

ISRN SICS-R--92/12--SE

# **Lexical Acquisition: the Swedish VEX System**

by

**Björn Gambäck**

# Lexical Acquisition: the Swedish VEX System

Björn Gambäck

September 1992

SICS research report R92:12  
ISSN 0283-3638

Swedish Institute of Computer Science  
Box 1263  
S-164 28 KISTA  
Stockholm, Sweden

## Abstract

The paper describes S-VEX, the lexical acquisition component of the Swedish Core Language Engine (S-CLE).<sup>1</sup> The S-CLE is a general purpose natural language processing system for Swedish developed from its English counter-part, the SRI Core Language Engine. In parallel with the development of the S-CLE, a Swedish version of the English VEX (Vocabulary EXpander) system was designed. S-VEX allows for the creation of lexicon entries by users with knowledge of an application domain but not of linguistics or of the detailed workings of the system. The approach taken is based on eliciting grammaticality judgments and information of inflected forms interactively from the user. The S-VEX system and the lexicon of the S-CLE is described, as well as the problems of the specific lexical acquisition task and their solutions.

The only “real” linguistic information the user needs to provide is the general category (noun, verb, or adjective) of the new lexical entry. All other information needed to create lexicon entries is obtained from the user’s answers to questions/examples presented to her by the system. For determining the morphological inflections, the system is equipped with knowledge of 41 different inflectional classes.

When constructing the syntactic usage part of the lexical entries, S-VEX is provided with pointers to entries in a “paradigm” lexicon for a number of representative word usages and declarative knowledge of the range of sentential contexts in which these usages can occur. This knowledge is encoded in 58 sentence patterns. The paradigm lexicon of the present system contains 62 different paradigms: 5 for nouns, 10 for adjectives, and all the others for verbs.

---

<sup>1</sup>A shorter version of this paper appears in L. Ahrenberg (ed.): *Papers from the Third Nordic Conference on Text Comprehension in Man and Machine*, Linköping, Sweden, 1992.

## Contents

Abstract	i
Table of contents	ii
1 Introduction	1
2 The S-CLE and its lexicon	3
3 The VEX system	5
4 Morphology	7
5 Syntactic usage	11
6 Eliciting Semantic Information	13
7 Incorporating New Entries; QVEX	14
8 Conclusions	15
9 Acknowledgements	15
References	16

## 1 Introduction

Every natural language processing system intended for real applications must incorporate a large lexicon. The problems associated with acquiring such a lexicon are often neglected or at least underestimated. The task of lexical acquisition has over the years been approached from several different directions, both with tools supporting the user in creating lexical entries and with tools that try to automate the acquisition process completely.

Complete automation of the lexical acquisition task has been the object of a number of projects, based on e.g. extracting information from large machine-readable dictionaries; however, the information available in such dictionaries does in general not contain all the information needed in the complicated lexica of more powerful natural language processing systems. An alternative approach is to try to elicit lexical information automatically from large corpora by some statistical method, as suggested by e.g. [Calzolari & Bindi 1990]. The problem with this method is that even if the corpora are enormous, they may still lack enough data for creation of complete lexical entries for complicated NL systems.

If the lexical acquisition system is designed with a specific application in mind, the task is somewhat simplified. This is the case in the TEAM system [Grosz *et al* 1987], which was designed specifically as a front end for databases of a particular kind. This means that lexical acquisition in TEAM is essentially a matter of determining the English counterparts of particular database relations, and that the possibilities for word behaviors are constrained by the kinds of relations that exist. Furthermore, TEAM's coverage of verb subcategorization is rather limited. Thus TEAM is able to allow the user to volunteer a sentence from which, with the help of some hard-wired auxiliary questions, it infers the syntactic and semantic characteristics of the way a verb and its arguments map into the database.

Getting rid of the user-interaction completely has proven to be feasible in applications where the lexicon mainly is of "part-of-speech character", i.e., contains only rudimentary information. Such a system is  $L^2$  [Rayner *et al* 1988] which uses only a formal grammar (later extended with a function-word lexicon and a morphological component [Hörmander 1988]) to learn a lexicon from a set of example sentences. The key idea in  $L^2$  was that the grammar, together with the known words, would place restrictions on the unknown words sufficient to infer lexical entries for them.

