Formalisms for encoding Polish multiword expressions

Radosław Moszczyński

December 2006

Abstract

Multiword expressions are problematic at many stages of natural language processing due to their idiosyncratic linguistic properties, and the ability to undergo syntactic transformations. In inflectional languages, such as Polish, processing them is especially difficult since the repertoire of variations they can undergo is greater than in the case of fixed word order languages.

The report presents several linguistic formalisms and evaluates them on the basis of their suitability for encoding Polish multiword expressions. It includes a detailed description of their most prominent features, and a selection of examples which show exactly how effective the formalisms are when used for Polish.

Keywords: idioms, multiword expressions, formal linguistics, natural language processing

1 Introduction

The primary aim of this report is to evaluate a number of linguistic formalisms from the point of view of encoding Polish multiword expressions.

In the report the term "multiword expression" will be used to refer to any kind of linguistic object that consists of several orthographic words, but should be treated as a whole for the purposes of natural language processing, due to idiosyncratic syntactic and/or semantic properties. In this sense multiword expressions can also be (and will be in this report) referred to as "idioms". A summary of several theoretical linguistic descriptions of multiword expressions can be found in [10].

Multiword expressions require a formal representation for several reasons. Some of them follow morphological or syntactic patterns that are incorrect from the point of view of grammar rules, and thus can cause failures of automatic analyzers. On the other hand, some multiword expressions are correct syntactically, but their meaning is not compositional, and therefore they should be recognized as wholes by such applications as electronic dictionaries.

The main difficulty with providing a formal description of idioms is the fact that many of them are not fixed phrases. They can undergo varying amounts of syntactic transformations, constituent substitutions, word order variations, as well as adverbial and adjectival modifications. The most complex idioms exhibit flexibility that is equal to that of regular sentences of a language, which implies that the formalism used to describe them should be powerful enough to be able to cover a large part of that language's grammar.

The structure of the report is the following. Section 2 provides more detailed information about multiword expressions and postulates the requirements that a successful formalism for encoding Polish idioms should meet. Section 3 provides a description of several formalisms which were developed for encoding multiword expressions in various languages — mostly English — and evaluates them from the point of view of the requirements presented earlier. Section 4 continues the presentation with a description of two formalisms developed for slightly different purposes, but which might be used to encode some types of idioms in a fairly successful manner. Finally, Section 5 provides conclusions and suggestions for processing multiword expressions in Polish.

2 Formalism requirements

2.1 Types of multiword expressions

Encoding multiword expressions for natural language processing has to be preceded by two decisions. Firstly, there is the question of how large a set of phrases should be encoded. Secondly, when the set has been defined, the remaining question is which constituents of the units should be treated as their invariable and necessary parts.

As far as the first aspect of the issue is concerned, the decisions should be based on the application. Polish multiword expressions can be roughly divided into two categories, both of which can pose problems for different types of automatic language processing tools.

The first category consists of "syntactic idioms", i.e., phrases that contain "bound words", which are not used independently or as parts of other multiword expressions. This category also covers phrases containing words that follow abnormal inflectional paradigms (i.e., different from the paradigms they follow when used outside a given multiword expression). Lastly, the category includes phrases whose syntactic structure is non-standard from the point of view of the general rules of the grammar. Both types of phrases are problematic mainly for morphological and syntactic analyzers, and often result in wrong lemmatization or syntactic analyses. Some examples of such phrases include:

> *na oklep, po omacku, na odczepnego, no przecież, nie ma to tamto,*
> *nie rób z tata wariata*[1]

The second category are idioms that are unexceptional from the point of view of grammar, but are idiosyncratic semantically. This means that their meaning is either non-compositional or only partially compositional [11], and that they should be considered as wholes during the process of establishing the meaning of a sentence. Such phrases should be recognized and treated in a

[1]These highly idiomatic phrases can be roughly translated as follows: *bareback*, *in darkness*, an adverbial used to describe things done to *make somebody leave one alone*, *but of course!*, *anyway*, *don't make a fool out of me.*

special way in such applications as context-sensitive electronic dictionaries (cf. [1]) and computer-aided translation tools. Encoding such units is also beneficial for the purposes of semantic analysis, as it allows to cut down on the number of ambiguous interpretations [9]. In Polish, such phrases include:

(1)	*flaki*	*z*	*olejem*
	tripe-Nom	with	oil-Inst

‘something boring’

(2)	*kwity*	*na*	*kogoś*
	papers-Nom	on	somebody-Acc

‘materials used to compromise somebody’s reputation’

(3)	*strugać*	*wariata*
	to sculpt	loony-Acc

‘to pretend to be crazy’

The first category of idioms is relatively straightforward to encode, as it permits little word order variation and almost no modifications. The second category is different in this respect. Its members usually constitute regular sentences, with a subject, possibly objects, and all the regular word order variations, transformations and modifications that are permitted by the language’s grammar. Encoding this type of idioms requires formalising a large part of the grammar of a given language. Due to this fact, it seems to be more sensible to try to recognize such idioms not within raw input, but rather the output produced by a dedicated syntactic analyzer. However, since most formalisms created for the purposes of multiword expressions try to encode all the phenomena related to such units on their own (which is more justified in the case of fixed word order languages like English), this kind of approach will also be followed in the examples presented in the Appendix.

The other plane of the scope problem is concerned with what components truly constitute a multiword expression and should be included in its description, and which are peripheral and/or very variable and can be left out.

This problem is basically limited to multiword expressions that belong to the second category, and can take an unlimited number of possible phrases to fill their subject and/or object slots. This can be seen in the following expression:[2]

(4)	*komuś*	*przyszło*	*do*	*głowy*	*coś*
	somebody-Dat	came	to	head-Gen	something-Nom

‘something came to somebody’s mind’

The most convenient way to encode this kind of idiom would be to assume that *przyszło do głowy* is its canonical form and that all the other constituents, which are the subject and the indirect object, need not be included in the description. However, the subject of the idiom cannot be omitted in its formal description, which is normally done in the descriptions of English idioms, because in Polish, which is a free word order language, the subject can be moved into a position that is between the “core” constituents of the expression. It is illustrated in the following example:

[2]Examples of multiword expressions typeset in italics are their informal representations. They do not aim at providing an exhaustive list of all the possible variants, but rather the most naturally sounding form.

(5)

Wczoraj	*mi*	*przyszedł*	*pewien*	*pomysł*	*do*	*głowy.*
yesterday	I-Dat	came	certain-Nom	idea-Nom	to	head-Gen

‘Yesterday an idea came to my mind.’

Also the indirect object needs to be encoded in the formal representation, for exactly the same reason as the subject. The following example shows how the indirect object moves in between the “core” words:

(6)

Wczoraj	*przyszedł*	*mi*	*do*	*głowy*	*pomysł.*
yesterday	came	I-Dat	to	head-Gen	idea-Nom

‘Yesterday an idea came to my mind.’

