POLENG - a Machine Translation System Based on an Electronic Dictionary

Krzysztof Jassem,
Adam Mickiewicz University
Faculty of Mathematics and Computer Science
ul. Matejki 48-49
60-769 Poznañ
jassem@math.amu.edu.pl

ABSTRACT

In [3] a general concept of an MT system - worked out as a result of parallel research on a format of the dictionary and the translation algorithm - was proposed. Interactions between the dictionary and the algorithm were pointed out. Here, a more detailed look into the algorithm is presented. New approach (in relation to that reported in [3]) to the analysis of the verbal group is shown. The new approach is reflected by a new format of verb entries in the dictionary. For the sake of completeness, the main premises of the system - reported in [3] - are recalled.

STRESZCZENIE

W pracy [3] przedstawiona została koncepcja systemu automatycznego tłumaczenia tekstu wypracowanego w wyniku badań prowadzonych równolegle nad formatem słownika i algorytmem translacji. Wskazano powiązania między słownikiem i algorytmem. W niniejszej pracy bardziej szczegółowo opisany jest algorytm tłumaczenia. W stosunku do stanu badań przedstawionego w [3] przyjęto nowe podejście do zagadnienia analizy grupy orzeczenia. Nowe podejście znajduje swoje odzwierciedlenie w nowym formacie haseł czasownikowych w słowniku. Dla pełności opisu podane zostaną główne założenia systemu przedstawione w pracy [3].

1. PREFACE

One of the main limitations of NLP systems is the lack of computer-readable dictionaries. Attempts to overcome this difficulty by designing electronic dictionaries have started in the late 80s. The first electronic dictionary of inflected forms of Polish was completed in 1996. Modern electronic dictionaries satisfy the conditions of automatic generation of inflected forms and quick access. Yet, most of such dictionaries are not supplied with semantic-linguistic information sufficient to be applied ‘directly’ in NLP systems. It seems that in order to obtain an electronic dictionary which can be successfully utilised in an NLP system, research on the format of the dictionary should be co-ordinated with research on the algorithm of NLP analysis.

System POLENG has been designed as a result of parallel research on the Polish-to-English electronic dictionary and the algorithm of translation of Polish text into English. The dictionary contains all and only information required by the algorithm.

2. ELECTRONIC POLISH-TO-ENGLISH DICTIONARY

2.1. Definitions

Definition 2.1

An electronic dictionary is a computer-readable dictionary stored on a hard (magnetic or optical) carrier.

The condition of computer-readability consists of two postulates:

- ♦ the format of the entries should be easy to process by a computing machine
- ♦ the dictionaries should contain all inflected forms (in explicit or implicit form).

In traditional 'human-readable' dictionaries only canonical forms of lexemes are usually included (possibly with hints to inflexion paradigms). Morphological analysis and synthesis is left to human competence. It is a hard task to supply a machine with 'human morphological intuition'. Therefore it is desirable for a computer-readable dictionary to include all inflected forms (inflected forms may be obtained mostly automatically, as will be shown in 2.1).

Definition 2.2 is accepted so as to comply with an MT system.

Definition 2.2.

An electronic bilingual dictionary is a dictionary which includes all and only information required by a translation algorithm.

The definition 2.2 assumes that a dictionary is designed in accordance to a given algorithm. In POLENG the dictionary is one-directional: Polish-to-English, since the corresponding algorithm is also one-directional.

2.2. Three phases of the dictionary

The dictionary is built in three steps. The first step consists in creating the dictionary of canonical forms. The task is carried out by lexicographers on the basis of traditional dictionaries. A computer application facilitates the task of coding lexical information in a computer-readable form. The result of this step is the first phase of the dictionary - the dictionary of canonical forms. The dictionary is stored in a text file and includes two types of information: a code of inflexion paradigm and encoded translation information - needed for the translation algorithm.

The second step is carried out automatically. On the basis of inflexion codes attached to the entries in the dictionary of canonical forms the dictionary of inflected forms is generated. The translation information attached to a canonical form is assigned to all derived inflected forms. The result of this step is the second phase of the dictionary - the dictionary of inflected forms. The structure is stored in a text file and contains redundant information - all inflected forms of one lexeme are assigned the same translation information.

The third step consists in automatically converting the dictionary of inflected forms into a modified finite-state automaton. The information redundancy is eliminated. The structure resulting form this step forms the third and final phase of the dictionary. Details about the structure may be found in [4] and [5]. The translation algorithm extracts information from the final phase of the dictionary.

2.3. Dictionary of canonical forms

2.3.1. Entry format

Entries in the dictionary of canonical forms have the following format:

Polish_entry[]Polish_code
English_entry(1)[]English_code(1)
...
English_entry(n)[]English_code(n)
where [] denotes a space.

For simplicity it is assumed that exactly one English noun and exactly one English adjective corresponds to one Polish noun (adjective). Such simplification is admissible when a specific domain of lexicon is determined. In the system POLENG the assumed domain of lexicon is computer science with elements of translatorics. Hence the format of entries for nouns and adjectives in the current version of the system is as follows:

Polish_entry[]Polish_code[]English_entry[]English_code

The full format of the entry is assigned to verbs. Simplification analogous to that assumed for nouns and adjectives would lead to errors in translation which could hardly be accepted (the problem is discussed closer in 2.4).

2.3.2. Single items and lexical phrases

Two types of entries are admissible in the dictionary:
- single items
- lexical phrases

A *single item* is an entry which includes one word or a reflexive verb in the field *Polish_entry* and includes one word, a reflexive verb or a phrasal verb in all fields *English_entry(k), k = 1...n*. The entry describing the Polish verb *rozgl¹daæ siê* (English: to look around) is a single item in this approach.

A *lexical phrase* is an entry which does not satisfy the condition of a single item.

Lexical phrases are assigned their type. A *type* of a lexical phrase corresponds to a non-terminal symbol in the source language grammar assumed in the translation algorithm.

The dictionary contains only phrases whose translation according to the algorithm would be incorrect.

Example 1.

The noun group 'przep³yw (English: flow) danych' (English: data, in genitive)' is parsed as a 'genitive_group'. According to the applied algorithm such a group is translated into the English expression 'flow of the data'. In order to obtain the idiomatically correct

translation of the phrase, i.e. 'data flow' the phrase should be stored in the dictionary. The type of the phrase in the dictionary is 'genitive_group'.

2.3.3. Coding inflexion properties.

The fields *Polish_code* and *English_code* include inflexion information which enables automatic derivation of inflected forms from canonical forms. The method of generating Polish forms differs from that of generating English forms. This is reflected in inflexion codes.

The process of encoding Polish canonical forms is preceded by preparing classification files (for nouns, verbs and adjectives separately). Each file consists of classes. A class corresponds to a group of nouns (verbs, adjectives) which have the same morphological and syntactical features. Each class is described by its code and the set of endings specific for inflected forms. A code of a canonical form in the dictionary must coincide with one of the codes in the classification file.

Example 2.

The code for the Polish noun 'amplituda' (English: amplitude) is R414;d-dzi. Beneath is an extract from the classification file of Polish nouns:

R4132;0-e = (-,-,-,-,-,-,-)	***%sanki***
(i,e1,om,i,ami,ach,i)	
R414 = (a,y,e,ê,¹,e,o)	
(y,0,om,y,ami,ach,y)	
R414;b-bi = (a,y,ie,ê,¹,ie,o)	***%ryba***
(y,0,om,y,ami,ach,y)	
R414;d-dzi = (a,y,zie,ê,¹,zie,o)	***%rada***
(y,0,om,y,ami,ach,y)	
R414;ch-sz = (cha,chy,sze,chê,ch¹,sze,cho)	***%mucha***
(chy,ch,chom,chy,chami,chach,chy)	

The code of the noun 'amplitude' coincides with that of the noun 'rada' (English: 'advice').

The generation of Polish inflected forms is carried out in the process of creating the dictionary - before the process of analysis in the translation algorithm. The inflected forms of English equivalents are generated in the process of translation.

Example 4

One of the inflected forms of the noun 'amplituda'is 'amplitudy'. The form may denote either genitive, singular or nominative (accusative), plural. Assigning any of the English equivalents 'amplitude' or 'amplitudes' in the dictionary might prove wrong when applied in the translation algorithm. Therefore generating the adequate inflected form (i.e. adding the suffix 's' or not) is left to the translation algorithm.

Such approach requires the inflexion part of *English_code* to be *constructive.*

DEF. 3. A code is called *constructive* if it enables the generation of inflected forms without the necessity of looking up a classification file.

Such assumption eliminates the time-consuming process of looking up a classification file during translation process. Examples of constructive codes will be given in the next paragraph.

2.3.4. Examples of entries in the dictionary of canonical forms.

In this paragraph examples of noun entries, adjective entries and phrasal entries will be presented.