The  $L^2$ -project proved the feasibility of the approach, but not only was the system able to induce a rather small set of features, it was also discouragingly slow when parsing sentences containing several unknown words. Several strategies for remedying these problems have been suggested: e.g., using explanation-based learning to extract templates describing grammatically correct phrases [Asker *et al* 1992], or using the information from an Artificial Neural Network to resolve morphological ambiguities [Eineborg 1992].

The situation described in the present paper is somewhat different from the ones above mainly due to the requirements of the natural language processing system for which the lexicon is to be acquired. This system, the Swedish Core Language Engine (or S-CLE for short) [Gambäck & Rayner 1992] is a general purpose natural language processing system for Swedish developed by the Swedish Institute of Computer Science from its English counter-part, the SRI Core Language Engine [Alshawi (ed.) 1992]. The key idea behind the system is indicated by the word “core”: the S-CLE is intended to be used as a building block in a broad range of applications, e.g., data-base query systems, text-to-speech/speech-to-text systems, and so on. The two copies of the CLE have been used together to form the BCI machine translation system [Alshawi *et al* 1991a].

The wide range of possible applications have put severe restrictions on the type of lexicon that can be used. An overview of the S-CLE and its lexicon follows in the next section. Section 3 then goes on to discuss the particular lexicon acquisition task in more detail, together with a description of the solutions adopted in the VEX system [Carter 1989].

The main part of this paper is concerned with the extensions and generalizations needed to apply the VEX strategy to Swedish — a Germanic language with an inflectional morphology which obviously is very much more complex than the one of English. Eliciting morphological information is the subject of Section 4, while Sections 5 and 6 discuss the problems of acquiring information on syntactic and semantic usage, respectively. Section 7 describes how the newly created lexical entries are incorporated into the S-CLE and its lexicon. Finally, Section 8 sums up the arguments.

## 2 The S-CLE and its lexicon

As noted above, the Swedish Core Language Engine (S-CLE) is a general purpose natural language processing system for Swedish developed from its English counter-part, the SRI Core Language Engine. The object of both versions of the CLE is the same: to map certain natural language expressions into appropriate predicates in logical form (or QLF, “Quasi-Logical Form” [Alshawi & van Eijck 1989]). The systems are based completely on unification. The Swedish grammar is fairly large and covers most of the common constructions in Swedish. There is a good treatment of inflectional morphology, covering all main inflectional classes of nouns, verbs and adjectives.

The function-word lexicon contains about 550 words, including most pronouns, conjunctions, prepositions, determiners, particles and “special” verbs. In addition, there is a “core” content-word lexicon (with common nouns, verbs and adjectives) and domain specific lexica. This part of the system is still under development and all these content-word lexica together have about 1300 entries.

The lexical entries contain information about inflectional morphology, syntactic and semantic subcategorization, and sortal (selectional) restrictions. Information about the linguistic properties of an entry is represented by complex categories that include a principal category symbol and specifications of constraints on the values of syntactic/semantic features. Such categories also appear in the S-CLE’s grammar and matching and merging of the information encoded in them is carried out by unification during parsing. Two categories can be unified if the constraints on their feature values are compatible.

In the actual “core” and domain lexica, this information is kept implicit and represented as pointers to entries in a “paradigm” lexicon with a number of words representing basic word usages and inflections. For these words only the complete set of feature values is explicitly specified.

Sortal restrictions are defined for logical form predicates (i.e., semantic lexicon entries). After possible semantic interpretations are constructed, the S-CLE applies these restrictions with reference to a user-definable hierarchy of sortal classes, to reject any interpretations in which the sort expected by some argument of a predicate is inconsistent with that of the object filling that argument.

An example of a lexicon entry is the following one for the transitive particle verb *smutsa ned* (“get dirty”, lit. “dirty down”):<sup>2</sup>

```

1  subst_record(lex,v_subj_obj_particle,
2    [v_subj_obj_particle,para_particle,
3    mf(morphc),mf(deponency)],
4    [smutsa,ned,
5    v1,n]).
6  subst_record(sense,v_subj_obj_particle,
7    [v_subj_obj_particle1,
8    v_subj_obj_particle,para_particle],
9    [smutsa_ned_3p,
10   smutsa,ned]).
11 sor(smutsa_ned_3p,
12   [[physev,nonprop,located],[living],[ ]],
13   =>[proposition]).

