DERIVATION AND THE TWO-LEVEL MODEL

Kristiina Jokinen

Research Unit for Computational Linguistics
University of Helsinki
Hallituskatu 11
SF-00100 Helsinki

Artikkeli käsittelee leksikaalisten sääntöjen esittämiseen kehittämääni formalismia, joka voidaan ymmärtää Koskenniemen (1983) kaksitasomorfologian laajennukseksi syntaksiin ja semantiikkaan päin. Formalismin avulla voidaan käsitellä leksikaalisten entryjen sisältämää syntaktis-semanttista tietoa, erityisesti sananmuodostuksessa tarvittavia rajoituksia ja piirteiden periytymistä. Entryjen syntaktis-semanttiset piirteet on koodattu entryihin templaatteina, ja leksikaaliset säännöt määrittävät entryjen väliset suhteet templaatti-vastaavuuksina.

1. Introduction

The paper presents a formalism to deal with syntactic and semantic restrictions in word-formation, especially with those found in derivation. The formalism is based on Jokinen (ms.), and its aim is to provide an extension to the finite-state morphophonology of Koskenniemi's Twol-Level Model (1983). Each lexical entry, i.e. a morpheme string, is assigned a set of templates that encode its syntactic and semantic properties, and a notion of lexical rule is introduced to determine correspondences between templates of the entries that stand in a lexical relation. Application of a rule can be implemented as a finite-state transducer.

2. The Two-Level Model and Derivation

Derivation is governed by formal and lexical restrictions. The former deal with morphophonological constraints, the latter with syntactic and semantic compatibility of the derivation. The surface form of a derived word is determined by the morphophonological rules of the grammar.

In the Two-Level Model (TWOL), formal constraints are described by continuation classes that determine possible continuations from a morpheme. However, there are two sources of overgeneration in TWOL. First, a continuation class can refer back to itself, and thus recursive morpheme strings are accepted (e.g. *hae+t+ut+ut+ut+utta* 'fetch+CUR+CUR+CUR+CUR+CUR'). Second, continuations based on morphophonological similarities fail to distinguish between entries that have the same inflectional properties, but due to semantics, differ in their derivational possibilities (e.g. stative 3-syllabic TA-verb *vilutta* 'feel cold' does not have a causative-curative derivative **vilut+ utta* 'make someone feel cold', though the activity verb of the same morphological type, *asetta* 'put', has a regular curative form *aset+utta* 'make someone put').

To deal with the lexical constraints, we propose a new type of rule, a lexical rule, which operates on the syntactic and semantic information encoded in the entries. Input for a rule consists of lexical entries, i.e. morpheme strings, and the rule determines bi-directional relations between the entries by relating their morphosyntactic and semantic information. Lexical rules are separate from the morphophonological ones, and they transmit the information encoded in the morphemes to word-forms used in the syntactic analysis.

Phonological realization of a morpheme string is taken care of by TWOL. Lexical representation is mapped to the correct surface representation as discussed in Koskenniemi (1983). Well-formedness of a string is automatically guaranteed in TWOL, and thus we avoid the completeness problem described by Calder & te Lindert (1987). The overall picture of TWOL and the proposed extension is presented in Figure 1.

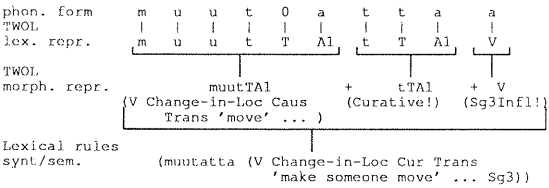

Figure 1.

3. The Two-Level Lexicon

Lexical entries in the TWOL lexicon are morpheme strings. Strings of length one are the stems and the affixes of the language, and their concatenations correspond to word-forms of the language. Lexical entries are recursively defined as follows:

> If A and B are lexical entries, then A+B is a lexical entry, where + marks the concatenation of the two entries governed by a lexical rule.

A lexical entry includes information about its morphophonological form (P), combinatorics (C) and syntactic and semantic properties (S). It is represented as a triple of the form:

< P, C, S >.

Morphophonological form is also the lexical representation of an entry, and it encodes e.g. morphophonological alternations. Combinatorics refers to concatenation of the morphemes. Each entry is assigned a continuation class that determines the set of affixes that can be concatenated to the entry in question. Continuation classes determine morpheme order and they also encode morphophonological selection between a stem and a suffix.

Syntactic and semantic properties associated with an entry are encoded into a set of template names. A template is an abbreviation for a body of information, and it refers either to an atomic feature or to a complex feature structure (cf. Karttunen 1986). A template is referred to by its name.[1] Lexical rules treat template names as atomic entities, and the internal structure of the templates is not 'seen' at the level of word-formation. For syntactic purposes, however, the names can be compiled into representations structured for the analysis at that level. This indeterminacy in the interpretation of the template names is an indication of the flexibility of the lexicon: the same data base can be interpreted in several ways.