Example 5.
Beneath a few noun entries are chosen:

*1) aksjomat **R314;t-ci** axiom **N1***
*2) automat **R314;t-ci** automaton **N2a***
*3) dodatek **R301;k-ki** appendix **N1ces***
*4) informacja **R423** information **N00***
*5) u¿ytkowniczka **R314;k-c** user **N1P***

Details on encoding Polish nouns may be found in [7]. English codes of the above nouns have the following meaning. The code N1 denotes regular nouns whose plural forms are generated from singular forms by adding the suffix 's'. (the letter 'P' at the end of the code in entry 5) marks a 'personal' noun). The code N00 is assigned to nouns which do not have plural forms. Codes N2a and N1ces denote irregular nouns. The plural form for irregular nouns is constructed in the following way: 1)'n' last characters are cut from the singular form, where 'n' is the digit in the code, 2) the string thus obtained is merged with the string of characters which follow the digit in the code. For example, the plural form of the form 'appendix' is obtained by cutting one character from the end -yielding 'appendi'- and merging the string 'appendi' with the string 'ces' - yielding 'appendices'.

Example 6
Beneath two examples of adjective entries are shown:

*1) d³ugi **P10** long **A1a***
*2) nieregularny **P351n** irregular **A0***

Entry 1) features a canonical form of the adjective 'd³ugi' from which inflected forms (here, an adverb derived from an adjective is also treated as its inflected form) are generated according to the paradigm P10 (specified in the classification file). The English equivalent has regular comparative and superlative forms (this property is marked as A1). The character 'a' at the end of the English code denotes the irregular (orthographically equal to the adjective) form of the derived adverb. Entry 2) describes an adjective which

does not form comparative and superlative degrees either in Polish (denoted by the character 'n' at the end of the code) or English (code A0).

Example 7
Now, a few examples of lexical phrases will be shown.

informacja wprowadzana do komputera ***R423*** *information input* ***N00***
liczba ca³kowita ***R414;b-bi*** *integer* ***N1***
³¹cze transmisji danych ***R532*** *information link* ***N1***
przetwarzanie informacji ***R530;ni-ñ*** *information processing* ***N00***
obszar wejœcia ***R314;r-rz*** *input area* ***N1***

A lexical phrase is assigned a code of the part of speech which it can replace in a sentence.
Information about the type of a lexical phrase is attached in the process of generating inflected forms.

2.4. Verb requirements

As mentioned in 2.1. the treatment of verbs differs from that of nouns and adjectives. The full format of an entry is assumed for verbs. The following definitions enable specific treatment of verbs in a bilingual dictionary as well as in a translation algorithm.

2.4. 1.What is a verb requirement?

DEF. 4. *Syntactical requirement* of a verb is a part of a sentence which is dependent on the verb and which is not translated correctly in a compositional way.

DEF. 5. *Syntactical-semantic requirement* of a verb is a pair: a syntactical requirement and a semantic condition.

Further on, the term 'requirement' will be understood as a 'syntactical-semantic requirement'.

Example 8
The Polish verb 'mówiæ' has 4 English equivalents depending on context: 'to speak', 'to tell', 'to say', 'to talk'. The correct translation of the expression 'mówi³ z tob¹' is ' he talked to you'. If the algorithm is allowed to translate phrases: 'mówi³' and 'z tob¹' in a compositional way then the output could take one of the following forms: 'he spoke with you', 'he told with you', 'he said with you'.
The expression 'pracowa³ z tob¹', on the other hand, is translated correctly in a compositional way: 'he worked with you'.
It is assumed here that a prepositional phrase starting with 'z' (English: 'with') is a requirement of the verb 'mówiæ' and it is not a requirement of the verb 'pracowaæ'.

Example 9.

The requirement of the verb 't³umaczyæ (coœ)' (English: 'to explain' (something)) is a pair: syntactical requirement of the noun phrase in accusative and the semantic condition of the noun phrase being non-personal. The requirement of the verb 't³umaczyæ (kogoœ)' (English: 'to excuse (someone)') is a pair: syntactical requirement of the noun phrase in the accusative and the semantic condition of the noun phrase being personal.

The translation algorithm benefits from taking verb requirements into account in the following aspects:

- requirements of Polish verbs enable correct analysis of source text
- requirements of English equivalents enable correct synthesis of target text
- requirements make it possible to determine the meaning of a verb and choose the adequate equivalent from a few possibilities.

2.4.2. Types of requirements of Polish verbs

Table 1 lists types of requirements of Polish verbs assumed in the system POLENG.

Marking	Type of requirement	Example of a sentence
D	noun phrase - genitive	Szukam ksi¹¿ki (I am looking for a book)
C	noun phrase - dative	Mówiê ci (I am telling you)
B	noun phrase - accus.	Widzê ciê (I can see you)
N	noun phrase - instrum.	Bawiê siê pi³k¹ (I am playing with a ball)
Ms	noun phrase - locative	
BK	infinitive	Chcê odjechaæ (I want to leave)
PM	adjective	On sta³ siê niebezpieczny (He has become dangerous)
PS	adverb	Zachowuje siê Ÿle (He behaves badly (He misbehaves))
¯E	rel. clause starting with '¿e'	Powiedzia³, ¿e przyjdzie (He said he would come)
BY	rel. clause starting with 'by'	Chcê, ¿ebyœ tu by³ (I want you to be here)
JK	rel. clause starting with 'jak'	S³yszê, jak on gra (I can hear him playing)
PT	object rel. clause	Wiem, kiedy (gdzie) on przyjdzie (I know when (where) he will come)
OM	locative adverbial phrase	Zostanê w domu (I shall stay at home)
prepC¹	prepositional phrase	Mówiê do ciebie (I am talking to you)

[1]Prepositinal phrases are coded with a string consisting of a preposition (in small letters) and the first letter of the case of a noun phrase (e.g. doD marks a prepositional group consisting of a preposition 'do' (English: 'to') and a noun phrase in genitive (Polish: 'dope³niacz').

Table 1

2.4.3. Types of requirements of English verbs

It is assumed that for each requirement of a Polish verb there corresponds a requirement of the equivalent English verb. Table 2 lists types of requirements of English verbs.

Marking	Type of requirement	Example of a sentence satisfying the requirement
0 (zero)	noun phrase	I like books
IN	infinitive	I must go
SI	'someone' + infinitive	I made him cry
TO	'to' + infinitive	I want to go
ST	'someone' + 'to' + infinitive	I want you to go
NG	'ing' partciciple	I like singing
SN	'someone' + 'ing' participle	I could see him crying
AD	adjective	he felt ill
AV	adverb	he behaves properly
QU	object rel. clause	I know what you like
TH	relative clause: 'that'	He said (that) he would come
LC	locative prepositional phrase	I am staying at home
prep[1]	prepositional phrase	This corresponds to something else.

[1]prepositional phrases are coded by naming the preposition

Table 2

2.4.4. Combinations of requirements.

Verbs may have more requirements than one. The order of requirements in Polish sentences is almost arbitrary. Moreover, requirements may precede the verb. The order of requirements in an English sentence is usually fixed and requirements are preceded by the verb. In the state of research reported in [3] requirements of a verb (if there existed more than one) were listed in a dictionary in an arbitrary order. The translation algorithm checked for occurrence of each permutation of verb requirements in a sentence. The algorithm proved ineffective. Moreover, the algorithm did not deal with sentences in which requirements preceded the verb. The solution, assumed here, benefits from the fact that not all permutations of requirements for a given word are likely to occur in a Polish sentence. All admissible orders of requirements specific for a Polish verb are coded in the dictionary. The translation algorithm is now able to verify all possible hypotheses. Precedence of a constitutive verb to its requirements in a sentence is no longer required.

Combinations of requirements are listed in Table 3 and Table 4.

Two requirements may be linked with symbols shown in Table 3.

Character	**Interpretation**

+	**Conjunction** (both requirements must occur in a sentence)
,	**Alternative** (at least one requirement occurs in a sentence)
;	**Disjunction** (exactly one of two requirements occurs in a sentence

Table 3

Groups of requirements are bounded with the following symbols:

Pair of symbols	Interpretation
{ }	Each order of requirements is admissible in a Polish sentence
[ ]	The order of requirements is rigid
< >	Optional (not obligatory) occurrence of the group of requirements in a sentence
( )	Order is irrelevant (concerns disjunction and single requirements)

Table 4

2.4.5. Examples of verb entries in the dictionary of canonical forms

Here are rules of describing verb requirements in the dictionary:

- one description of English verb requirements corresponds to one description of Polish verb requirements,
- corresponding descriptions are placed in the same line and are separated by a hyphen,
- the order of the requirements is consistent with the order of their realisation in the English sentence (symbols given in Table 4 inform of possible permutations in a Polish sentence)
- characters linking Polish requirements are repeated in English requirements

Example 9
The entry describing the verb 'mysleæ' (English: to think) has the following form:

myæleæ C31N
think V3ought3ought
<oMs> - <of>
(¯E) - (TH)N

The first line contains information about the inflexion class ('C31') and the property of imperfection (character 'N').