```

The three different parts of the entry are for syntax (*lex*), semantics (*sense*), and sortal restrictions (*sor*), respectively. The first two are represented as “substitution records”, indicating the implicit nature of the lexicon. The lists are to be read as pair-wise declarations, i.e., the items in the lists on lines 2–3 and 7–8 correspond to the ones on lines 4–5 and 9–10, respectively.

The *lex* entry states that this is a verb following the paradigm *v\_subj\_obj\_particle*, the verb being *smutsa* and the particle *ned*. Lines 3 and 5 contain the morphological information (*mf* stands for “morphological feature”), and state that the morphological category is *v1* (a verb of the first conjugation) and that this verb not is deponent. The *sense* entry declares the name of this particular usage of the verb to be *smutsa\_ned\_3p* (where *3p* indicate that it is a three-place predicate).

The sortal (selectional) restriction entry show that the logical form of *smutsa\_ned\_3p* has three slots, one for the predicate itself, one for the subject, and one for the object, each with its own list of restrictions. The predicate is a physical event that is nonpropositional and located; the subject is a living being, while the object can be anything (the list of restrictions is empty). The *=>* is to be read as an implication, showing the resulting sort of the phrase: a proposition.

---

<sup>2</sup>In the current version of the CLE, the lexicon format has been slightly simplified.

### 3 The VEX system

In the English CLE new lexicon entries can be added by the users with a tool developed for the purpose. This lexicon acquisition tool, the Vocabulary EXpander (VEX), is fully described in [Carter 1989]. In that paper, the interested reader can also find a description of the assumptions behind, as well as a thorough discussion of, and motivation for, the general VEX strategy. This document is only concerned with the system in general and the issues relevant for the Swedish version in particular.

VEX allows for the creation of lexicon entries by users with knowledge both of English and of a specific application domain, but not of linguistic theory or of the way lexical entries are represented in the CLE. It asks the user for information on the grammaticality of example sentences, and for selectional restrictions on arguments of predicates, and writes to disc a set of instructions that can immediately be used by the CLE to create appropriate lexical entries automatically in main memory.

This strategy was chosen since it was deemed too risky to allow the user to volunteer sentences to VEX, given the wide syntactic coverage of the CLE and the fact that it is intended to interface to a multitude of back-end systems in a broad range of possible applications. Instead, VEX itself presents example sentences to the user and asks whether or not they are acceptable. In addition, the logical forms produced are of a fairly neutral, conservative nature, and correspond one-to-one to the individual surface subcategorization(s) that are identified; i.e., different senses of a word will be mapped onto different predicates, leaving it to the back-end to make whatever it needs to of the relationship between them.

The lexical acquisition procedure in the VEX system is based on the different types of information that is to be extracted. The only “real” linguistic information the user needs to provide is the general category (noun, verb, or adjective) of the new lexical entry. All other information needed to create lexicon entries is obtained from the user’s answers to questions/examples presented to her by the system. The procedure mainly goes along the following lines:

1. The user provides the new lexical entry and its general category.
2. The system suggests regular (inflectional) forms of the word, which the user either accepts by default, or rejects by entering the correct (irregular) form.

3. The system elicits information about the applicability of syntactic sub-categorization paradigms by presenting the user with example sentences, for which the user provide grammaticality judgements.
4. Once a set of paradigms has been established, VEX asks for a name for the predicate corresponding to each one.
5. The user chooses the sortal (selectional) restrictions on the final logical form predicate and its arguments.
6. VEX writes out to disc a set of “implicit” lexical entries.

The produced implicit lexical entries are instructions interpreted by CLE code that makes substitutions, for words and predicate names, in entries for the paradigms that VEX knows about. The results of these substitutions are explicit, feature-based entries, which are then compiled directly into the format used by the parser itself.