The templates are of two types: *feature templates* (f-templates) encode syntactic and semantic information, and *operational templates* (o-templates) encode lexical rules (cf. templates and lexical rules in DPATR, Karttunen 1986). The two types are formally distinguished by an exclamation mark at

---

1 However, we will often use the short term template instead of template name. Because lexical rules operate on template names only, so no misunderstanding is possible.

the end of the name of the o-template. An entry may have several f-templates, but an o-template always appears alone. F-templates can also subsume other f-templates, and thus template names have implication relations like *AgSubj* > *Agentivity*, i.e. if an entry has the feature 'agentive subject', it also has the feature 'agentive'.

Sample entries are given in Figures 2 and 3. Figure 2 presents the Finnish verb entry for *muutta* 'move, change, turn into', and Figure 3 the continuation class *A/V*. Capital letters mark morphophonemes, and parentheses are used to differentiate between various senses of the same morphophonological form. Template names are to be interpreted as follows:

Trans = transitive, **Intrans** = intransitive, **Caus** = causative, **AgSubj** = subject-argument with the feature 'agentive', **PathArg** = argument referring to a path moved from one place to another (Jackendoff 1983), **ChangeArg** = argument referring to a change from one state to another, **Change-in-Loc** = verb class that includes verbs denoting change in location, **Change-in-State** = verb class that includes verbs denoting change in state. The semantics of the entries is expressed as an English translation between the quotes, and the template **Ftrs** is to remind that the template description is only partial. The o-templates encode lexical rules that are used to form curative, passive, reflexive and frequentative verb forms.

```
muutT  A/V  "((V Change-in-Loc Caus AgSubj PathArg
                    ((Trans 'change, move' Ftrs)
                      (Intrans 'move house' Ftrs))
              (V Change-in-State Trans Caus ChangeArg 'turn into' Ftrs))";
```

Figure 2.

```
LEXICON  A/V
    A1       /V;
    A1t      3tA/V      "( (Curative!) )";
    U        /V         "( (U-Passive!) )";
    A1UTU    /V         "( (Reflexive!) )";
    e        (e)le/V    "( (Frequentative!) )";
```

Figure 3.

4. Lexical rules in TWOL

As described in Jokinen (ms.), lexical rules are encoded into affixes and they operate on stems. They determine the relation between a stem and its derivation in terms of template correspondences. A relation is permitted, i.e. an application of a rule is accepted, if each of the template correspondences is accepted. Thus derivation in the sense of deriving one form from another is not included in the formalism, but the description is declarative. A lexical rule is defined as a triple:

< N, I, O >

where N is the name of the rule, I a set of input templates, and O a set of output templates. The input templates refer to the templates of the stem, and the output templates to the templates of the result.[2] Each template name referred to by a rule must be explicitly present in the input template list. If a name is embedded in an implication relation, it must be spelled out before the application of the rule.

The rule determines three kinds of correspondences between the input and output templates. If the input has templates not explicitly mentioned in the rule, these are transferred to the output as such.

Restrictions specify failing (negative) conditions of the rule. The reserved name FAIL is used as the output correspondent of a forbidden input template name. If any of the templates having FAIL as the correspondent appears in the input's template list, application of the rule is blocked.

Operations describe the manipulation of the input information: change in the template interpretation, deletion of a template from the output, and adding of a template that is missing in the input. The two last alternatives related to the reserved template NONE: deletion has NONE in the output, adding in the input. The input templates are considered obligatory (except

---

[2] The definition can be compared to the interpretation of a functor in Hoeksema and Janda (1988): every functor-category is represented as a triple consisting of the argument (input-category), the value (output-category) and the operation performed. However, we regard the operations considered by Hoeksema and Janda (addition, permutation, replacement, subtraction) as operations concerning the morphophonological realization of a rule rather than its lexical functioning that we are interested in.

for NONE): if any of them is missing in the argument, the rule application fails.

Specifications list the templates to be added to the output as an indication of the rule application. An argument may already have specification templates, in which case specifications appear redundant.

Optional correspondences are intended to help rule writing by allowing adjustment of the rule with respect to different inputs: they encode disjunctions of the same rule. They are operations, but the input templates are not considered obligatory: if they are found, the operation indicated is performed, otherwise no action is taken.

No special correspondence type is needed to state necessary (positive) conditions of a rule: this is already expressed by operational correspondences. On the other hand, obligatory templates that do not 'change', are expressed by an identity relation: this kind of operational correspondence guarantees that the input template is mapped as it is onto the output.

None of the correspondence types is obligatory in a rule. However, at least one of them must be present: if there is nothing to say about a relation between two entries, no rule exists at all.[3]

---

[3] As Krister Linden suggested to me, the rule can be formalized with the help of three primary operations Insert, Delete and Exist representing the bit-vector operations bit-and, bit-or and find, respectively. If P, Q, R and S represent correspondences, and Oper is an abbreviation for the Insert, Delete and Change (= Delete and Insert) operations, the rule can be expressed as the following logical formula:

RESTR:		-Exist (P)			
OPER:	&	[Exist (Q)	&	Oper (Q)]	
SPEC:	&	[Exist (R)	v	(-Exist (R) &	Insert (R))]
OPT:	&	[(Exist (S)	&	Oper (S)) v	-Exist (S)]

In other words, restrictions refer to negative existence in the bit vector, operations to existence check and either insertion, deletion or change, specifications to existence check and insertion if not found, and finally, options to existence check and insertion, deletion or change or negative existence.