The second line contains the English equivalent as well as its constructive code of inflexion. The string 'V3ought3ought' describes the way to obtain Past and Participle forms. In order to obtain the Past form 3 last letters should be cut from the end of the infinitive form - yielding the string 'th'- then the string 'th' should be concatenated with the string 'ought' - yielding the form 'thought' (the Participle form is obtained similarly).

The last two lines characterise verb requirements. The third line concerns the requirement of a prepositional phrase consisting of the preposition 'o' and a noun phrase in

*locative. The corresponding requirement of the English equivalent is the prepositional phrase starting with 'of'. Hence the translating algorithm will translate the sentence 'Myœla³em **o** tobie' into the sentence 'I thought **of** you'. The brackets '< >' indicate that constructions with the verb 'myœleæ' without requirements are also admissible (e.g. Myœlê - I am thinking). The fourth line concerns constructions of the type 'Myœlê, ¿e...'. The character 'N' at the end of the line indicates that in connection with a given requirement the verb does not form continuos forms. Thus a sentence 'Myœlê, ¿e przyjdzie' is translated into 'I think (that) he will come' (Present Simple).*

Example 10

The following entry is given in the dictionary of canonical forms (numbers of lines are added for convenience):

1) ***mówiæ C51N***
2) ***speak V3oke3oken***
3) ***<{PS,oMs}> - <{AV,about}>***
4) ***tell V3old3old***
5) ***{C + <B;oMs>} - {0 + <0;about>}***
6) ***[C + ¯E] - [0 + TH]***
7) ***talk V1***
8) ***{(doD;zN) + <oMs>} - {(to;to) + <about>}***
9) ***say V1id1id***
10) ***(B) - (0)***
11) ***(¯E) - (TH0)N***

Here is the interpretation of the entry:

A form of the verb 'mówiæ' should be translated into an appropriate form of the verb 'to speak' in one of the following cases:

- the verb appears without requirements (e.g. in the sentence 'Umiem mówiæ' - 'I can speak'). This is determined by the characters '< >' in the third line.

- the verb appears in the sentence with the requirement of an adverb (e.g. 'Mówiê g³oœno' - 'I speak loudly') or with the requirement of a prepositional phrase 'o' e.g. 'Mówiê o historii Polski' - 'I speak about the history of Poland', or with both requirements in any order. If both requirements occur in the sentence, then the order of the corresponding requirements in the English sentence is consistent with the order in the dictionary (i.e. the adverb precedes the prepositional phrase).

A form of the verb 'mówiæ' should be translated into an appropriate form of the verb 'to tell' if in a sentence the form appears with the requirement of a noun phrase in dative. Moreover a sentence may include either a noun phrase in accusative, or a prepositional phrase 'o'. The order of the requirements is arbitrary. However, the order of requirements in sentences 'Mówiê ci, ¿e...' - 'I tell you that...' is rigid (noun phrase in dative must precede a relative clause). Therefore the requirements in line 6) are bounded by markers of rigid order ('[ ]').

A form of the verb 'mówiæ' should be translated into an appropriate form of the verb 'to talk' if in a sentence the form appears with the requirement of a prepositional phrase

‘do’, e.g. ‘Mówiê do ciebie’ - I am talking to you’ or a prepositional phrase ‘z’, e.g. ‘Mówiê z tob¹’ - ‘I am talking to you’. Optionally a prepositional phrase ‘o’ may appear in the sentence.

A form of the verb ‘mówiæ’ should be translated into an appropriate form of the verb ‘to say’ if in a sentence the form appears with the requirement of a noun phrase in accusative, e.g. ‘Mówi³ coœ’ - ‘He said something’ (this is a simplification, e.g. the phrase ‘mówiæ prawdê’ should be translated into ‘to tell the truth’. The way to deal with such phrases is to insert them into the dictionary as lexical phrases). In constructions of the type ‘mówiæ ¿e...’ - ‘to say that...’ the English verb does not appear in continuos forms (which is marked with the final ‘N’).

2.5. The dictionary of inflected forms

The dictionary of inflected forms is generated automatically from the dictionary of canonical forms. One canonical form generates:

- up to 14 forms of a noun
- up to 30 forms of a verb (including participles, and nouns derived from verbs)
- up to 37 forms of a noun (including adverbs) - the phenomenon of syncretism is taken into account in order not to produce duplicate forms.

An entry in the dictionary of inflected forms has the following format:

IF\ID\MI\CP\PI/RS$_1$/CE$_1$/EI$_1$ [|/RS$_2$/CE$_2$/EI$_2$ [...]]

The brackets ‘[ ]’ denote optional occurrence of a few different English equivalents (currently this concerns only verbs).

IF - inflected form
ID - identifier
MI - morphological information
CP - canonical Polish form
PI - Polish inflexion code
RS - requirements string
CE - canonical English form
EI - English inflexion code

During the generation process the strings of characters marking verb requirements for one English equivalent are concatenated (the character ‘&’ is inserted between concatenated strings).

Example 11
The inflected form ‘mówiê’ (Present tense, singular, 1st person) is generated from the canonical form ‘mówiæ’. Beneath the full form of the entry describing the form ‘mówiê’ is given.

mówiê\1050\OP1\mówiæ\C51N\<{PS,oMs}>-<{AV,about}>/speak/V3okeV3oken|

{C+<B;oMs>}-{0+<0;about>}&[C+¯E]-[0+TH]/tell/V3old3old|
{(doD;zN)+<oMs>}-{(to;to)+<about>}/talk/V1|
(B)-(0)&(¯E)-(TH)N/say/V1id1id
(for better readability the entry is divided into 4 lines)

Example 12
The inflected form 'pamiêci' - which is common for a few cases in the declension of the noun 'pamiêæ' (English: memory) is represented in the dictionary in the following form:

pamiêci\3380\¯DP:¯CP:¯MsP:¯MM:¯DM:¯BM:¯WM\pamiêæ\N442;æ-ci\0/memory/E3

Details on the value of the MI field may be found in [7].
The value of the only RS (requirement string) field is equal to 0. Syntax - semantic categories which may be included in the field have not been determined yet.

2.6. Dictionary-automaton

The dictionary of inflected forms is converted into a modified finite-state automaton in order to reduce access time. The format of the dictionary-automaton is described in detail in [5]. The translation algorithm extracts information from the automaton.

2.7. Other parts of speech

Only nouns, verbs, adjectives, ordinal numerals and adjective pronouns may be stored in the dictionary of canonical forms. Some other pats of speech (participles, adverbs) appear in the dictionary of inflected forms. The remaining parts of speech are included in 'an internal dictionary' of the translation algorithm. They are stored in the form of PROLOG clauses and are loaded to RAM memory whilst starting the translation program. The following facts stand behind such approach:

- the Polish lexicon is small enough to be stored in RAM memory after eliminating forms derivable from nouns, verbs and adjectives
- there is no point to turn on the 'generative machinery' for parts of speech which are not flexional
- conjunctions, prepositions, particles, pronouns play a special role in the translation process and they have to be supplied with a specific kind of information.

3. TRANSLATION ALGORITHM

The translation algorithm is based on the parser of Polish expressions described in [6]. The parser uses DCG grammar. The rules of the parser are written in the form of Horn's clauses in order to be implemented by the PROLOG machinery. The reconstruction of the parser in order to implement it in the translation system consisted in:

- introducing a parameter of the phrasal structure of the English output
- introducing new non-terminal symbols
- widening sets of parameters
- introducing the module of extracting information from the electronic dictionary
- solving the problem of the left-hand recursion
- changing the approach to analysis of verbal groups (DCG grammar rules proved ineffective).

The algorithm is not modular. It does not produce any intermediate structures. The only product of the analysis of a Polish expression is a functor representing a phrasal structure tree of the corresponding English expression.

The scheme of the algorithm is given in Figure 3.

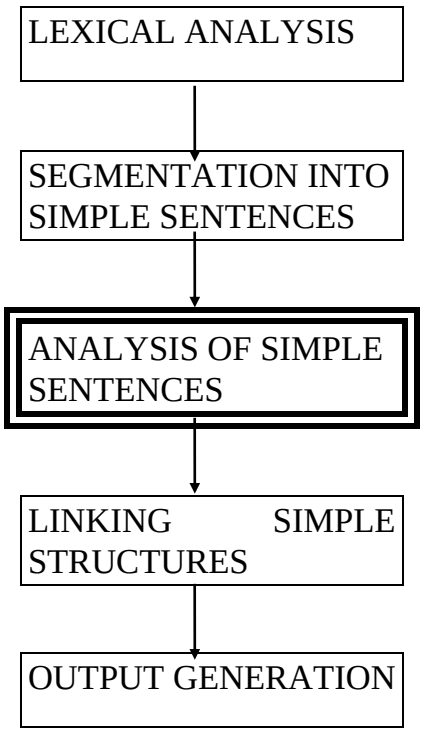

Figure 3.

3.1. Lexical analysis.

During lexical analysis the Polish input expression is segmented into single words. Words are searched first in the 'internal PROLOG dictionary' and second in the dictionary-automaton described in 2. If a word is not found, the analysis fails. Occurrence of lexical phrases is not verified in this phase.