In parallel with the development of the S-CLE, a Swedish version of the VEX system was designed. Of course, most of the processing in this system, the S-VEX, equals the one in the corresponding part of the original English VEX. The following sections will describe in more detail the parts of the system that differ the most, i.e., the morphological acquisition component, as well as the retrieval of – and motivation for – some of the Swedish-specific syntactic paradigms (the points 2 and 3 above).

First, however, we need to define the notion of (sub-categorization) “paradigm”. The definition adopted here is the one from [Carter 1989], namely that

**Definition 1**

*A paradigm is any minimal non-empty intersection of entries, or, equivalently, any maximal set of categories with the same distribution among entries. That is, every category in a paradigm will occur in exactly the same set of entries in the ideal lexicon as every other category (if any) in that paradigm; and every entry will be a disjoint union of paradigms.*

Here, it is assumed that a lexicon can be described in terms of (a small set of) such paradigms, relying on the fact that the open-class words exhibit at least approximate regularities.<sup>3</sup> This contrasts with the view of e.g. [Gross 1975], that *every* word is in some way idiosyncratic.

<sup>3</sup>The system does not attempt to cope with closed-category words. They have to be entered into a specific function-word lexicon by a skilled linguist.

## 4 Morphology

Moving on to describing the parts of the VEX that needed to be reworked the most when the scheme was generalized to Swedish: After the user has entered a new lexical entry and its general category, the S-VEX system would go to work.

First the morphological inflections are determined: the system knows of 14 different inflectional classes of Swedish nouns, 3 classes of adjectives, and 24 classes of verbs. The user is presented with a menu containing the possible regular forms of a particular inflection. Depending on the major category, a number of such inflected forms must be retrieved: 6 for adjectives, 4 for nouns, and 7 for verbs. By using the information obtained from previous questions, and choosing the order of the questions carefully, the alternatives are narrowed down as fast as possible. Thus the aim is to make most “menus” contain only one alternative which the user can accept as default, or reject by entering the correct (irregular) form.

The key items in determining the regular forms are so called “inflectional affix descriptions”, which contain information about the morphological rules that can be applied to obtain a particular inflected form. Such a description is the following for the supine form of verbs:

```

1 inflectional_affix_description(verb,
2   supine,Supinum,
3   suffix,root,
4   [Humpen har,*form*],
5   [[v_v_affix],
6    [v_v_supine_stem,v_v_affix]],
7   [v:[vform=supine]],
8   yes).
```

Supinum and supine are the Swedish and internal names for the supine form. Line 3 indicates that the supine is formed by adding a suffix to the root form. This is followed by an example that is to be presented to the user, where the inflected form of the entry to be created is going to be substituted for \*form\*. This form is created using some of the S-CLE’s 25 rules for regular morphology and 70 rules for affixes. The Prolog list of lists on lines 5-6 show the morphological rules that can be applied; either the ordinary verb = verb + affix rule (`v_v_affix`) on its own, or after having changed the stem to a special one for the supine. When applying these rules, the feature values specified in the list on line

7 should be set; i.e., in this case the single item specifying that the main category verb (*v*) should have the verb form feature set to the supine. The *yes* on line 8 simply indicates that irregular entries are allowed for this inflection.

In a first version, the S-VEX was kept completely as a stand-alone system not relying on any of the S-CLE code. Since the morphology was defined explicitly in that version, it allowed for some-what more efficient code. However, the disadvantages of such duplication of information made us decide on using the morphological rules of the S-CLE for S-VEX also.

One of the morphological (*morph*) rules used in the S-CLE is the previously mentioned main rule for verbs, which (some-what simplified) looks like this:

```
morph(v_v_affix,
[v:[deponent=Dep, gaps=Gaps, subcat=Sub, agr=Agr,
  vform=(\imper\supine_stem), vform=OutForm,
  synmorphc=Morph, nullmorphv=NotOutForm, ...],
v:[deponent=Dep, gaps=Gaps, subcat=Sub, agr=Agr,
  vform=(imper\supine_stem), vform=InForm,
  synmorphc=Morph, nullmorphv=NotOutForm, ...],
'V_AFFIX':[synmorphc=Morph, notvform=NotOutForm,
  invform=InForm, vform=OutForm, ...]]).
```