This makes all the attempts to formally encode such complex idioms much more difficult, because the subject and the object can be phrases of arbitrary length and complexity.

For this reason, encoding “semantic” multiword lexemes at the level of immediate constituents, which is the way pursued by most formalisms created for English, does not appear to be optimal in the case of Polish.[3] It would be more reasonable to describe slots in the syntactic structure with the names of the syntactic functions their realizations perform (subject, object), and let specialized, external grammar implementations decide what exactly can be a subject, an object, etc. In this way, the formalism used for multiword expressions would not have to be able to account for the syntactic structure of the possible slot realizations, such as noun phrases or non-finite clauses.

Immediate constituents cannot be abandoned completely, as they are the only way in which it is possible to encode ungrammatical idioms that do not exhibit regular syntactic relations. It might be possible to assign a valid syntactic structure to such idioms as *no nie*, but the structure would always be arbitrary, and it is not clear how such bending of the data to the requirements of the formalism could be justified from the linguistic point of view.

2.2 Variations and modifications

The first phenomenon that a formalism needs to capture is the fact that many multiword expressions contain constituents that are either optional or can be chosen from a limited or an unlimited set of lexemes. As will be seen in Section 3, most formalisms provide distinct operators to handle optional elements, alternative elements, and elements that can be chosen freely from a whole class of words.

It is much easier to limit the realizations of such “empty slots” in a precise way when a negation operator is available. If a constituent of a multiword expression can assume a wide range of realizations of a specific feature (e.g., case) except for one, it is much easier to negate the incorrect value instead of listing all the possible ones.

Another property of multiword expressions that a formalism needs to cover is their ability to undergo syntactic transformations (such as passivization and nominalization) and other word order variations. Possible operations of this type need to be defined for all idioms for two reasons. First of all, a program

[3]However, this way of encoding will be used in the examples related to formalisms that follow such conventions.

for recognizing idioms needs to be able to do so also in the cases in which the idiom has assumed a form that is different from its canonical one. Related to this is the fact that often idioms can undergo only a limited set of operations, beyond which they lose their idiomatic meaning and become plain combinations of words. This is illustrated by the following examples:

(7)

Jan	*wyłożył*	*kawę*	*na*	*ławę.*
John	put out	coffee-Acc	on	table-Acc

'John stated the matters clearly.'

(8)

Kawa	*została*	*wyłożona*	*na*	*ławę.*
coffee-Nom	Auxiliary	put out	on	table-Acc

'The coffee was put out on the table.'

In the examples above, (7) is idiomatic, whereas the passivized version in (8) has only a literal interpretation. Knowing exactly which transformations are possible is necessary to be able to tell the difference between phrases that should be interpreted idiomatically and literally.

Undergoing syntactic transformations is mostly a property of the more complex semantic idioms, which to some extent behave like regular sentences, but even some ungrammatical syntactic idioms have a variable word order (e.g., *to nic* and *nic to*). However, in the case of the complex idioms, the transformations can be defined in terms of syntactic structure changes and, in some cases, their semantic significance, i.e., the way in which they influence meaning.

The word order variations in syntactic idioms which lack a proper grammatical structure do not influence the meaning of the expressions, and usually cannot be assigned a linguistic explanation. Thus, they should be treated as permutations of constituents. A successful formalism should be able to express the fact that an idiom is a permutable set of elements, which might include either single words or immutable lists of words.

The last property a formalism should be able to capture is information about agreement between the constituents of multiword expressions. This concerns only the more complex idioms that have a proper subject-predicate-object structure, and is not strictly necessary to be able to successfully recognize multiword expressions. It is possible to assume a "no wrong input" approach and create a system that will work satisfactorily in some applications. However, such a system will accept ungrammatical input and will not be suitable for generation.

As far as generating correct idioms is concerned, encoding agreement is not enough. Another type of information that is needed is semantics. However, a detailed description of the semantic features of multiword expressions, and the requirements for handling them, is beyond the scope of this report.

2.3 Other issues and potential problems

This section describes several problematic cases related to multiword expressions. The examples come from Polish, but similar phenomena, except for (10), can also be observed in English. Neither of the formalisms described in Section 3 is able to account for them without some further extensions.

Consider the following examples:

(9) `<nominal phrase :  Nom> przy <nominal phrase :  Loc>`

W	*autobusie*	*był*	*człowiek*	*przy*	*człowieku.*
in	bus-Loc	there was	man-Nom	near	man-Loc

‘There was a lot of people on the bus.’

(10) `Rok <->owski`

Zaczął	*się*	*Rok*	*Mozartowski.*
began		year-Nom	Mozart-Adj

‘Mozart Year has begun.’

(11) `<noun> przez małe/duże <the noun’s first phoneme>`

To	*była*	**sz***tuka*	*przez*	*duże*	*“***sz***”*
it	was	art-Nom	with	capital	“a”

‘It was really great art.’

In example (9) there is a need to encode agreement between the constituents of the multiword expressions, but in this case the agreement is not concerned with inflectional features — it is the base forms of the words that should be the same. The same feature is required to handle multiword expressions that are pragmatically marked repetitions of the same word, like in the following example:

– Myślisz, że to obejrzy?
‘– Do you think he’ll watch it?’
– **Obejrzy, obejrzy***, na pewno będzie ciekawy.*
‘– He’ll watch it, he’ll watch it, he will be curious.’

However, even such seemingly easy phenomena display irregularities that need to be handled separately, such as incomplete repetitions:

– A jak mu się nie spodoba?
‘– And what if he doesn’t like it?’
– **Spodoba się, spodoba** <**. . .** >*, niech tylko to zobaczy.*
‘– He’ll like it, he’ll like it, the moment he sees it.’

Example (10) displays a phenomenon in which the slot-filling operation needs to be performed below the level of individual words, the part to be inserted being the stem.

Example (11) contains another kind of “agreement”. Like in the second case, the analysis has to go below the level of words, but here it is even lower, as the elements that have to stay in agreement are not morphemes, but individual graphemes.

3 Formalisms

3.1 IDioms As REgular eXpressions (IDAREX)

3.1.1 Overview

The IDAREX formalism discussed below is an extension of regular expressions created at Xerox laboratories by Lauri Karttunen, Pasi Tapanainen, and

Giuseppe Valetto specifically for linguistic purposes [3]. IDAREX provides a formal way of encoding multiword lexemes with the use of regular expressions. Each IDAREX pattern describes a set of possible realizations of an idiom. The machinery behind the formalism that makes it possible to process such encoded units is based on Xerox's finite-state compilers.

The formalism is independent of the underlying compiler — there were several of them developed by Xerox over the years. The first one was IFSM (Kattunen and Yampol), then came FSC (Tapanainen), and finally XFST (Karttunen). Unfortunately, all of these tools are proprietary and not freely available.