If a word found in the dictionary is derived from either a noun or an adjective canonical form, one PROLOG clause is added to the PROLOG database. If a word found in the dictionary is derived from a verb infinitive, the PROLOG database is enriched with as many clauses as there may exist possible distributions of verb requirements (according to the dictionary).

Example 13

The following clauses are generated if a lexical analysis deals with the word 'mówiê'. Each clause corresponds to one possible distribution of requirements in a sentence (compare examples 10 and 11).

verb1(1,[poj,o1,ozn,orzecznik],[niedokon],
[¿e_],[th_],[say,$said$,$said$],[mówiê|A],A).

verb1(1,[poj,o1,ozn,orzecznik],[niedokon],
[b],[no],[say,$said$,$said$],[mówiê|A],A).
(*) verb1(2,[poj,o1,ozn,orzecznik],[niedokon],
[oMsORD2,zNORD1],[to,about],[$talk$,$talked$,$talked$],[mówiê|A],A).
verb1(2,[poj,o1,ozn,orzecznik],[niedokon],
[zN,oMs],[to,about],[$talk$,$talked$,$talked$],[mówiê|A],A).
verb1(2,[poj,o1,ozn,orzecznik],[niedokon],
[oMsORD2,doDORD1],[to,about],[$talk$,$talked$,$talked$],[mówiê|A],A).
verb1(2,[poj,o1,ozn,orzecznik],[niedokon],
[doD,oMs],[to,about],[$talk$,$talked$,$talked$],[mówiê|A],A).
verb1(1,[poj,o1,ozn,orzecznik],[niedokon],
[oMs],[about],[$talk$,$talked$,$talked$],[mówiê|A],A).
verb1(1,[poj,o1,ozn,orzecznik],[niedokon],
[zN],[to],[$talk$,$talked$,$talked$],[mówiê|A],A).
verb1(1,[poj,o1,ozn,orzecznik],[niedokon],
[doD],[to],[$talk$,$talked$,$talked$],[mówiê|A],A).
verb1(2,[poj,o1,ozn,orzecznik],[niedokon],
[c,¿e_],[no,th_],[$tell$,$told$,$told$],[mówiê|A],A).
verb1(2,[poj,o1,ozn,orzecznik],[niedokon],
[oMsORD2,cORD1],[no,about],[$tell$,$told$,$told$],[mówiê|A],A).
verb1(2,[poj,o1,ozn,orzecznik],[niedokon],
[c,oMs],[no,about],[$tell$,$told$,$told$],[mówiê|A],A).
verb1(2,[poj,o1,ozn,orzecznik],[niedokon],
[bORD2,cORD1],[no,no],[$tell$,$told$,$told$],[mówiê|A],A).
verb1(2,[poj,o1,ozn,orzecznik],[niedokon],
[c,b],[no,no],[$tell$,$told$,$told$],[mówiê|A],A).
verb1(1,[poj,o1,ozn,orzecznik],[niedokon],
[c],[no],[$tell$,$told$,$told$],[mówiê|A],A).
verb1(2,[poj,o1,ozn,orzecznik],[niedokon],
[oMsORD2,ps_ORD1],[av_,about],[$speak$,$spoke$,$spoken$],[mówiê|A],A).
verb1(2,[poj,o1,ozn,orzecznik],[niedokon],
[ps_,oMs],[av_,about],[$speak$,$spoke$,$spoken$],[mówiê|A],A).
verb1(1,[poj,o1,ozn,orzecznik],[niedokon],
[oMs],[about],[$speak$,$spoke$,$spoken$],[mówiê|A],A).
verb1(1,[poj,o1,ozn,orzecznik],[niedokon],
[ps_],[av_],[$speak$,$spoke$,$spoken$],[mówiê|A],A).
verb1(0,[poj,o1,ozn,orzecznik],[niedokon],
[],[],[$speak$,$spoke$,$spoken$],[mówiê|A],A).

Let us discuss one of the clauses:
(*) verb1(2,[poj,o1,ozn,orzecznik],[niedokon],
[oMsORD2,zNORD1],[to,about],[$talk$,$talked$,$talked$],[mówiê|A],A).

The first argument is equal to the number of requirements described by the clause. The above clause deals with 2 requirements. The second parameter is the list of morphological features (singular,1st person, indicative mode, present tense). The third argument stands for 'imperfect, not reflexive', the next two arguments describe requirements. Here, two requirements are encoded: the requirement of a prepositional phrase consisting of a preposition 'o' and a noun phrase in the locative and the requirement of a prepositional phrase consisting of the preposition 'z' and a noun phrase in the instrumental. The strings ORD2 and ORD1 mark the inversion of the requirements order in the English syntax, e.g. the first Polish requirement 'oMs' corresponds to the English requirement 'about' (PP beginning with the preposition 'about') and should be preceded by PP 'to' (corresponding to Polish PP 'zN') in the English output.

Searching lexical phrases is executed during analysis of simple sentences (see 3.3).

3.2. Segmentation into simple sentences

The problem of tenses is one of the most severe difficulties of automatic translation from Polish to English. There exist only three tenses in Polish grammar (present, future, past). Working out an algorithm choosing an appropriate tense for an English equivalent of a Polish sentence is a difficult task. A few techniques have been applied here. One of them takes advantage of information stored in the dictionary in order to choose between a continuos or a simple form. Another technique consists in analysing time adverbial phrases. These techniques would not succeed in determining a proper tense of clauses in a complex sentence. A technique based on the surface structure of the sentence has been assumed for complex sentences. The technique consists in supplying non-terminal symbols which correspond to simple sentences with an additional 'tense shifting' parameter.

Example 14
The rule given below segments a sentence into simple sentences:

```
% Schemat: dopóki <zdanie szeregowe> dopóty <zdanie szeregowe>

sentence(as_long_as(T1,T2)) -->
        left_conj(7),                          %dopóki
        simple_sentence(Cz1,shift_to_present,Neg1,T1),
        right_conj(7),                         %dopóty
        simple_sentence(Cz2,_,Neg2,T2).
```

The parameter 'shift_to_present' 'tells' the algorithm to assign Present Tense to the first simple_sentence if it is either in Present or Future Tense. According to this technique both Polish sentences 'Dopóki ktoœ mnie s³ucha (Present tense), dopóty nie przestanê mówiæ' and 'Dopóki ktoœ mnie bêdzie s³ucha³ (Future Tense), dopóty nie przestanê mówiæ' will be translated into the same English sentence 'As long as anyone is listening to me, I shan't stop talking'.

3.3. Analysis of a simple sentence

A simple sentence consists of a verbal phrase and optionally a noun phrase and an adverbial phrase. The three phrases may be formed in any order. To execute the analysis of a simple sentence rules for analysing verbal phrases, noun phrases, adjective phrases and adverbial phrases must be defined. The three last types of phrases are parsed according to rules of DCG grammar by a top-down, left-first parser. The algorithm of parsing verbal phrases is different and will be presented in 3.3.3.

3.3.1. Translating with DCG rules

In order to obtain a structure of an English phrase corresponding to a Polish phrase each non-terminal symbol is supplied with an additional parameter - the last parameter of non-terminal symbols. Example 15 shows the way in which the parameter (the last argument) is made to store information about English - rather than Polish - structure.

Example 15

Two following rules (presented here in a simplified form) enable parsing 'attributive noun phrases' which consist of an adjective phrase and a noun construction.

```
attr_noun_phr(P,C,R,G,L,O, adj_gr(T1,T2)) -->
                                adj_phr(P1,R1,L1,_,T1),
                                noun_constr(P,C,R,G,L,O,T2),
                                {syncretism(P1,R1,L1,P,R,L)}.

attr_noun_phr(P,C,R,G,L,O,adj_gr(T2,T1) -->
                                noun_constr(P,C,R,G,L,O,T1),
                                adj_phr(P1,R1,L1,_,T2),
                                {syncretism(P1,R1,L1,P,R,L)}.
```

The first rule deals with constructions in which an adjective phrase precedes a noun phrase. The second rule deals with the inverse order of phrases. Both types of constructions assign the same value to the 'structural parameter'; the value describes the structure in which an adjective phrase precedes a noun phrase (this order is assumed in the algorithm as the only correct one for English noun phrases).

3.3.2. Dealing with left-handed recursion

The system described in [7 included rules with left-handed recursion. In order to implement the rules in a top-down parser, the problem of left-hand recursion had to be solved. The scheme to avoid the recurrence, assumed here, consists in defining a partial order on non-terminal symbols. The only admissible productions in a new grammar are of the form $A \rightarrow BC$, where B is a terminal or B is not less or equal A according to the order

(the constructive proof that such grammar can always be designed may be found in [1]). Example 16 shows the order of terminals used to describe a noun phrase.