This rule specifies that a verb can be formed from adding an affix to a stem under some conditions specified by the different feature values; however, as shown many features do not impose any explicit restrictions on the usage of this particular rule, but are rather “passed on up” by Prolog variables as e.g., agreement (*agr*) and subcategorization (*subcat*).<sup>4</sup> Other feature values must unify explicitly, as for example the syntactic morphological category feature (*synmorphc*) here. Apart from that feature, the most important ones in this rule are *nullmorphv* and *vform*. The first of these indicate inflected forms that can be formed without affixing, i.e., where the root form and the inflected form are the same. The second one cooperates with the affix features *invform* and *notvform* to pose restrictions on the verb form on the original stem (here imperative or “supine stem”) and on the inflected form produced (which here can be anything that is neither imperative nor “supine stem”).

The restrictions imposed are made more explicit in affix rules, e.g.

<sup>4</sup>Many features will also get their “default” values and can thus be left out completely.

```
lex('-r',
  ['V_AFFIX': [lexform='-r',
               invform=impera, vform=(fin/\present),
               notvform=(\ (fin/\present)),
               symmorphc=(1\3\/(4\sub3))]]).
```

Where the root (*invform*) is the imperative, while the form ending with the '-r' is the present. This affix rule can only be used for verbs belonging to either first, third, or (the 3rd subcategory of the) fourth conjugation.

The combinations of stems and affixes that pass through these morphological rules are then fed to “segmentation rules” imposing specific restrictions on the spelling of the inflected forms. About 60 such rules for pre-, in- and suffixing exist and they can be rather complicated. A simple one is the following defining how '-er' may be suffixed to a word depending on the word's root-ending:

```
suffix_rule(['-er'],
  [pair(v3er, v3),
   pair(c4er, ec4)],
  pair(c0er, c0)).
```

Here *v3*, *c0*, and *c4* are just convenient short-hands for different groups of letters (all vowels but 'a', all consonants, and the consonants 'l', 'n', and 'r', respectively). The general format of a suffix rule is

```
suffix_rule({suffix}{exceptions}{default})
```

That is, in the above rule the default is to just add '-er'. However, a second-to-final 'e' could be dropped in words ending with some specific consonants.

The inflected forms surviving the spelling checks have passed the conditions sufficient to be presented to the user; however, the previous answers may make some guesses irrelevant. For example, if we know the verb to be weak, there is no point in suggesting a supine form ending with '-it', since that ending is for strong verbs only. The final guesses are produced after such “filter conditions” have been taken into account. To make as efficient use as possible of the information available, the ordering of the questions for different inflected forms is crucial.

As an example of the morphological acquisition process, suppose the user wishes to define the phrasal verb *smutsa ned* (lit. “dirty down”).

After the user has entered the root form (the imperative), S-VEX would realize that *ned* is a particle, and only request the inflected forms of the verb itself. For each inflection either a “menu” of alternative forms, or the one and only possible regular form (within square brackets) is presented. The user can accept the form suggested (or indicate that the form not is applicable) by typing “Return”, choose a form’s number from the menu, or volunteer an (irregular) form. The user’s choice for the seven non-root forms of the verb (“Infinitiv”, etc.) are in bold face.

Ge böjningsformerna av verbet "smutsa"  
 ("Return" om en form inte förekommer):  
 Infinitiv, t.ex.  
 Humpen ska smutsa [smutsa]: **CR**

Preteritum indikativ, t.ex.  
 Humpen smutsade igår  
 1 smutsade  
 2 smutsa  
 Ge ett nummer, den korrekta formen,  
 eller "Return": 1

Supinum, t.ex.  
 Humpen har smutsat  
 1 smutsat  
 2 smutsatt  
 3 smutsait  
 Ge ett nummer, den korrekta formen,  
 eller "Return": 1

Presens indikativ, t.ex.  
 Humpen smutsar nu  
 1 smutsar  
 2 smutsa  
 Ge ett nummer, den korrekta formen,  
 eller "Return": 1

Presens particip, t.ex.  
 En smutsande hump [smutsande]: **CR**

Perfekt particip, t.ex.  
 Ett bojum har blivit smutsat  
 1 smutsat  
 2 smutsatt  
 Ge ett nummer, den korrekta formen,  
 eller "Return": 1