The regular expression formalism employed in IDAREX is presented in detail in [7]. One of its most important features is that in addition to describing regular languages (i.e., sets of strings) it also makes it possible to encode regular relations (which are mappings between two regular languages).

In IDAREX regular relations are used to implement two-level morphology, which is a formalism for encoding morphological alterations developed by Kimmo Koskenniemi in 1980s. Words in this formalism can be represented at either the *lexical level* (which is abstract) or the *surface level* (which represents concrete realizations of words) — hence *two-level* morphology. The exact way in which the two levels are used in IDAREX is explained below.

3.1.2 IDAREX operators and macros

There are four ways for describing individual words in IDAREX expressions (cf. [16] and [3]), all of them are marked by the way colon is used:

1. :surface-form — e.g., `:house`
2. :surface-form morphological-variable: — e.g., `:record Verb:`
3. base-form morphological-variable: — e.g., `graduate Verb:`
4. word-class-variable — e.g., `ADV`

The first two classes of expressions describe words that allow no variation, i.e., they cannot be inflected or modified in any other way. The only difference between them is that the second class directly specifies the word's part of speech, which is required for ambiguous cases (e.g., *record* as a verb and *record* as a noun). The third class describes words that can assume various forms on the surface level. In the example above the verb *graduate* can appear in any number, tense, etc., which is indicated by `Verb:`. The last class is a variable that, in the example above, stands for all adverbs and adverbials. Such variables are useful for encoding idioms that can be modified by a numerous group of words that would be unpractical to list individually.

The set of operators used to describe operations on words and phrases is the following:

- *nothing* — words succeed each other (concatenation)
- `*`, `+`, `|` — basic regular expressions operators
- parentheses `()` — mark an optional part of the idiom
- brackets `[]` — group an expression

- semicolon ; — finishes an expression

Combined with two-level morphology, the operators can be used to create "local grammars" capable of generating the whole set of possible realizations of the idioms they describe. Some examples are provided below and in Section A.1.

```
kick: :the Adj* :bucket;

:an :iron [:hand|:fist] (:in :a :velvet :glove);
```

IDAREX has one more very powerful feature which is the possibility to create macros. Macros are intended for capturing syntactic generalizations in a concise way, without the need to explicitly list all the transformed variants of each multiword expression.

Let's consider the following Polish idiomatic expression:

(12)	NP_Nom	*nabrać*	*wody*	*w*	*usta*
	Noun-NP	take in	water-Gen	in	mouth-Acc

'to keep one's mouth shut'

For the sake of illustration, we will assume that the idiom can have two unique realizations — one with the subject appearing before the verb, and another one with the subject postponed to a position after the verb.[4] Apart from this, the verb can be modified on the left or the right hand side with an adverb (in the case the subject-verb order is inverted, only a left hand side modification is possible). In order to account for all these variants using IDAREX, it would be necessary to write the following complex expression:

```
[NP_Nom Adv* :nabrać Adv* :wody :w :usta |
Adv* :nabrać NP_Nom :wody :w :usta];
```

However, the adverbial modifications and the subject-verb inversion are valid operations for many other idioms, so it is possible to define a macro to avoid listing all the variants for each expression.

The adverb modification macro is the simpler one of the two and can be defined like this:

```
AdvMacro
[Adv* $1 | $1 Adv*]
```

The macro for handling subject-verb inversion could be defined in the following way. Note that macros can be nested in order to achieve even greater compactness.

```
WOMacro
[$1 AdvMacro($2) $3 |
AdvMacro($2) $1 $3]
```

In order to use the macro, the user needs to pass it arguments which are IDAREX expressions, which is shown below:

[4]In reality, the subject can also be realized at the end of the sentence, and in some infrequent cases also between the objects.

```
WOMacro(NP_Nom :nabrać fix(:wody :w :usta))
```

The `fix` macro in the example above is used to group the words which have a fixed order and do not accept modifications. Passing the above expression as an argument for the macro would instantiate the `$n` variables with the individual words in the idiom (variable `$3` would be instantiated with `:wody :w :usta` because it is treated as a whole). Then the macro would generate all the possible realizations of the idiom, as shown in the first example.[5]

3.1.3 Advantages and disadvantages of IDAREX

Despite being used in several large projects, IDAREX does have several drawbacks. It is very surface-processing oriented, which results in overgeneration and the possibility for the rules to accept ungrammatical input.

IDAREX provides separate operators for optional elements, alternative and part-of-speech variables. The latter can be combined with regular expression operators, which makes it possible to mark repetitive elements. Marking words that can inflect is possible with two-level morphology operators, which are limited to the surface and the lexical level (no inflection and unlimited inflection, respectively). Some sources (e.g., [15]) show that the part-of-speech variables can be further constrained, but it is never explained in any detail, and therefore it is not clear exactly how specific the constraints may be.

Transformations can be defined with macros, which are limited to word order variations. There is no possibility to fully encode more complex operations, such as passivization, because the formalism does not provide a means for indicating that a given word changes some of its syntactic features, such as case. Also, IDAREX does not support encoding agreement information.[6] All the tools created to process the rules are now a part of commercial projects, which makes them unavailable.

On the whole, IDAREX appears to be a very efficient tool for encoding the simple "syntactic" idioms (although a permutation operator would be a useful extension). In the case of the more complex multiword expressions, it can only be used for recognition, as it fails to account for more complex operations even in such languages as English.

3.2 PhraseManager

3.2.1 Basic mechanisms and features

The approach to encoding multiword expressions presented in this section has been used in a lexical database system consisting of two major parts — WordManager and PhraseManager, the former being used for mapping word forms into their lexemes, and the latter for mapping multiword expressions into their canonical forms and for recognizing such expressions in text [17].

PhraseManager consists of two formalisms, which are described in [12]. The first one is "Rule Knowledge Specification" and it is intended for linguists. The

[5]Actually, it would overgenerate one adverb, but overgeneration is a trade-off for a simple and effective method of encoding.

[6]Some sources [15] claim that there is a way to encode agreement information in IDAREX, but it has not been implemented in the processing tools. Nevertheless, the way in which agreement should be encoded is not presented.

other one is "Entry Knowledge Specification" which is used by lexicographers. The whole approach leads to a clear division between the job of describing linguistic operations that idioms can undergo, and the job of collecting and describing the idioms proper.

Rule Knowledge Specification offers linguists the possibility to define rules that describe clitics, idiom classes, periphrastic inflection and syntactic transformations. This is to ensure that the system can recognize and generate all the possible realizations of syntactically flexible multiword expressions. The formalism is powerful enough to capture a considerable part of English grammar. The following paragraphs contain a description of its most important features.

Periphrastic inflection rules are used to describe the so-called analytical forms, in which inflectional features are contained not in a single word, but in a multiword expression. This is the case e.g., in English passive forms, which consist of an auxiliary verb and a past participle.