Example 16

Ascending order of non-terminal noun symbols:

SYMBOL	*DESCRIPTION*
noun_phr_series	*a series of noun phrases; consists of a few noun_phrases linked by specific 'two side' conjunctivas*
noun_phr	*a noun phrase; consists of a few noun 'sub noun phrases' linked by conjunctivas of alternative or conjunction*
sub_noun_phr	*sub noun phrase; consists of a 'simple noun phrase' modified by a participle or a prepositional phrase*
simple_noun_phr	*simple noun phrase; consists of a main numeral and a noun construction*
clause_noun_constr	*clause noun construction; consists of a 'genitive construction' optionally modified by a clause*
genitive_constr	*genitive construction; consists of an 'attributive noun series' and (optionally) a noun phrase series in the genitive*
attr_noun_series	*attributive noun series; consists of a few attributive noun phrases*
attr_noun_phr	*attributive noun phrase; consists of an adjective phrase and a noun construction*
noun_constr	*noun construction; consists of a 'simple' noun or a special verbal phrase (concerns gerunds)*

Table 5

(The problem of pronouns is left out here).

Nestled constructions are obtained by means of right-hand recursion.

Example 17

The example shows two rules which implement the phenomenon of 'nestling' by means of right-hand recursion (parameters are omitted here):

noun_phr → sub_noun_phr, conjunctive, noun_phr

genitive_constr → attr_noun_series, noun_phr_series(genitive)

3.4. Verbal group parsing

Top-down-left-first parsing is not effective for parsing Polish verb groups. This is due to free order of Polish sentences: verb requirements may appear in a sentence before a verb. For example, in the sentence 'On mi o tym nie powiedzia³' (English: 'He didn't tell me about it') the requirements of the verb 'powiedzia³' are realised by a noun group 'mi' (English: 'me') and the PP 'o tym' (English: 'about it'). Both phrases precede the verb in the sentence. If TDLF parsing were assumed for verb groups, then the occurrence of all possible variants of requirements would have to be verified - including requirements not admissible (according to the dictionary) for the constitutive verb. Such approach seemed inefficient. The algorithm presented below proved efficient enough to translate Polish sentences in the near-real time.

3.4.1. Algorithm of parsing a verbal group

Phr is a sequence of words to be parsed by the algorithm.

1. Find the first verb *Verb1* in *Phr*. Denote the sequence of verbs from the beginning of the phrase *Phr* to *Verb1* (exclusive) as *Sequence1*.
2. If one of possible requirements of *Verb1* is infinitive form of another verb
then go to **3**
else go to **4**
3. Look for an infinitive in the sequence of verbs following *Verb1*.
If an infinitive is found
then
a) denote the found infinitive as *Verb2*
b) denote the sequence between *Verb1* and *Verb2* as *Sequence2*
c) take *Sequence := Sequence1 + Sequence2* (concatenation)
d) take *Verb:= Verb2*

{To visualise this point of the algorithm let us consider the sentence 'Ja (I) mu (him) chcę (want) o tym (about it) powiedzieć (tell)'. Both the direct object ('mu') and the indirect object ('o tym') are the requirements of the verb 'powiedzieć'. According to the algorithm: 1) 'mu' is denoted as *Sequence1 3*a) 'powiedzieć' is denoted as *Verb2, 3*b) 'o tym' is denoted as *Sequence2, 3c) Sequence* has the value *'mu o tym'* , 3d) *Verb* has the value *powiedzieć*

else
e) take *Sequence := Sequence1*
f) *Verb := Verb1*

4.
It is assumed that *Sequence* may consist of the following elements:
realisation of requirements of:
a noun phrase
a prepositional phrase
an adverb
auxiliary forms of the verb 'być' ('to be') in future tense

the reflexive pronoun ‘się’ (‘oneself’)
the particle ‘nie’ (‘not’)
other adverbial phrases

Sequence is parsed by means of the predicate *PREMODIFIERS* whose task is to collect all constituents of *Sequence* which may be requirements of a verb into the list *ReqList* and the rest of constituents into the list *ModList*

For the following sentences: ‘Ja (I) często (often) o tym (about it) mówię (speak?,tell?) and ‘Ja (I) często (often) o tym (about it) mówię (speak?, tell?) swojej (my) matce (mother) the lists found by the predicate *PREMODIFIERS* are the same and consist of one element each: *ModList* = [[freq_adv],[often]], *ReqList* = [[oMs, pron(it)]]}

5a) Consider clauses *verb1* (generated in lexical analysis) characteristic for *Verb* which have the largest value of the first parameter (the parameter describes the number of the verb requirements).

For the verb ***mówię*** the verb1-clauses under consideration will have the first parameter equal to 2.

b) Among the set of verb1-clauses under consideration choose clauses in which the beginning of the Polish requirement list (4th argument) coincides with *ReqList.*

For both sentences at issue only one verb1-clause will fit:

verb1(2,[poj,o1,ozn,orzecznik], [niedokon],
[oMsORD2,cORD1], [no,about], [$tell$,$told$,$told$], [mówię|A], A).

if failure (no clauses found)
then go to **7**
else
for each chosen verb1-clause
denote the list of requirements yet to fulfil as *ReqRest;*

RecRest for both sentences (and the only chosen verb1-clause) will consist of the requirement of a noun phrase in dative.

verify by means of the *POSTMODIFIERS* predicate whether *ReqRest* is realised by the sequence of words following *Verb.*
if yes
then go to **8**
else go to **6**

ReqRest will be realised by the sequence ‘swojej matce’ occurring in the second sentence and will be not realised by the empty sequence in the first sentence.

6. If number_of_requirements (assumed in **5.) > 0**
then decrease number_of_requirements by 1 and go back to **5b**
else go to **7**

The verb1-clause chosen in 5b with number_of_requirements equal to 1 for the sentence ‘Ja często o tym mówię’ will be:

verb1(1,[poj,o1,ozn,orzeznik], [niedokon],
[oMs], [about], [$speak$, $spoke$, $spoken$], [mówię|A], A)

7.

This point of the algorithm is reached only if the requirements in *Reqlist* have fulfilled none of the requirements in verb1-clauses.

If *ReqList* is empty
then failure

The analysis fails only if a verb is assumed to have an obligatory requirement which is not fulfilled in the analysed sentence.

else move the first element of *ReqList to ModList* and go back to **5b**

Let us assume that the verb 'pracować' (to work) has no requirements in the dictionary (see 2.3). If a sentence 'Ja (I) z tobą (with you) pracuję (work)' is parsed, the *PREMODIFIERS* predicate inserts the PP 'z tobą' into *ReqList.* The algorithm reaches point 7. The PP 'z tobą' is moved to *ModList* and *RecList* becomes an empty list. The condition in 5b is satisfied.

8. a) Determine the order of adverbial phrases according to rules of English syntax.
b) Determine the order of English verb requirements according to the dictionary information (currently stored in verb1 clauses).

3.4.2. Special verbal constructions

Special verbal constructions are used when one of the verb requirements is realised 'outside' the verbal phrase. There are two types of special verbal constructions:

- **(-R) constructions**
- **R-shifted constructions**

DEF. 4. A verbal construction **VC** is called **(-R) construction** if there exists a verb1-clause **V1** for the constitutive verb of the construction **VC** such that:

- the list of Polish requirement in **V1** includes **R**
- all requirements but **R** from the list are fulfilled in **VC**

DEF. 5. A verbal construction **VC** is called **R-shifted construction** if the requirement **R** in **VC** has to be transferred into the subject of the English sentence.

3.4.3. Algorithm of parsing a (-R) construction

The algorithm is analogous to that of parsing a 'regular' verb group. The differences occur at point 5b and point 8b (we denote a missing requirement as 'R').

5b) Among the considered set of clauses choose clauses which include R in their list of Polish requirements (4^{th} argument). Delete R from the list. Choose clauses in which the beginning of the list thus obtained coincides with *ReqList.*

8) Do not take into account the English requirement corresponding to R while determining the order of English requirements.

3.4.4. Algorithm of parsing a R-shifted construction

The algorithm is analogous to that of parsing a (-R) construction. The main differences occur in predicates PREMODIFIERS and POSTMODIFIERS. For both 'regular' and (-R) constructions the predicates determine the English equivalents of verb objects (e.g. the equivalent of the pronoun 'mnie' is 'me'). In the R-shifted constructions the object satisfying the conditions of the requirement R is not assigned its English equivalent (e.g. in the sentence 'Poinformowano mnie' (I have been informed) the word 'mnie' is not transferred into the pronoun 'me' by the POSTMODIFIERS predicate).

3.4.5. Applications of special verbal constructions.

Passive participles

A passive participle which modifies a noun phrase is a (-B) construction, where B marks a requirement of a noun phrase in accusative.

Example 17.
The phrase given below is a noun modified by a passive participle:
'Dialog t³umaczony na jêzyk angielski' ('A dialogue translated into English').
The verb 't³umaczyæ' (to translate) which appears in the above phrase has two requirements: a requirement of an NP in accusative and the requirement of a PP 'na'. Only one of the requirements (that of PP) appears inside the verb phrase. Therefore the verb phrase is parsed as a (-B) construction.