Passiv, t.ex.  
 Vadån smutsas av humpen [smutsas]: **CR**

## 5 Syntactic usage

Having decided on the morphology, the system moves on to try to determine syntactic usage. S-VEX adopts a *copy and edit* strategy in constructing the syntactic part of the lexical entries. It is provided with pointers to entries in a “paradigm” lexicon for a number of representative word usages and declarative knowledge of the range of sentential contexts in which these usages can occur. This knowledge is encoded in 58 sentence patterns (or `s_patterns`). S-VEX elicits grammaticality judgements from the user to determine which paradigm (or set of paradigms) occurs in the same contexts as the word being defined, and then constructs the new entries by making substitutions in these paradigm entries. Each use of a paradigm will give rise to one distinct predicate.

Each `s_pattern` consists of three lists: one with the example to be presented to the user, one with paradigms that it must follow the pattern, and one with paradigms that (optionally) can follow it.

```
s_pattern(
  [humpen,*verbar*,*partprep1*,bojumet],
  [v_subj_obj_particle,v_subj_specpp],
  []).

s_pattern(
  [bojumet,*partprep1*,#nospace#,
   *verbas*,av,humpen],
  [v_subj_obj_particle],
  []).
```

The above example shows that a phrasal verb such as *lita på* (“rely on”) that takes a compulsory prepositional phrase complement (i.e., a verb that adheres to the `v_subj_specpp` paradigm definition) can be the main verb in a sentence of the form “np *verb prep* np” (e.g., “John *relies on* Mary”), but not in one of the form “np *prep verb* by np”<sup>5</sup> (e.g., \* “Mary *relies on* by John”). Verbs belonging to the `v_subj_obj_particle` paradigm (e.g., the previously mentioned *smutsa ned*) can, however, be main verbs even in such sentences.

The special entry `#nospace#` indicates that the verb form in such a sentence should be formed by “gluing together” the particle and the verb (i.e., *nedsmutsa* rather than *ned smutsa*).

---

<sup>5</sup>“av” (in the pattern) is Swedish for “by”.

Entries in the paradigm lexicon are distinguished not only by the type and number of arguments they take, but also by phenomena such as subject raising, “tough-movement”, reflexives, and equi-NP deletion. In total 62 different paradigms are known by the present system; 5 for nouns, 10 for adjectives, and all the others for verbs. Many of these paradigms are essentially the same in the Swedish and English systems, the most significant difference being that the Swedish version has special ones for reflexives. For example the paradigm `v_subj_specpp` above would be complemented with `v_subj_refl_specpp` (“gifta sig med”; “marry”, lit. “marry oneself to”).

S-VEX’s task is to discover the behavior of the new word or phrase by presenting as few example sentences to the user as possible, and then to find the minimal subset of the paradigms that between them account for that behavior. The sets of paradigms and sentences are progressively reduced by removal of paradigms for a different part of speech or number of words from those of the new phrase, and of sentence patterns which either do not correspond to any surviving paradigms, or whose grammaticality can be deduced from that of other patterns in the subset.

The remaining sentence patterns, with forms of the item being defined substituted in, are presented to the user, who states which of them are grammatical. S-VEX then tries to find a minimal set of paradigms which, together, occur in all and only the contexts the user has marked as grammatical. If there is exactly one such set, all is well; if there are none, the user is asked to reconsider her decisions, as described earlier; and if there are several, it normally means that S-VEX has been unable to distinguish two senses of the word being defined. When this occurs, the user is asked to define the two senses separately.

Although grammaticality judgments are sometimes variable and indeterminate, they are much less so than judgments of semantic acceptability, which do not play any part in S-VEX’s main decision-making process. In order to remind the user to judge grammaticality rather than semantic well-formedness, S-VEX presents example sentences containing “nonsense nouns” such as “humpen”, “vadån” and “bojumet”.<sup>6</sup>

Continuing the example from the previous section: after morphological information has been supplied, S-VEX presents the user with a list of sentences (divided into two by a measure of how complicated they are) and invites her to specify which ones are grammatical in the domain in

---

<sup>6</sup>Translated to English, this would (obviously?) be “thingummy”, “whatsit” and “bojum”.

question:

Här är några tänkbara meningar.