A rule for periphrastic inflection in English passive forms is given below (it reflects such phrases as *was given*, etc.):

```
(Cat V) (Mod Ind) + (Cat V) (Mod Part) (Tense Past) =
(POS 1) (CFORM 2) (PERC 1) (Cat V) (Form Passive)
```

The left hand side contains a description of the kind of strings that the rule should match, which in this case is an indicative verb (`(Cat V) (Mod Ind)`) followed by a past participle (`(Cat V) (Mod Part) (Tense Past)`). The right hand side indicates how the whole periphrastic inflection cluster should be treated in further processing. The interpretation of the symbols is as follows:

- `(POS 1)` — the position of the cluster as a whole, after it has been identified; in this case, it is supposed to be inserted in the position held by the first of its constituents
- `(CFORM 2)` — the citation form should be established on the basis of the second constituent
- `(PERC 1)` — the inflectional features are to be percolated from the first constituent
- `(Cat V)` — the category of the whole cluster should be verbal
- `(Form Passive)` — the cluster indicates passive voice

The rules for transformations make it possible to define syntactic operations which multiword expressions can undergo. Such rules are used subsequently in the lexicographers' formalism. Two simple rules for noun-adjective inversion are presented below:

```
(VP V (NP Det Adj N)) → (VP V (NP Det N Adj))
(NP Adj N) → (NP N Adj)
```

The nested lists represent tree structures of the individual phrases. In each list, the first element is the root of the tree, and the following ones are the daughter nodes.

When idioms are being processed in PhraseManager, all words that contain clitics are separated into two parts. The way it is done is described by the linguists with clitic rules. The example below contains a simple rule for dividing *cannot* and *can't* into *can* and *not*.

```
(Cat V) (Mod Ind) (Tense Pres) {``can''} + (CElement not)
=
(Cat V) (Mod Ind) (Tense Pres), (CElement not)
```

The left hand side of the rule specifies the strings that are to be matched, and the right hand side shows what should be generated as output. The elements of the rule should be interpreted as follows:

- `+` and `,` indicate that the surrounding elements should be interpreted as continuous and separated, respectively
- the word `can` in curly braces indicates the orthographic form of the string which is supposed to be matched
- `(CElement not)` is a reference to an entity `not` which should be defined elsewhere as being either orthographic *not* or *'t*

The final elements that need to be defined by the linguist using Rule Knowledge Specification are the syntactic classes that multiword expressions can belong to. This is because one of the assumptions of PhraseManager is that multiword expressions can be categorized into syntactic classes, on the basis of their internal structure. This assumption reflects a classification that has been created for a subset of English and German in [4].

Each class is assigned with a set of properties that describe its transformational potential and possible adverbial and adjectival modifications. These global patterns can be subsequently augmented by the lexicographer with rules applying to individual units, should they exhibit some idiosyncrasy. A typical rule for a syntactic class is shown below:

```
SYNTAX-TREE
(VP V (NP Art Adj N AdvP))

MODIFICATIONS
V >

TRANSFORMATIONS
Passive, N-Adj-inversion
```

The first part of the rule describes the constituent structure of the class in the form of a nested list. The second part lists possible modifications, which in this case are limited to a right-hand side modification of the verb. The last part of the rule lists all the possible transformations that multiword expressions belonging to this particular class can undergo.

All the classes that the linguist defines are arranged hierarchically into a tree. This is supposed to make it easier to capture small variations in the behavior of individual classes. For example, the rule above defines a class that has a specific constituent structure and can undergo specific transformations. However, it is possible that there exists a class with an identical constituent structure, but with a different list of possible transformations. Therefore, assuming that the constituent structure is referred to as "Verb_NP_with-Adv", the linguist could establish a hierarchy of syntactic structures, whose rough approximation is shown below:

```
ROOT
|
+--Verb_NP_with-Adv
|  |
|  +--with-Passive
|  |
|  +--with-N-Adj-inversion
|  |
|  +--...
|
+--...
```

With such a hierarchy, it is possible to refer to a higher element in order to indicate that the idiom belong to a class which accepts all the possible transformations. The syntax of referring to the defined syntactic classes is described in more detail in the remaining part of this section.

The other formalism employed in PhraseManager, Entry Knowledge Specification (EKS), is intended for use by lexicographers, who are expected to add descriptions of multiword expressions into the system, and indicate their transformational potential by referring to the linguistic rules specified earlier by the linguist.

A typical entry encoded with EKS consists of several parts. An example is shown below:

```
HEADPHRASE
the <coast> <be> clear

RESTRICTIONS
the (Cat Art)
coast (Cat N)
be (Cat V) (Pers 3rd) (Num 'coast)
clear (Cat Adj)

MODIFICATIONS
-

CLASS
(PHClass NP.VP.fixed)
```

The first part of the rule defines the canonical form of the multiword expression and lists all of its constituents. The words that are surrounded by angled brackets can be inflected, by default without any limits.

The following part lists more detailed requirements concerning the constituents. The most important features encoded here are the constraints that limit the range of inflection a word can undergo, and agreement information. In this example the verb *be* should agree in number with the noun *coast*.

The third part is used to define non-standard modifications that go beyond what has been described by the linguist in the rule for the syntactic class. Lastly, the rule indicates exactly which syntactic class the multiword expression belongs to. The notation indicates the path in the syntactic class tree (described above), each step being a node of the tree positioned lower in the hierarchy than the preceding one.

Examples of the way in which the formalism can be used for Polish can be found in Section A.2.

3.2.2 Concluding remarks

Similarly to IDAREX, PhraseManager provides distinct operators for marking alternative/optional elements, and words that can inflect. Unlike in IDAREX, inflection can be precisely limited to a narrow set of possible feature combinations. Information about agreement can be included in the idiom entries and is a part of the Entry Knowledge Specification.

As far as transformations are concerned, the decision to separate the tasks of linguists and lexicographers results in the possibility to describe them in a declarative way (using Entry Knowledge Specification), without the need to get involved in the complex linguistic rules when it is not necessary. Moreover, because of the two-tier architecture, it should be possible to use Entry Knowledge Specification only, and replace the Rule Knowledge Specification backend with other linguistic formalism that for some reasons is more desirable to use.

The disadvantages are limited to basically two issues. One of them is the fact that it is impossible to encode semantic information. However, due to the way the rules and the whole system are built, it seems that adding such a functionality would be possible, should it be necessary. The other disadvantage has to do with complexity. As it is shown in the Appendix, the formalism creates too much overhead in the case of “syntactic idioms”, whose structure is usually hard to define, and whose word order variations are permutations rather than real transformations.

3.3 OntoSem

3.3.1 General description

The OntoSem project is being developed at the University of Maryland, Baltimore County. It is a “text processing environment that takes as input unrestricted text and carries out its tokenization, morphological analysis, syntactic analysis, and semantic analysis to yield TMRs [Text-Meaning Representations]” [8]. TMRs are formal representations of meaning in the form of an interlanguage.

