k_3}) = v_{3,k_3}] \\
&\Longrightarrow \\
&\exists \iota \in I[\iota(a_{1,1}) = v_{1,1} \wedge \ldots \wedge \iota(a_{1,k_1}) = v_{1,k_1} \wedge \\
&\qquad \iota(a_{2,1}) = v_{2,1} \wedge \ldots \wedge \iota(a_{2,k_2}) = v_{2,k_2} \wedge \\
&\qquad \iota(a_{3,1}) = v_{3,1} \wedge \ldots \wedge \iota(a_{3,k_3}) = v_{3,k_3}].
\end{aligned}$$

The intuition is that as long as the values for A_3 are fixed, allowable combinations of values for $A_1 \cup A_2$ in I can be found by looking at A_1 and A_2 individually.

We say that a lexicon transformation $R = (A, C, \tau)$ is *allowed* by an independence system D if the following condition is satisfied for each node ν in τ that is labelled by a component c and a node ν' that is its child: Let A_1 be the set of attributes at leaves that are descendants of ν', and let A_2 be the set of attributes at leaves that are descendants of the other children of ν. Let A_3 be the union of the restrictors at ancestors of ν. Now (A_1, A_2, A_3) should be in D.

Theorem 3 *If a lexicon base $B = (A, V, I)$ satisfies an independence system D, if a lexicon transformation R is allowed by D, and if R maps B to lexicon L, then $B(L) = B$.*

The proof by induction on the height of τ is fairly straightforward but tedious.

In the running example, there are a number of triples in D but most are trivial, such as $(\emptyset, \{\text{gloss}, \text{example}\}, \{\text{lemma}, \text{pos}\})$. Another triple in D is $(\{\text{lang}\}, \{\text{lemma}, \text{pos}, \text{gloss}, \text{example}\}, \emptyset)$, but only because we assume in this example that one lexicon is designed for one language only. In general, there will be more interesting independency, typically if a lexical entry consists of a number of unconnected units, for example one explaining syntactic usage of a word, another explaining semantic usage, and another presenting information on etymology.

The implication of Theorem 3 is that transformations between lexica preserve the information that they represent, as long as the transformations respect the dependencies between sets of attributes. Within these bounds, an attribute a may be located in a restrictor in τ anywhere between the root node and the leaf node labelled a.

6 Implementation

The mathematical framework in this paper models a restricted case of merging and restructuring a number of input lexica. An implementation was developed as a potential new module of LEXUS, which is a web-based tool for manipulating lexical resources, as described by Kemps-Snijders et al. (2006).

The restriction considered here involves only one input lexicon, and we have abstracted away from a large number of features present in the actual implementation, among which are provisions to interact with the user, to access external linguistic functions (e.g. morphological operations), and to rename attributes. These simplifications have allowed us to isolate one essential and difficult problem of lexicon merging, namely how to carry over the underlying information from one lexicon to another, in spite of possible significant differences in structure.

The framework considered here assumes that during construction of the target lexicon, the information present in the source lexicon is repeatedly narrowed down by restrictors, as explained in Section 3. Each restrictor amounts to a loop over all combinations of the relevant attribute values from the currently considered part of the source lexicon.

Let us consider a path from the root of the lexicon transformation to a leaf, which may comprise several restrictors. The number of combinations of attribute values considered is bounded by an exponential function on the total number of attributes contained in those restrictors. Motivated by this consideration, we have chosen to regard a lexicon transformation as if its input were an expanded form of the source lexicon, or in other words, a lexicon base.

However, in terms of the actual implementation, the discussed form of restrictors must be seen as a worst case, which is able to realize some of the most invasive types of restructuring. Next to restrictors that select combinations of attribute values, our lexicon transformations also allow primitives that each represent a loop over all *nodes* of the presently considered part of the source lexicon that are labelled by a chosen component or attribute. By using only such primitives, the time complexity remains polynomial in the size of the input lexicon and the size of the input lexicon transformation. This requires an implementation that does not expand the information contained in a source lexicon in terms of a lexicon base. A full description of the implementation would go beyond the context of this paper.

7 Conclusions

We have introduced a class of lexicon transformations, and have shown interesting completeness and consistency properties.

The restrictors in our lexicon transformations are able to repeatedly narrow down the information contained in the source lexicon based on attribute values, while constructing the target lexicon from the top down. Existing types of tree manipulations, such as tree transducers, do not possess the ability to repeatedly narrow down a *set* of considered nodes scattered throughout a source structure, and therefore seem to be incapable of expressing types of lexicon transformations allowing the completeness results we have seen in this paper.

One could in principle implement our lexicon transformations in terms of technologies such as XQuery and XSLT, but only in the sense that these formalisms are Turing complete. Our restrictors do not have a direct equivalent in these formalisms, which would make our type of lexicon transformation cumbersome to express in XQuery or XSLT. At the same time, their Turing completeness makes XQuery and XSLT too powerful to be of practical use for the specification of lexicon transformations.

A tentative conclusion seems to be that our class of lexicon transformations has useful properties not shared by a number of existing theories involving tree manipulations. This justifies further study.

Acknowledgements

This work was done while the author was employed at the Max Planck Institute for Psycholinguistics. The work was motivated by suggestions from Peter Wittenburg and Marc Kemps-Snijders, whose input is gratefully acknowledged.

References

D. Chan Ka-Leung and D. Wu. 1999. Automatically merging lexicons that have incompatible part-of-speech categories. In *Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, pages 247–257, University of Maryland, USA, June.

G. Francopoulo, N. Bel, M. George, N. Calzolari, M. Monachini, M. Pet, and C. Soria. 2006. Lexical markup framework (LMF) for NLP multilingual resources. In *Proceedings of the Workshop on Multilingual Language Resources and Interoperability*, pages 1–8, Sydney, Australia, July.

Z. Fülöp and H. Vogler. 1998. *Syntax-Directed Semantics: Formal Models Based on Tree Transducers*. Springer, Berlin.

E.R. Harold and W.S. Means. 2004. *XML in a Nutshell*. O'Reilly.

H. Jing, Y. Dahan Netzer, M. Elhadad, and K.R. McKeown. 2000. Integrating a large-scale, reusable lexicon with a natural language generator. In *Proceedings of the First International Conference on Natural Language Generation*, pages 209–216, Mitzpe Ramon, Israel, June.

M. Kemps-Snijders, M.-J. Nederhof, and P. Wittenburg. 2006. LEXUS, a web-based tool for manipulating lexical resources. In *LREC 2006: Fifth International Conference on Language Resources and Evaluation, Proceedings*, pages 1862–1865.

J.W. Lloyd. 1984. *Foundations of Logic Programming*. Springer-Verlag.

M. Monachini, F. Calzolari, M. Mammini, S. Rossi, and M. Ulivieri. 2004. Unifying lexicons in view of a phonological and morphological lexical DB. In *LREC 2004: Fourth International Conference on Language Resources and Evaluation*, pages 1107–1110, Lisbon, Portugal, May.

N. Ruimy. 2006. Merging two ontology-based lexical resources. In *LREC 2006: Fifth International Conference on Language Resources and Evaluation, Proceedings*, pages 1716–1721.

P. Walmsley. 2007. *XQuery*. O'Reilly.