Enklare meningar:

- 1 Humpen smutsar ned.
- 2 Humpen smutsar ned vadån.
- 3 Humpen smutsar sig ned vadån.
- 4 Ned vadån smutsar sig humpen.
- 5 Det smutsar ned en hump.
- 6 Vadån nedsmutsas av humpen.

Mer komplicerade meningar:

- 7 Humpen smutsar ned att finnas.
- 8 Humpen smutsar sig ned att finnas.

Ge numren på de meningar som är grammatiska  
i din domän för det ord/fras som ska  
definieras, eller "?" för hjälp: 2 6

S-VEX now has enough information to decide that *smutsa ned* behaves syntactically as a transitive particle verb. It goes on to try to acquire selectional restrictions on the resulting predicate; the topic of the next section.

## 6 Eliciting Semantic Information

Once a set of paradigms has been established, S-VEX asks for a name for the predicate corresponding to each one, and then for sortal restrictions for the predicate and its arguments. These restrictions may be given directly as a list (interpreted conjunctively) of atoms occurring in the sort hierarchy currently in force, or indirectly as a pointer to sortal restrictions on another predicate or one of its arguments. If an explicit list is provided, its members are checked for existence in the sortal restriction hierarchy and for mutual consistency (e.g. the list "male female" would normally be rejected), but no check is made for the existence of other predicates referred to, since these may not yet have been defined or incorporated into the system.

S-VEX allows the user to specify any number of alternative sets of restrictions on a predicate. However, the use of more than one set is discouraged, because if the alternative restrictions are assigned to distinct predicates then the S-CLE will be able to provide the back-end system with more information than would otherwise be possible.

When a noun is being defined, S-VEX not only elicits sortal restrictions on the basic predicate, but also asks whether a relational sense should also be defined. If the user answers that it should, sortal restrictions are elicited for the new argument (e.g. for the kinds of things that can *have* mothers or weights), and a set of possible relational constructions are presented, for the user to decide which ones are grammatical and can mean the same thing. In this way, dimensional nouns can be distinguished from non-dimensional ones; for example, “John’s weight is 60 kilos” can be paraphrased as “John has a weight of 60 kilos”, but “John’s mother is Mary” cannot be paraphrased as “John has a mother of Mary”.

## 7 Incorporating New Entries; QVEX

When sortal restrictions have been acquired, S-VEX writes out a set of “implicit” lexical entries, creating a new file whose name reflects the word or phrase being defined, or, at the user’s option, extending an existing file of user entries. The first type of file can be used as input to QVEX, a “quick” version of S-VEX that defines a set of words essentially as copies of an original. This can be useful when, for example, defining a list of proper names whose properties as far as the S-CLE is concerned are identical.

Implicit lexical entries are instructions interpreted by S-CLE commands that make substitutions, for words and predicate names only, in entries for the paradigms that S-VEX knows about. The results of these substitutions are explicit, feature-based entries, which the S-CLE can compile directly into the internal format used by the parser and semantic interpreter. Both substitution and compilation happen automatically and are hidden from the user; thus as soon as a word is defined with S-VEX, it can be used in an input sentence.

Neither S-VEX nor the substitution commands make any check for existing entries for words and phrases receiving new definitions. If the new entries are intended as replacements for any existing ones, it is up to the user to remove the outdated ones by hand.

## 8 Conclusions

The application-independence of the S-CLE leads to a style of lexical acquisition different from that used in dedicated natural-language front ends. Fully automatic lexical acquisition for a lexicon of such a complexity as the one in the S-CLE still is far from feasible. Instead the paper has described an interactive technique based on the user's judgments of sentence grammaticality in a specific domain.

The lexicon acquisition strategy in the S-VEX system builds on "morphological paradigms", i.e., definitions of how regular inflected forms can be construed, and on a limited number of syntactic/semantic (subcategorization) "paradigms". A subset of these paradigms are selected for the construction of new lexical entries. S-VEX's concentration on paradigms allows a wide range of subcategorization types to be recognized and dealt with.