OntoSem’s creators take a different approach to multiword expressions than most other projects. They are not interested in automatic recognition of such units, but rather seek to encode their meaning in order to be able to incorporate them in TMRs, so that the latter faithfully render the meaning of the sentences they represent.

Since the representation of multiword expressions in OntoSem differs only slightly from single words, its authors claim the whole phenomenon can be treated as a regular feature of the language, and not something special, as it is usually done. It is not clear, though, how to treat this claim in the view of the fact that many multiword expressions are odd or malformed from the grammatical point of view, or consist of words that cannot be used independently.

OntoSem’s formalism employs a Lisp-like notation in which attributes and their values are represented as nested lists. An example entry for the phrasal verb *go unpunished* is provided below:

```
go-v25
  anno
    definition "phrasal: go unpunished"
    example "The crime went unpunished."
  syn-struc
    subject        $var1
    v              $var0
    adj            $var2   root unpunished
  sem-struc
    PUNISH
      theme        ^$var1
```

Each multiword expression is represented by a single headword, in this case *go*. All the non-head elements are listed in the syntactic structure. They can be constrained to appear only with particular inflectional features. Their forms can be constrained even further in the semantic part, where it is possible to state, e.g., that the phrasal verb *to go down* means "sink" if and only if its subject is a marine vehicle.

If the described multiword expression has irregular structure, the entry may explicitly state what kind of function it performs in sentences (clause, adverbial, etc.). If the expression exhibits regular structure, the OntoSem parser by default assumes that all the possible transformations might apply to it. In the case of units that loose their idiomatic meaning when subjected to specific operations, it is possible to block transformations altogether by encoding the multiword expression in terms of immediate constituents and not syntactic functions. Unfortunately, there is no way to block selected transformations.

By default the constituents listed in the syntactic structures are assumed to permit all their normal adverbial or adjectival modifications. Also, the constituents are assumed to be able to undergo all the usual morphological processes. In case it is needed, they can also be blocked by specifying the required grammatical features explicitly.

The second level of indentation divides the description into three parts. The first one contains metadata, in this case a definition and a usage example. The syntactic structure in the second part is divided further on into individual constituents. The `var` elements are variables assigned to each of the constituents. These are used to bind the syntactic functions to semantic roles. It is exemplified in the semantic structure part above, which indicates that the subject is also the theme. The specification following `$var2` says that the base form of the word serving the `adj` function is *unpunished*.

Apart from the operators mentioned in the above example, OntoSem makes it possible to specify other grammatical features (e.g., `cat n` or `form infinitive`), and to indicate that a constituent is optional. Also the semantic structure part allows a far more detailed representation, but semantics is not the focus of this report.

Some examples of Polish multiword expressions encoded with OntoSem's formalism can be seen in Section A.3.

3.3.2 Adequacy for encoding Polish idioms

One feature of OntoSem that is worth adopting is the possibility to encode the constituents of multiword expressions in two ways: either at the level of dependencies, or at the level of immediate constituents. The former could be used for "semantic", and the latter for "syntactic" idioms.

Apart from this, the formalism has several drawbacks. Idioms that are syntactically idiosyncratic can only be described as being completely frozen, which is not true for some Polish multiword expressions (cf. Section 2). In the case of such expressions, allowing word order variations requires listing all the possible variants as separate units, which might lead to an undesirable proliferation of similar entities. What is lacking for a proper description of idioms that follow regular grammar patterns is the possibility to block individual transformations, since many multiword expressions can undergo only a limited set of transformations beyond which they lose their idiomatic meaning.

OntoSem is not compatible with the aim of this report, which is finding an appropriate formalism for surface processing of multiword expressions. The formalism features advanced mechanisms to handle semantics, which are largely omitted in the report, but syntactic processing relies on external components. Thus, OntoSem might be useful for processing semantic idioms after the tools for surface processing have been developed — it is not such a tool on its own.

4 Other formalisms

This section presents two more formalisms that to some extent are suitable for encoding Polish idioms, even though their primary purpose is different. The first one has been presented in [2], and could be extended to a fully-fledged formalism even though at the moment it is rather inconsistent. The book it is described in is also noteworthy because of the lexicographic material it contains.

The other formalism is the query language of Poliqarp, a concordancer developed and maintained at IPI PAN. The language is very expressive and is under constant development, and the tools to process it are currently freely available.

4.1 Bogusławski and Danielewiczowa

The formalism presented in [2] was created for phraseological descriptions. The book is a lexicographic study that also covers single-word lexemes, but the bulk of it is devoted to multiword expressions and various syntactic patterns.

The formalism makes it possible to create very detailed descriptions of multiword expressions. A typical example is presented below:

> *ktoś*$_i$ **jest po** $_{\_j}$ **głębszych** ○
> $\mathbf{K}_{\_}$: $_{\_j}$: fraza liczebnikowa
> |**S**| *i* wypił $_{\_j}$ kieliszków mocnego alkoholu
> |**P**| A SYT *pot.*
> ▽ *Franio jest po 2 głębszych, nie może siadać za kierownicą.*

Translated to English, it assumes the following form:

> *somebody*$_i$ **is after** $_{\_j}$ **deeper ones** ○
> $\mathbf{K}_{\_}$: $_{\_j}$: numeral phrase

|**S**| *i* had $\_{j}$ shots of liquor
|**P**| A SYT *pot.*
▽ *Frank had two shots of liquor, he can't drive.*

In the above example, *ktoś*$_i$ serves the purpose of a slot that can be filled with a noun phrase in the nominative case. $\_{j}$ indicates an empty slot that can be filled with something different than a nominal noun phrase, in this case a numeral. The subcategorization constraint is directly stated in the line beginning with **K**, whose purpose is to provide information about empty valence slots. The ∘ symbol means that the described unit can be used independently as a sentence (after fulfilling all the valence requirements). The **S** line contains semantic information, and **P** covers pragmatics. The last line provides a usage example.

Other features of the formalism are illustrated by these (abbreviated) entries:[7]

|<**więcej**> **czadu** ∘ |
. . .

-tak ≺taki≻ _, **że nie wiem** ∘ [tak $_\underline{\underline{\ }}$że$\underline{\underline{\ }}nie\underline{\underline{\ }}$wiem]
. . .

Words enclosed between < and > are optional. The symbol - marks the first part of an alternative, whereas ≺ and ≻ enclose the other one. The vertical bars in the first example indicate that the multiword expression cannot be attached to a bigger structure, neither on the left nor on the right side.

The part in the square brackets contains "prosodic information". Its purpose is mainly to indicate continuity of segments — the double underline symbol means that it is impossible to insert anything between the words which it connects.

There are two other lines that can appear in the descriptions, which are not included in the above examples. One of them is **F** which contains information about inflection, and particularly deficiencies, such as a word being restricted only to plural form within the multiword expression in question. The other one is **TR** which informally describes the unit's place in a sentence's thematic structure.