Passive voice

Verbal phrases in passive voice are also treated as (-B) constructions.

Example 18
According to the dictionary, two English verbs correspond to the Polish verb 'przedstawiaæ'. If the verb requirement is a personal noun phrase, then the English equivalent is 'to introduce'. If the requirement is not personal, then the English equivalent is 'to present'. Let us compare two Polish sentences:
Ten mêzczyzna nie zosta³ jeszcze przedstawiony.
Ten dowód nie zosta³ jeszcze przedstawiony.

The first sentence should be translated into: 'That man has not been introduced yet'. The second sentence should be translated into: That proof has not been presented yet'. The algorithm is able to overcome this difficulty. While parsing (-R) constructions the algorithm checks whether in the verb1-clause (generated from dictionary information) the syntactical part of the requirement is linked with the semantic condition.

Relative clauses

There are four types of relative clauses which contain (-R) constructions. Table 6 lists all of the types.

TYPE OF 'MISSING' REQUIREMENT	TYPE OF RELATIVE CLAUSE	EXAMPLE OF A POLISH CLAUSE	EXAMPLE OF AN ENGLISH CLAUSE
noun phrase	'który' type	nauczyciel, którego lubiê	the teacher I like
prep. phrase	'który' type	mê¿czyzna, z którym rozmawiam	a man I am talking to
noun phrase	'kto type	wiem, czego szukasz	I know what you are looking for
prep. phrase	'kto' type	wiem, o kim myœlisz	I know who you are thinking of

Table 6

The missing requirement of a Polish noun phrase may correspond to an English prepositional phrase (see Table 6, row 4) and conversely.

Questions

There are four types of questions which contain (-R) constructions. Table 7 lists all of the types.

TYPE OF 'MISSING' REQUIREMENT	TYPE OF QUESTION	EXAMPLE OF A POLISH QUESTION	EXAMPLE OF AN ENGLISH QUESTION
noun phrase	'który' type	Jakich nauczycieli lubisz?	What teachers do you like?
prep. phrase	'który' type	Z któr¹ pani¹ rozmawia³eœ	Which women did you talk to?
noun phrase	'kto type	Czego szukasz?	What are you looking for?
prep. phrase	'kto' type	O kim myœlisz?	Who are you thinking of?

Table 7

Clauses with shifted subjects

This concerns clauses in which a noun phrase playing a role of a requirement in a Polish clause becomes a subject in the corresponding English clause. Analysis of such clauses includes parsing of R-shifted constructions.

'SHIFTED' REQUIREMENT	EXAMPLE OF A POLISH CLAUSE	ENGLISH EQUIVALENT
noun phrase, accusative (B)	poinformowano mnie	I have been informed
noun phrase, dative (C)	powiedziano mi	I have been told
noun phrase, dative (C)	podobasz mi siê	I like you

3.5. Techniques to improve efficiency of the algorithm

System POLENG aims at translation in near-real time. Several techniques have been used in order to achieve this aim. They were reported in more detail in [3]. Here, a short summary of the techniques is presented.

3.5.1. Descending order of rules

The rules of replacing one non-terminal symbol should be set in the order 'from the longest to the shortest' (the formal definition would require a little more sophisticated approach).

Suppose that following rules are given in the grammar assumed in the algorithm:

$\mathbf{B \rightarrow C_1}$
$\mathbf{B \rightarrow C_1C_2}$
$\mathbf{A \rightarrow BD}$

Setting the order of rules for replacing 'B' in the descending order, i.e.:

$\mathbf{B \rightarrow C_1C_2}$
$\mathbf{B \rightarrow C_1}$

enables to block backtracking in the third rule, which in Arity Prolog formalism is expressed in the following way:

$\mathbf{A \rightarrow [!\ B\ !]D}$

3.5.2. Using alternatives

Replacing a few rules with one rule including alternatives may improve efficiency. Replacing two rules:

A → BD
A → CD.

by the rule:

A → (B;C)D (where ';' denotes alternative)

will not improve efficiency. Using alternatives for right-hand symbols which do not appear in the first position decreases complexity of the algorithm. Replacing the rules:

A → BC
A → BD

by the rule:

A → B(C;D)

results in not repeating the same operations if B fails.

3.5.3. Replacing non-terminal symbols with parameters.

Suppose that the following productions are given in the grammar:

A→CD
B→CE

Creating a new grammar decreases complexity of parsing. To obtain the new grammar:

- remove symbols A, B from the set of non-terminals
- add a new non-terminal F
- replace two rules of the above shape by one rule with an additional parameter:

F(P) → C(D and P=P_1 ; E and P=P_2)
Thanks to such modifications parsing the symbol 'C' is executed once and not twice.

3.5.4. Filters (negative rules)

It is sometimes easy to anticipate that a certain hypothesis on a structure of a sentence or of its part must fail. Negative rules are rules which are verified in order to eliminate some hypotheses 'beforehand'.

Negative rules play a role of a set of filters or a sieve which lets through only rules which may succeed.

Example 19

Below some of the filters for a noun phrase are listed (Arity Prolog symbols):

noun_phr(P,R,G,L,O,T,[],_) :- !,fail.
(a noun phrase cannot be an empty list)
noun_phr(P,R,G,L,O,T) → (conj_by; conj_¿e; conj_left; conj_right;),
anything, !, fail.
(a noun phrase cannot start with some kinds of conjunctivas)
noun_phr(P,R,G,L,O,T) → noun1(P1,R1,G1,L1,T1),
{R \=R1, !, fail}.
(a noun phrase of a given gender cannot start with a noun of a different gender)

4.TESTING THE SYSTEM

The system was tested in two aspects:

1) Is the program able to cope with all possible sets of requirements (given in the dictionary) for a given verb? This type of testing seems crucial because the algorithm of parsing verbal groups is the most complex part of the translation algorithm.

2) Is the program able to cope with a piece of text which has not been prepared for testing?

The testing session is presented in the appendices The first part of the session consisted in translating some simple sentences prepared to be used in verifying the requirements of two interesting verbs: ‘mówiæ’ and ‘t³umaczyæ’. This part is printed in Appendix 1. The second part is a translation of the original abstracts of papers [3] and [4]. This is given in Appendix 2.

The screen output was re-directed to the file ‘sesja.txt’. The appendices include the contents of the file. The only adjustments concern fonts. The parameter representing the English phrasal structure tree may look a little strange. Some pieces of translation may seem not perfect. However, the session print has not been corrected for the sake of authenticity. Imperfect translations are discussed in footnotes.

APPENDIX 1. THE PRINT OF THE SESSION VERIFYING IMPLEMENTATION OF VERB REQUIREMENTS

Twój student lubi mówiæ.

Your student likes speaking.

Parse Tree:
elem_sent(no_num(adj_gr(your,student)),kcz_wym_bezok(present_simp_o3(like,tak),ing(tak,[speak])))

Wczoraj mówi³ o automatycznym t³umaczeniu tekstu.

He spoke about the machine text translation yesterday.

Parse Tree:
elem_sent(he,kcz([spoke,prep(about,no_num(frd([machine,text,translation]))),yesterday]))

Czy on umie mówiæ inteligentnie?

Can he speak intelligently?

Parse Tree:
czy(inv(elem_sent(no_num(pron([he])),kcz_wym_bezok(present_simp_o3_full(can,tak),inf(tak,
[speak,intelligently])))))

Nie wiem ale o in¿ynierii jêzyka naturalnego mówi³ ciekawie.

I don’t know but he spoke absorbingly about the natural language processing.

Parse Tree:
but_phr(elem_sent(I,kcz([don_t,know])),elem_sent(he,kcz([spoke,adv(absorbingly),prep(about,no_num(frd
([natural,language,processing]))]))))

Czy administrator sieci mówi³ ci?

Did the administrator of the network tell you?

Parse Tree:
czy(inv(elem_sent(no_num(genitive_of(administrator,no_num(network))),kcz([did,tell,prep(no,no_num(pron(
[you]))]))))

Mówi³ mi, ale nie wiem o czym mi mówi³.

He told me but I don’t know what he told me about.

Parse Tree:
but_phr(elem_sent(he,kcz([told,prep(no,no_num(pron([me])))])),elem_sent(I,kcz([don_t,know,que(what
(nwym(he,kcznwym([told,prep(no,no_num(pron([me])))]),prep(about))))])))

Nie mówi³ ci o b³êdach twojego programu?

Didn’t he tell you about the errors of your program?[1]

Parse Tree:
czy(inv(elem_sent(he,kcz([didn_t,tell,prep(no,no_num(pron([you]))),prep(about,no_num(genitive_of(errors,
no_num(adj_gr(your,program)))))])))

O tym mi nie mówi³.

He didn’t tell me about it.

Parse Tree:
elem_sent(he,kcz([didn’t,tell,prep(no,no_num(pron([me]))),prep(about,no_num(pron([it])))]))
---

[1] Correct translation: ‘Didn’t he tell you about the errors **in** your program?’

Zawsze mówi³em ci ¿e on ciê lubi.