The project has shown that the strategy adopted in the VEX system is widely applicable, both for different applications and domains, and for different languages. The frame-work has been generalized to such an extent that it should be a relatively easy task to create a system based on the same principles for other languages.

## 9 Acknowledgements

The work reported here was funded by the Swedish Institute of Computer Science. I would like to thank Manny Rayner, Anna Lövgren and Christer Samuelsson for many helpful discussions and (unwilling) debugging. Several very special "Thanks!" go to David Carter, SRI Cambridge for providing both expertise, friendship and coding on the generalization of the frame-work. Dave, I'm sorry that I repaid you by bringing you to your first punk-rock concert!

## References

- [Alshawi (ed.) 1992]  
Alshawi, H. (ed.) (1992). *The Core Language Engine*, Cambridge, Massachusetts: The MIT Press.
- [Alshawi & van Eijck 1989]  
Alshawi, H. and J. van Eijck (1989). "Logical Forms in the Core Language Engine", *the 27th Annual Meeting of the Association for Computational Linguistics*, Vancouver, British Columbia, pp. 25–32.
- [Alshawi et al 1989]  
Alshawi, H., D.M. Carter, J. van Eijck, R.C. Moore, D.B. Moran, F.N.C. Pereira, S.G. Pulman and A.G. Smith (1989). *Research Programme in Natural Language Processing: Final Report*, SRI Research Report, Cambridge, England.
- [Alshawi et al 1991a]  
Alshawi, H., D. Carter, B. Gambäck and M. Rayner (1991). "Translation by Quasi Logical Form Transfer", *the 29th Annual Meeting of the Association for Computational Linguistics*, University of California, Berkeley, California, pp. 161–168.
- [Alshawi et al 1991b]  
Alshawi, H., C. Brown, D. Carter, B. Gambäck, S.G. Pulman and M. Rayner (1991). *Bilingual Conversation Interpreter: A Prototype Message Translator. Final Report*, joint SICS (R91011) and SRI (CCSRC-018) Research Report, Stockholm, Sweden and Cambridge, England.
- [Asker et al 1992]  
Asker, L., B. Gambäck and C. Samuelsson (1992). "EBL<sup>2</sup>: An Approach to Automatic Lexical Acquisition", *the 14th International Conference on Computational Linguistics*, Nantes, France, pp. 1172–1176.
- [Carter 1989]  
Carter, D. (1989). "Lexical Acquisition in the Core Language Engine", *the 4th Conference of the European Chapter of the Association for Computational Linguistics*, Manchester, England, pp. 137–144; also available as Report CCSRC-012, SRI International, Cambridge, England, and in [Alshawi (ed.) 1992].

[Calzolari & Bindi 1990]

Calzolari, N. and R. Bindi (1990). "Acquisition of Lexical Information from a Large Textual Italian Corpus", *the 13th International Conference on Computational Linguistics*, Helsinki, Finland, Vol. 3, pp. 54–59.

[Eineborg 1992]

Eineborg, M. (1992). "Learning Swedish Morphology with a Neural Network", SICS Technical Report — T92010, Stockholm, Sweden.

[Gambäck & Rayner 1992]

Gambäck B. and M. Rayner (1992). "The Swedish Core Language Engine", *the 3rd Nordic Conference of Text Comprehension in Man and Machine*, Linköping, Sweden. Also available as SICS Research Report – R92013, Stockholm, Sweden.

[Gross 1975]

Gross, M. (1975). *Méthodes en Syntaxe*, Hermann, Paris.

[Grosz *et al* 1987]

Grosz, B.J., D.E. Appelt, P. Martin, and F.C.N. Pereira (1987). "TEAM: An Experiment in the Design of Transportable Natural-Language Interfaces", *Artificial Intelligence*, **32**:173–243.

[Hörmander 1988]

Hörmander, S. (1988). "The Problems of Learning a Lexicon with a Formal Grammar", SICS Research Report — R88019, Stockholm, Sweden.

[Rayner *et al* 1988]

Rayner, M., Å. Hugosson and G. Hagert (1988). "Using a Logic Grammar to Learn a Lexicon", *the 12th International Conference on Computational Linguistics*, Budapest, Hungary, pp. 524–529. Also available as SICS Research Report – R88001, Stockholm, Sweden.