The formalism presented above has been created with human users in mind, and therefore too little information is stated explicitly, and too much has to be inferred from the general knowledge of the language, which is something that computers cannot do. Therefore, it does not seem to be immediately suitable for computational purposes, although the book remains a great source of lexicographic information.

The information that is missing from multiword expression descriptions is first and foremost word order variations.[8] Besides, the descriptions generally fail to account for possible adverbial and adjectival modifications, and often do not list all the potential lexical variations. Possible syntactic transformations are neither provided, nor is there a place to include them in the descriptions.

[7]The first one roughly means 'pump it up!', and the second one '(something is) so ..., I can't believe it!'.

[8]However, the prosodic information indicates which words cannot be separated, which in some cases also reflects the possible word order variations within a unit.

As far as semantics is concerned, the informal explanations of the units provided in the **S** lines is unsuitable for computational purposes. At times they seem to be inconsistent, as they either refer to the indexed slots included in the multiword expression patterns ("$\_i$ nie $\_j$, wbrew temu, co ktoś mówi" — '$\_i$ is not $\_j$, contrary to what somebody says'), or use plain, vocal descriptions ("możliwości wiadomej osoby w wiadomej dziedzinie nie są nieprzeciętne" — 'the given person's abilities in the given field are not exceptional').

4.2 Poliqarp's query language

The query language of Poliqarp [6, 5] is described in detail on the official website of the IPI PAN corpus of Polish.[9] Poliqarp is the primary tool for searching the corpus, and is under constant development.[10] For the purposes of this report, the presentation of the query language will be limited to orthographic level queries and morphosyntactic tag queries.

Orthographic level queries are invoked using the `orth` directive. The string to be matched can be specified either directly, or indirectly by means of regular expressions. The following examples contain sample queries (each word needs a separate directive):

```
[orth="na"][orth="oklep"]
[orth="na|po"][orth="o.*"]
```

The morphosyntactic level queries can refer to an arbitrary number of inflectional features thanks to the conjunction operator. Constraining the set of matched words is made easier because the negation operator is supported by the language. Complex queries can refer both to the orthographic and the morphosyntactic level (names of parts of speech in the examples below follow the convention used in the IPI PAN corpus):

```
[pos="subst" & case="gen" & number="pl"]
[pos="subst" & case!="voc"]
[pos="adj" & case="gen" & orth="[a|e].*"]
```

Queries that refer to the morphosyntactic level can also be combined with simple regular expressions:

```
[pos="adv"]*[pos=adj]
```

All these abilities could be combined, e.g., to handle one of the problematic cases mentioned in Section 2.3 (*Rok* ⟨...⟩*-wski*):

```
[base="rok"][orth=".*wski.*"]
```

The drawbacks of encoding multiword expressions with Poliqarp's language are twofold. Firstly, there is no way to list possible transformations in a declarative way to be handled by an external mechanism. It is also impossible to indicate that an idiom is a permutable list of words. Therefore, word order variations need to be explicitly indicated by listing all the possible realizations of an expression.

[9]`http://korpus.pl/pl/cheatsheet/index.html`
[10]`http://poliqarp.sf.net`

Moreover, as of yet the query language does not support unification which is necessary to handle agreement. However, an implementation of this mechanism is planned for the future.

On the whole, Poliqarp seems to be well suited for encoding idioms of the syntactic kind, in the case of which the drawbacks mentioned above are not crucial. Some examples of Polish idioms formalized in the language are included in Section A.5.

5 Conclusions

Polish multiword expressions can be divided into two groups. On the one hand there are idioms that are problematic for surface language processing (the "syntactic idioms"), and on the other one are units whose non-compositional meaning requires special attention in lexicographic and translation applications (the "semantic idioms").

Both types include non-fixed phrases that exhibit a certain amount of varying word order and modifications/substitutions, but it is the second type that is much more complex from this point of view. Formalization of the syntactic idioms can be achieved with relatively simple means, whereas semantic idioms need rules that reflect a large part of a language's grammar.

As the presentation of the formalisms in Section 3 has shown, the attempts to account for all the phenomena labeled as multiword expressions with a single tool are not successful, especially in the case of inflectional languages. The formalisms tend to be either too weak for handling the most complex idioms, or too powerful for the simple ones. We believe the rough division of idioms presented above is a good approximation of how multiword expressions should be processed computationally. We postulate processing both groups separately at different stages of the NLP workflow, using different tools if necessary.

Syntactic idioms display abnormal structure or contain constituents that do not appear independently, and thus are problematic to all kinds of automatic analyzers. Therefore, we believe they should be recognized before syntactic analysis, presumably during the preprocessing stage when all kinds of named entities are tagged.

Semantic idioms are in many cases as complex as any non-idiomatic sentences in the given language, and thus their formal representation requires a full-featured grammar. In our opinion, the process of their recognition should take place during syntactic analysis, and only if it is required by the application (e.g., in the case of electronic dictionaries).

A Appendix

Actual usage of the presented formalisms will be based on three multiword expressions of varying degrees of complexity. The first one is the following:

(13) *no* *nie*
oh no
'oh, please...'

A formal syntactic structure of the expressions is hard to establish, because the constituents do not represent any well-defined parts of speech. The expression does not allow for any variations apart from switching the constituents' order (however, it would be hard to call it a transformation bearing in mind that the expression's structure cannot be established — it is rather a random permutation; also, such word order variations do not belong to the set of transformations defined in [14]).

The second example is a conventionalized curse that functions as an independent sentence:[11]

(14)

niech	*NP_Acc*	*szlag*	*trafi*
let	NP_Acc	*	hit_Future

'damn NP_Acc'

The NP_Acc slot can be filled with any pronoun or noun, as well as complex nominal phrases, such as *to wszystko* ('all this'). The word order is very variable, since *Niech NP_Acc trafi szlag!*, *Szlag niech NP_Acc trafi!*, *Niech trafi NP_Acc szlag!*, and *Niech szlag trafi NP_Acc!* are all acceptable. However, like in the previous example, the variations do not belong to the set of transformations listed in [14].

(15)

NP-Nom	*wziąć*	*nogi*	*za*	*pas*
NP-Nom	take	legs-Acc	behind	belt-Acc

'to run away'

In the case of this multiword expression, it is necessary to include the subject in the description. It can be absent altogether, it can appear at the very beginning of the multiword expression without breaking its continuity, but it can also appear after the verb, between the core constituents. The subject can be of arbitrary length and needs to agree in morphosyntactic features (number, gender, and person) with the verb.

The verb can be modified with adverbial phrases, both on the left hand side and the right hand side. However, if the subject is postponed to a position after the verb, all the potential right hand side adverbials need to be attached after the subject, and not directly after the verb.