I have always told you that he likes you.

Parse Tree:
elem_sent(I,szfcz(kcz([have,always,told,prep(no,no_num(pron([you]))))]),tha(elem_sent(no_num(pron([he])),
kcz([likes,prep(no,no_num(pron([you]))))])))))

Wczoraj mówi³eœ ¿e mnie nie lubi.

You said he didn't like me yesterday.[2]

Parse Tree:
elem_sent(you,kcz([said,th0(elem_sent(he,kcz([didn_t,like,prep(no,no_num(pron([me]))))])))),yesterday]))

Nie s³ysza³eœ jak mówiê do ciebie.

You didn't hear me talking to you.

Parse Tree:
elem_sent(you,kcz([didn_t,hear,sng(tak,jak(me,kcz([talk,prep(to,no_num(pron([you]))))])))]))

O czym do mnie mówi³eœ?

What did you talk to me about?

Parse Tree:
what(inv(nwym(you,kcznwym([did,talk,prep(to,no_num(pron([me]))))]),prep(about))))

Mówi³em ¿e on bêdzie do ciebie mówi³ o b³êdach programu, mimo ¿e ciê lubi.

I said he would talk to you about the errors of the program[1] although he likes you .

Parse Tree:
although2(elem_sent(I,kcz([said,th0(elem_sent(no_num(pron([he])),kcz([would,talk,prep(to,no_num(pron(
[you]))),prep(about,no_num(genitive_of(errors,no_num(program))))])))]),elem_sent(he,kcz([likes,prep(no,
no_num(pron([you]))))])))

Czy on bêdzie chcia³ ze mn¹ mówiæ jutro?

Will he want to talk to me tomorrow?

Parse Tree:
czy(inv(elem_sent(no_num(pron([he])),kcz_wym_bezok(future_simp_o2(want,tak),to(tak,[talk,to,me,
tomorrow])))))

Nie wiem bo o tym ze mn¹ nie mówi³.

---

[2] Automatic transformation of time adverbs caused ambiguity here.

I don’t know because he didn’t talk to me about it.

Parse Tree:
because(elem_sent(I,kcz([don_t,know])),elem_sent(he,kcz([didn_t,talk,prep(to,no_num(pron([me]))),
prep(about,no_num(pron([it])))])))

Czy ty coœ mówisz?

Are you saying something?

Parse Tree:
czy(inv(elem_sent(no_num(pron([you])),kcz([are,saying,prep(no,no_num(smth_or_smone([something])))])))

Mówiê ¿e oni coœ mówi[1].

I say they are saying something.

Parse Tree:
elem_sent(I,kcz([say,th0(elem_sent(no_num(pron([they])),kcz([are,saying,prep(no,no_num(smth_or_smone(
[something])))]))]))

On t[3]umaczy[3], gdzie kupi[3] programy.

He explained where he had bought the programs.

Parse Tree:
elem_sent(no_num(pron([he])),szfcz(kcz([explained]),zaimpspyt([where],noinv(elem_sent(he,kcz([had,
bought,prep(no,no_num(programs))])))))

On próbowa[3] ciê t[3]umaczyæ ale nie wierzyliœmy mu.

He tried to justify you but we didn’t believe him.

Parse Tree:
but_phr(elem_sent(no_num(pron([he])),kcz_wym_bezok(past_simp(tried,tak),to(tak,[justify,you]))),
elem_sent(we,kcz([didn_t,believe,prep(no,no_num(pron([him])))])))

Jeœli chcesz t[3]umaczyæ aproksymacjê funkcji to musisz nauczyæ siê mówiæ wolno.

If you want to explain the approximation of the function, you must learn to speak slowly.

Parse Tree:
if_then(elem_sent(you,kcz_wym_bezok(present_simp_o1(want,tak),to(tak,[explain,the,approximation,of,the,
function]))),elem_sent(you,kcz_wym_bezok(present_simp_o1(must,tak),inf(tak,[learn,to,speak,slowly]))))

Bêdê mówi[3] wolno gdy¿ bêdê t[3]umaczy[3] aproksymacjê tobie.

I shall speak slowly because I shall explain the approximation to you.

Parse Tree:
because(elem_sent(I,kcz([shall,speak,adv(slowly)])),elem_sent(I,kcz([shall,explain,prep(no,no_num(
approximation)),prep(to,no_num(pron([you])))])))

Mnie nie musisz t³umaczyæ.

You don't have to explain to me[3].

Parse Tree:
elem_sent(you,kcz_wym_bezok(present_simp_o1(must,nie),inf(tak,[explain,to,me])))

Nie wiem, czy wiesz, ¿e bêdê t³umaczy³ z jêzyka angielskiego.

I don't know if you know that I shall translate from English.

Parse Tree:
elem_sent(I,kcz([don_t,know,que(czy(noinv(elem_sent(you,szfcz(kcz([know]),tha(elem_sent(I,kcz([shall,
translate,prep(from,no_num(frd([English])))])))))))]))

Czy bêdziesz t³umaczy³ na jêzyk polski?

Will you translate into Polish?

Parse Tree:
czy(inv(elem_sent(you,kcz([will,translate,prep(into,no_num(frd([Polish])))]))))

Chcesz ¿ebym t³umaczy³ na jêzyk niemiecki z jêzyka angielskiego?

Do you want me to translate from English into German?

Parse Tree:
czy(inv(elem_sent(you,kcz([do,want,sto(tak,by2(me,kcz([translate,prep(from,no_num(frd([English]))),prep(
into,no_num(frd([German])))])))])))))
Na jakie jêzyki umiesz t³umaczyæ?

What languages can you translate into?

Parse Tree:
pytzaimpm(zaimpmpyt([what],no_num(languages)),inv(nwym(you,kcznwym(kcz_wym_bezok(
present_simp_o1_full(can,tak),inf(tak,[translate])),prep(into))))

Ja umiem t³umaczyæ mowê z jêzyka angielskiego ale ten system t³umaczy tekst z jêzyka polskiego na jêzyk angielski.

I can translate the speech from English but this system is translating the text from Polish into English[4].

Parse Tree:
but_phr(elem_sent(no_num(pron([I])),kcz_wym_bezok(present_simp_o1(can,tak),inf(tak,[translate,the,speech,
from,English]))),elem_sent(no_num(adj_gr(this,system)),kcz([is,translating,prep(no,no_num(text)),prep(
from,no_num(frd([Polish]))),prep(into,no_num(frd([English])))])))

Czy ten system mo¿e t³umaczyæ mowê na tekst?

---

[3] The correct English translation would require an object (e.g. You don't have to explain anything to me)
[4] The algorithm assumes noun groups to be definite.

Can this system translate the speech into the text?[4]

Parse Tree:
czy(inv(elem_sent(no_num(adj_gr(this,system)),kcz_wym_bezok(present_simp_o3_full(can,tak),inf(tak,
[translate,the,speech,into,the,text])))))

Mówi³em ci ¿e ten system t³umaczy tekst na tekst.

I told you that this system translated the text into the text[4] .

Parse Tree:
elem_sent(I,szfcz(kcz([told,prep(no,no_num(pron([you])))]),tha(elem_sent(no_num(adj_gr(this,system)),kcz(
[translated,prep(no,no_num(text)),prep(into,no_num(text))])))))

Nie musisz siê t³umaczyæ.

You don't have to excuse yourself.

Parse Tree:
elem_sent(you,kcz_wym_bezok(present_simp_o1(must,nie),inf(tak,[excuse,yourself])))

Czy ja siê t³umaczê?

Am I excusing myself?

Parse Tree:
czy(inv(elem_sent(no_num(pron([I])),kcz([am,excusing,myself]))))

APPENDIX 2. PRINT OF THE SESSION VERIFYING TRANSLATION OF PAPER ABSTRACTS

(Entire sentences from original texts are bolded).

Realizacja.

The realisation.

Parse Tree:
no_num(realization)

Realizacja elektronicznego s³ownika polsko-angielskiego.

The realisation of the electronic Polish-to-English dictionary.

Parse Tree:
no_num(genitive_of(realisation,no_num(adj_gr(electronic,'Polish-to-English',dictionary))))

Realizacja elektronicznego s³ownika polsko-angielskiego skonstruowanego pod k¹tem implementacji[5] w automatycznym t³umaczeniu tekstu w formacie zmodyfikowanego automatu skoñczonego.

The realisation of the electronic Polish-to-English dictionary designed in the point of view of the implementation in the machine text translation in the format of the modified finite automaton.

Parse Tree:
no_num(genitive_of(realisation,part(no_num(adj_gr(electronic,'Polish-to-English',dictionary)),
kcznwym([designed,prep([in,the,point,of,view,of],no_num(implementation)),prep([in],no_num(frd([machine,
text,translation]))),prep([in],no_num(genitive_of(format,no_num(adj_gr(modified,finite,automaton)))))])))

W poni¿szej pracy przedstawiona jest realizacja elektronicznego s³ownika polsko-angielskiego skonstruowanego pod k¹tem implementacji w automatycznym t³umaczeniu tekstu w formacie zmodyfikowanego automatu skoñczonego.