5. An Example

Below is given a sample rule for the productive U-passivization in Finnish. The morphemes listed in (1a) encode the rule given in (1b). The four types of correspondence are written on separate lines, and abbreviations are used to name the correspondence type.

(1a) {U-PASS} =(/U/, /UTU/, /TU/, (/nTU/, /rTU/, /sTU/)
(1b)

U-PASSIVE!							
RESTR:	AGSUBJ	PASS	EMOTIVE	COMM	STATE	MODAL	WEATHER
	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL
OPER:	CAUS	TRANS		SUBJ	OBJ		
	PASS	INTRANS		NONE	SUBJ		
SPEC:	AUTOM						

The rule says that U-passivization is not possible from verbs that have agentive subjects, are already passives, or belong to emotive, communication, state, modal or weather verbs. On the other hand, the verb must be causative, transitive and have subject and object arguments.[4] The operations map the input template names **causative** and **transitive** onto the output template names **passive** and **intransitive**, the object argument of the input to the subject argument of the output, and delete the subject argument of the input by mapping it to NONE on the output.[5] Finally, the result is specified as having the feature **automative**.[6]

An output of the U-passive! rule applied to the sample verb *muutta* 'move, change; move house; turn into' is given below. Only the sense 'turn into' fulfils the requirements of the rule; the two other senses have agentive subjects. The semantics of the result is not specified.

---

[4] Transitivity of course presupposes the subject and object arguments, but their explicit presence is required because of the argument changing relation that the passivization rule encodes.

[5] Implications of the object-subject-correspondence on the syntactic level (e.g. case marking) are not spelled in the rule, but is part of the syntactic interpretation of the templates OBJ and SUBJ.

[6] This encodes the special meaning of the **U**-passives in Finnish: an event is conceived as automative that takes place without any overt causer. Thus verbs with clearly agentive subjects fail to form **U**-passives, see Jokinen (ms.).

muutTu /V "((V Change-in-State Autom Pass Intrans ChangeArg
'PassOf(turn into)' Ftrs))"

6. Finiteness

For pointing out the finiteness of the formalism, I am grateful to Krister Linden for the following observation. Given the set of templates T, we can construct a power-set of T with $2^{|T|}$ elements. We then construct a non-deterministic finite state transducer with one state for each element in the power-set. A rule in the proposed formalism defines a non-deterministic transition from one state to a set of other states. As such, an equivalent deterministic FST can be constructed for any non-deterministic FST used as an acceptor since a deterministic FSA can always be constructed that accepts exactly the same language as a non-deterministic FSA (Hopcroft & Ullman 1979). In this case, the deterministic FST has $2^{2|T|}$ number of states.[7]

REFERENCES

Calder, J. and E. te Lindert 1987. The Protolexicon: Towards a High-Level Language for Lexical Description. In E. Klein and J. van Benthem (eds.) *Categories, Polymorphism and Unification.* Centre for Cognitive Science, University of Edinburgh, Institute for Language, Logic and Information, University of Amsterdam. 355-370.

---

[7] A rough estimate of the complexity of a minimized FSA is the following (also called to my attention by Krister Linden). Using a bit-vector stack and Insert, Delete and Find operations, it is possible to construct a one-way non-deterministic push-down transducer with a space-time complexity of $O(RN|T|^2S^N)$, where
R = max number of Insert, Delete and Find operations per rule
N = max number of morphemes per word
|T| = number of templates
S = max number of rules per morpheme.

Hoeksema, J. and R.D. Janda 1988. Implications of Process-Morphology for Categorial Grammar. In R.T. Oehrle, E. Bach and D. Wheeler (eds.) *Categorial Grammars and Natural Language Structures*. Dordrecht: D. Reidel Publishing Company. 199-247.

Hopcroft, J. and J.D. Ullman 1979. *Introduction to Automata Theory, Languages and Computation*. Reading, Mass: Addison-Wesley. Jackendoff, R. 1983. Semantics and Cognition. Cambridge, Mass: The MIT Press.

Jokinen, K. ms. Lexicon, Word-Formation and Grammar. Manuscript for a PhD. Thesis. Department of General Linguistics, University of Helsinki.

Karttunen, L. 1986. *D-PATR: A Development Environment for Unification-Based Grammars*. CSLI Report 61, Stanford.

Koskenniemi, K. 1983. *Two-Level Morphology: A General Computational Model for Word-Form Recognition and Production*. Publications 11, Department of General Linguistics, University of Helsinki, Helsinki.