Although not very frequent, the expression also has a nominalized variant (*wzięcie nóg za pas*), in which the verb becomes a gerund, and the noun *nogi* switches case to genitive.

A.1 IDAREX

(16)
```
[:no :nie | :nie :no];
```

The description produces exactly two strings, which reflects changing the order of the two constituents.

(17)
```
[:niech NP_Acc :szlag :trafi] | \
[:niech :szlag NP_Acc :trafi] | \
[:szlag :niech NP_Acc :trafi] | \
[:niech :trafi NP_Acc :szlag] | \
[:niech :szlag :trafi NP_Acc];
```

[11]The word marked with the asterisk is untranslatable.

The description produces five strings that reflect the possible word order variations.

(18) `NP_Nom wziąć:  :nogi :za :pas;`

The subject-verb inversion and adverbial modifications in the example can be handled by means of the following macros (the mechanism is described in more detail in Section 3.1.2):

```
AdvMacro
[Adv* $1 | $1 Adv*]

WOMacro
[$1 AdvMacro($2) $3 $4 $5 | AdvMacro($2) $1 $3 $4 $5]
```

When the macro is expanded, the following expression results:

```
[NP_Nom Adv* wziąć:  Adv* :nogi :za :pas] | \
[Adv* wziąć:  Adv* NP_Nom :nogi :za :pas];
```

However, such a description suffers from overgeneration, as it can produce a variant in which the right hand side adverbial incorrectly appears before the postponed subject. A possible ad hoc solution is to create another macro for inserting an optional adverb on the right hand side of the subject.

It is not possible to encode subject-verb agreement, which makes the description unsuitable for generation, and creates a risk of accepting wrong input. There is also no way to handle nominalization, as it is not possible to indicate that a given word becomes a gerund or switches case. This can be overcome by relaxing the constraints and encoding the noun *nogi* on the lexical, and not the surface level. However, this might further increase the number of spurious realizations.

A.2 PhraseManager

(19)
```
HEADPHRASE
nie no

RESTRICTIONS
-

MODIFICATIONS
-

CLASS
(PHClass Neg.Qub.permutable)
```

Although the expression is extremely simple and consists of words whose parts of speech are hard to define, in PhraseManager it is necessary to create a separate syntactic class for it. We reserve the top level class (`Neg.Qub`) for other possible expression with the same structure that do not have varying word order, and create a `permutable` subclass for the expression in question:

```
SYNTAX-TREE
(Interjection Neg Qub)

MODIFICATIONS
-

TRANSFORMATIONS
Permutation
```

The syntax tree encodes a hypothetical "interjection" type expression that consists of a negation element and a "qub" element whose part of speech cannot be reliably established. The permutation rule has the following form:

```
(Interjection Neg Qub) → (Interjection Qub Neg)
```

(20)
```
HEADPHRASE
niech noun szlag trafi

RESTRICTIONS
noun (Cat N) (Case acc)

MODIFICATIONS
-

CLASS
(PHClass Qub.NP.N.V.permutable)
```

Since the morphological restrictions section does not provide an alternative operator, it is not clear how to indicate that the `noun` element can also be a pronoun or a complex noun phrase. The syntactic class can be defined as follows:

```
SYNTAX-TREE
(Sentence Qub NP N V)

MODIFICATIONS
-

TRANSFORMATIONS
Permutation
```

The syntax tree defines a sentence-type expression with a flat, four element structure. In this case, the transformation is a group of four rules:

```
Permutation
(Sentence Qub NP N V) → (Sentence Qub N NP V)
(Sentence Qub NP N V) → (Sentence Qub V NP N)
(Sentence Qub NP N V) → (Sentence N Qub NP V)
(Sentence Qub NP N V) → (Sentence Qub N V NP)
```

(21)
```
HEADPHRASE
NP <wziąć> nogi za pas

RESTRICTIONS
NP (Cat Noun) (Case Nom)
wziąć (Cat V) (Num 'NP) (Gen 'NP) (Pers 'NP)
nogi (Cat Noun) (Num Pl) (Case Acc)
za (Cat Prep)
pas (Cat Noun) (Num Sg) (Case Acc)

MODIFICATIONS
-

CLASS
(PHClass NP.VP.NP.PP)
```

Unlike in IDAREX, the description above accounts for subject-verb agreement. The syntactic class the expression belongs to can be defined as follows:

```
SYNTAX-TREE
(Sentence NP (VP V N PP))

MODIFICATIONS
V <,>

TRANSFORMATIONS
Subject-Verb-Inversion
```

There are several problems with defining the transformations for this class of multiword expressions. First of all, as it has been noted above, if the verb is modified on the right and the subject is postponed, the latter needs to be put after the modifier. Since in PhraseManager the transformation rules are independent of the modification rules, it is not possible to establish a condition of the form "if verb has been modified, postpone the subject to a position after the adverb". Therefore, just as in the case of IDAREX, the description can overgenerate the incorrect variant. The inversion rule can have the following form:

```
(Sentence NP (VP V N PP)) → (Sentence (VP V N PP) NP)
```

PhraseManager does not provide the means to create rules that change morphosyntactic features of a given word. Since nominalization requires transforming the verb into gerund and switching the case of the object, it follows that nominalization is impossible to encode by any means other than indicating that the object can inflect and relaxing the constraints on it in the `RESTRICTIONS` section so that other cases are allowed. However, this can lead to overgeneration and accepting wrong input. It would also only work with the assumption that the underlying morphological system treats gerunds as subforms of words that belong to the class `V`.

A.3 OntoSem

The descriptions in this section are missing meaning definitions and semantic data. In OntoSem all the syntactic transformations are handled by an external grammar component, which is why the entries below reflect only the constituent structures of the expressions.

The first two expressions are encoded at the immediate constituent level, which in the case of OntoSem implies that they cannot undergo any word order variations. The only way to account for the variations in their case is to list all the possible variants as separate entries or artificially assign them with a dependency structure, which might be plausible in the second example, but not in the first one.

(22)
```
nie-1
  syn-struc
    neg  $var0  root nie
    qub  $var1  root no
```

(23)
```
trafić-1
  syn-struc
    qub      $var0  root niech
    n        $var1  cat noun  case acc
    n        $var2  root szlag
    v        $var3  root trafić
```

(24)
```
wziąć-1
  syn-struc
    subject     $var0
    v           $var1  root wziąć
    object      $var2  root noga  number pl  case acc
    pp
      prep      $var3  root za
      prep-obj  $var4  root pas  number sg  case acc
```

A.4 Bogusławski and Danielewiczowa

The descriptions are limited to the surface form and the prosodic information. It is interesting to note that in the case of (27) the double underscores, which indicate the lack of potential for modifications also reflect the inability to switch the order of the given constituents. This, however, is not the case in (26).

The descriptions do not account for possible transformations and word order variations. At this moment, the formalism does not include the means to encode such information (other than listing all the variants).

(25) | ▀**no nie** $\prec$**nie no**$\succ$ ○ [no$\underline{\underline{\ }}$nie]

(26) | **niech** *kogoś/coś*$_i$ **szlag trafi** | ○ \
[niech$\underline{\underline{\ }}$kogoś/coś$_i$$\underline{\underline{\ }}$szlag$\underline{\underline{\ }}$trafi]

(27) | *ktoś*$_i$ **wziął nogi za pas** | ○ \
[ktoś$_i$ wziął nogi$\underline{\underline{\ }}za\underline{\underline{\ }}$pas]

A.5 Poliqarp

In its current form, the language of Poliqarp does not allow querying for phrases, so the descriptions below are limited to the word level. It also does not have a means to account for morphosyntactic agreement between constituents. Due to the purposes it has been created for, it also does not currently have any possibilities to encode transformations, and therefore all word order variants are handled with the alternative operator.

(28)
```
[orth="no"][orth="nie"] | \
[orth="nie"][orth="no"]
```

The description produces exactly two strings, which reflects changing the order of the two constituents.

(29)
```
[orth="niech"][pos="subst"][orth="szlag"][orth="trafi"] | \
[orth="niech"][pos="subst"][orth="trafi"][orth="szlag"] | \
[orth="niech"][orth="szlag"][pos="subst"][orth="trafi"] | \
[orth="niech"][orth="trafi"][pos="subst"][orth="szlag"] | \
[orth="niech"][orth="szlag"][orth="trafi"][pos="subst"]
```

A series of alternatives that can produce five possible realizations.[12]

(30)
```
[pos="subst" & case="nom"]? \
[pos="adv|qub"]*[base="wziąć"][pos="adv|qub"]* \
[pos="subst" & case="nom"]? \
[orth="nogi"][orth="za"][orth="pas"]
```

A very complex description that suffers from overgeneration, as it can produce a variant in which the right hand side adverbial incorrectly appears before the postponed subject, as well as a variant with two subjects. There is now way to account for the possible complexity of the subject.[13]

By combining the `base` operator with appropriate morphosyntactic restrictions (e.g., `[base="noga" & case="acc" & number="pl"]` and `[base="noga" & case="gen" & number="pl"]`, it is possible to account for nominalization in the last example. However, there is no way to actually indicate that it is a transformation — it is necessary to use the alternative operator, which further complicates the already complex description.

Acknowledgments
I would like to thank Agata Savary and Janusz S. Bień for reading the initial version of the report and providing valuable suggestions. I would also like to thank Adam Przepiórkowski for his numerous improvement suggestions and his help in preparing the final version of the text.

[12]For the sake of simplicity, the expressions are missing additional constraints on the case of the object.

[13]A phrasal version of Poliqarp is in the making, so in the future it will be possible to substitute the `[pos="subst"]` simplification with a query for a nominal group: `[type="NG"]`, as described in [13].

References

[1] D. Bauer, F. Segond, and A. Zaenen. Locolex: The translation rolls off your tongue. In Proceedings of ACH-ALLC, Santa Barbara, CA, 1995.

[2] A. Bogusławski and M. Danielewiczowa. Verba polona abscondita. Sonda słownikowa III, volume XXIV of Semiosis Lexicographica. Elma Books, Warszawa, 2005.

[3] E. Breidt, F. Segond, and G. Valetto. Formal description of multi-word lexemes with the finite-state formalism idarex. In Proceedings of the 16^{th} Conference on Computational Linguistics, volume 2, pages 1036–1040, Morristown, NJ, 1996. Association for Computational Linguistics. `http://acl.ldc.upenn.edu/C/C96/C96-2182.pdf`.

[4] J. Brundage, M. Kresse, U. Schwall, and A. Storrer. Multiword lexemes: A monolingual and contrastive typology for NLP and MT. Technical Report IWBS 232, IBM Deutschland GmbH, Institut für Wissenbasierte Systeme, Heidelberg, 1992.

[5] D. Janus. Metody przeszukiwania i obrazowania jego wyników w dużych korpusach tekstów. Master's thesis, Uniwersytet Warszawski, Wydział Matematyki, Informatyki i Mechaniki, Warsaw, 2006.

[6] D. Janus and A. Przepiórkowski. Poliqarp 1.0: Some technical aspects of a linguistic search engine for large corpora. In J. Waliński, K. Kredens, and S. Goźdź-Roszkowski, editors, The proceedings of Practical Applications of Linguistic Corpora 2005, Frankfurt am Main, 2006. Peter Lang.

[7] L. Karttunen, J.-P. Chanod, G. Grefenstette, and A. Schiller. Regular expressions for language engineering. Natural Language Engineering, 2(4):305–328, 1996. `http://www2.parc.com/istl/members/karttune/publications/jnle-97/rele.pdf`.

[8] M. McShane, S. Nirenburg, and S. Beale. The description and processing of multiword expressions in ontosem. Technical Report 07-05, Institute for Language and Information Technologies. University of Maryland Baltimore County, 2005. `http://ilit.umbc.edu/ILIT_Working_papers/ILIT_WP_07-05_Multiword_Exprs.pdf`.

[9] M. McShane, S. Nirenburg, and S. Beale. Multi-word entities in human- and machine-oriented lexicons. `http://ilit.umbc.edu/MargePub/multiWordFinal.doc`, 2006.

[10] R. Moszczyński. Formal approaches to multiword lexemes. Master's thesis, Uniwersytet Warszawski, Wydział Neofilologii, Warsaw, 2006. `http://www.mimuw.edu.pl/~jsbien/RM/3301-MGR-FL-A-81032003614.pdf`.

[11] G. Nunberg, I. A. Sag, and T. Wasow. Idioms. Language, 70(3):491–538, 1994. `http://lingo.stanford.edu/sag/papers/idioms.pdf`.

[12] S. Pedrazzini. Phrase Manager: A System for Phrasal and Idiomatic Dictionaries. Georg Olms Verlag, Hildeseim, Zürich, New York, 1994.

[13] A. Przepiórkowski. On heads and coordination in a partial treebank. In J. Hajič and J. Nivre, editors, Proceedings of the TLT 2006, pages 163–174, 2006.

[14] Z. Saloni and M. Świdziński. Składnia współczesnego języka polskiego. Wydawnictwo Naukowe PWN, Warszawa, 4 edition, 1998.

[15] F. Segond and E. Breidt. IDAREX: Formal description of German and French multi-word expressions with finite state technology. Technical Report MLTT-022, Rank Xerox Research Centre, Grenoble, 1995.

[16] F. Segond and P. Tapanainen. Using a finite-state based formalism to identify and generate multiword expressions. Technical Report MLTT-019, Rank Xerox Research Centre, Grenoble, 1995.

[17] C. Tschichold. English multi-word lexemes in a lexical database. In M. F. Verdejo, editor, Proc. of the ESSLI Workshop on the Computational Lexicon, 1995.