In the below paper the realisation of the electronic Polish-to-English dictionary designed in the point of view of the implementation in the machine text translation in the format of the modified finite automaton is presented.

Parse Tree:
elem_sent(prep([in],no_num(adj_gr(below, paper))),
no_num(genitive_of(realisation,part(no_num(adj_gr(electronic,polish-to-english,dictionary)),
kcznwym([designed,prep([in,the,point,of,view,of],no_num(implementation)),prep([in],no_num(frd([machine,
text,translation]))),prep([in],no_num(genitive_of(format,no_num(adj_gr(modified,finite,automaton)))))])))),
kcz_bierna(be(is,tak),kcznwym([presented])))

Struktura logiczna automatu reprezentuj¹cego s³ownik pojedynczych wyrazów.

The logical structure of the automaton representing the dictionary of the single words.

Parse Tree:
no_num(genitive_of(adj_gr(logical,structure),part(no_num(automaton),kcz([representing,prep(no,no_num
(genitive_of(dictionary,no_num(adj_gr(single,words)))))])))

Struktura logiczna automatu reprezentuj¹cego s³ownik pojedynczych wyrazów oraz struktura logiczna automatu odpowiadaj¹cego s³ownikowi fraz wyrazowych.

The logical structure of the automaton representing the dictionary of the single words and the logical structure of the automaton corresponding to the dictionary of the word phrases.

Parse Tree:
and_phr(no_num(genitive_of(adj_gr(logical,structure),part(no_num(automaton),kcz([representing,prep(no,
no_num(genitive_of(dictionary,no_num(adj_gr(single,words)))))])))),no_num(genitive_of(adj_gr(logical,
structure),part(no_num(automaton),kcz([corresponding,prep(to,no_num(genitive_of(dictionary,no_num
(adj_gr(word,phrases)))))])))))

Zobrazowane s¹ struktura logiczna automatu reprezentuj¹cego s³ownik pojedynczych wyrazów oraz struktura logiczna automatu odpowiadaj¹cego s³ownikowi fraz wyrazowych.

The logical structure of the automaton representing the dictionary of the single words and the logical structure of the automaton corresponding to the dictionary of the word phrases are described.

Parse Tree:
elem_sent(and_phr(no_num(genitive_of(adj_gr(logical,structure),part(no_num(automaton),kcz([representing,
prep(no,no_num(genitive_of(dictionary,no_num(adj_gr(single,words)))))])))),no_num(genitive_of(adj_gr

---

[5] 'More perfect' translation of the phrase 'pod k¹tem implementacji...' would be 'with view to implement...'

(logical,structure),part(no_num(automaton),kcz([corresponding,prep(to,no_num(genitive_of(dictionary,
no_num(adj_gr(word,phrases)))))]))))),kcz_bierna(be(are,tak),kcznwym([described])))

Przechowywania s³ownika-automatu w pliku binarnym na noœniku magnetycznym.

Storing the dictionary-automaton in the binary file on the magnetic carrier.

Parse Tree:
no_num(kcz([storing,prep(no,no_num(dictionary-automaton)),prep([in],no_num(adj_gr(binary,file))),
prep([on],no_num(adj_gr(magnetic,carrier)))]))

Zaprezentowano metodê przechowywania s³ownika-automatu w pliku binarnym na noœniku magnetycznym.

The method of storing the dictionary-automaton in the binary file on the magnetic carrier was presented.

Parse Tree:
kcz_bezosobowa(no_num(genitive_of(method,no_num(kcz([storing,prep(no,no_num(dictionary-automaton)),
prep([in],no_num(adj_gr(binary,file))),prep([on],no_num(adj_gr(magnetic,carrier)))])))),be(was,tak),
kczwwym([presented]))
---

algorytm odszukiwania has³a (wyrazu lub frazy).

The algorithm of searching the entry (the word or the phrase).

Parse Tree:
no_num(genitive_of(algorithm,no_num(kcz([searching,prep(no,braces(no_num(entry),or_phr(no_num(word),
no_num(phrase))))])))

Wydzielania informacji w tak przetworzonym s³owniku.

Extracting the information in the dictionary thus processed.

Parse Tree:
no_num(kcz([extracting,prep(no,no_num(information)),prep([in],no_num(thus_phr(dictionary,processed)))]))

Podano algorytm odszukiwania has³a (wyrazu lub frazy) i wydzielania informacji w tak przetworzonym s³owniku.

The algorithm of searching the entry (the word or the phrase) and extracting the information in the dictionary thus processed was given.

Parse Tree:
kcz_bezosobowa(no_num(genitive_of(algorithm,and_phr(no_num(kcz([searching,prep(no,braces(no_num
(entry),or_phr(no_num(word),no_num(phrase))))])),no_num(kcz([extracting,prep(no,no_num(information)),
prep([in],no_num(thus_phr(dictionary,processed)))])))))),be(was,tak),kczwwym([given]))

W poni¿szej pracy przedstawiona zostanie konstrukcja dwujêzycznego s³ownika elektronicznego.

In the below paper the design of the bilingual electronic dictionary will be presented.

Parse Tree:
elem_sent(prep([in],no_num(adj_gr(below,paper))),no_num(genitive_of(design,no_num

(adj_gr(bilingual,electronic,dictionary)))),kcz_bierna(be([will,be],tak),kcznwym([presented])))

W poni¿szej pracy przedstawiona zostanie konstrukcja dwujêzycznego s³ownika elektronicznego skonstruowanego w celu implementacji w komputerowym t³umaczeniu zdañ jêzyka polskiego na jêzyk angielski.

In the below paper the design of the bilingual electronic dictionary designed in the purpose of the implementation in the computerised translation of the sentences of Polish into English will be presented.

Parse Tree:
elem_sent(prep([in],no_num(adj_gr(below,paper))),no_num(genitive_of(design,part
(no_num(adj_gr(bilingual,electronic,dictionary)),kcznwym([designed,prep([in,the,purpose,of],
no_num(implementation)),prep([in],no_num(adj_gr(computerised,kcz([translation,of,prep(no,no_num
(genitive_of(sentences,no_num(frd([Polish])))))),prep(into,no_num(frd([English])))])))])))),kcz_bierna
(be([will,be],tak),kcznwym([presented])))

Zaprezentowany zostanie algorytm translacji w oparciu o ten s³ownik.

The algorithm of the translation on the basis of this dictionary will be presented.

Parse Tree:
elem_sent(no_num(genitive_of(algorithm,prep(no_num(translation),[on,the,basis,of],no_num(adj_gr(this,
dictionary))))),kcz_bierna(be([will,be],tak),kcznwym([presented])))

Wskazane zostan¹ powi¹zania miêdzy budow¹ s³ownika a algorytmem translacji.

The interactions between the structure of the dictionary and the algorithm of the translation will be pointed out.

Parse Tree:
elem_sent(prep(no_num(interactions),prep([between],no_num(genitive_of(structure,no_num(dictionary))),
[and],no_num(genitive_of(algorithm,no_num(translation))))),kcz_bierna(be([will,be],tak),kcznwym
([pointed,out])))

Prace nad s³ownikami elektronicznymi oraz systemami in¿ynierii jêzyka naturalnego.

The works[6] on the electronic dictionaries and the NLP systems.

Parse Tree:
prep(no_num(works),[on],and_phr(no_num(adj_gr(electronic,dictionaries)),no_num(frd([NLP,systems]))))

Prace nad s³ownikami elektronicznymi oraz systemami in¿ynierii jêzyka naturalnego powinny byæ œciœle ze sob¹ skoordynowane.

The works[6] on the electronic dictionaries and the NLP systems should be co-ordinated strictly with each other.

Parse Tree:
elem_sent(prep(no_num(works),[on],and_phr(no_num(adj_gr(electronic,dictionaries)),no_num(frd([NLP,
systems])))),kcz_wym_bezok(should,inf(tak,[be,co-ordinated,strictly,with,each,other])))

---

[6] The correct translation should transfer the Polish plural form ‘prace’ into the English singular form ‘work’. The algorithm is able to cope with such a difficulty only for nouns which are marked in the dictionary as not having plural forms (e.g. information). The noun ‘work’ is not marked as a noun of this type.

Wysuniêta zostanie hipoteza ¿e prace nad s³ownikami elektronicznymi oraz systemami in¿ynierii jêzyka naturalnego powinny byæ œciœle ze sob¹ skoordynowane.

The hypothesis will be advanced that the works[7] on the electronic dictionaries and the NLP systems should be co-ordinated strictly with each other.

Parse Tree:
elem_sent(no_num(hypothesis),kcz_bierna(be([will,be],tak),kcznwym([advanced])),elem_sent(prep(no_num(works),[on],and_phr(no_num(adj_gr(electronic,dictionaries)),no_num(frd([NLP,systems])))),
kcz_wym_bezok(should,inf(tak,[be,co-ordinated,strictly,with,each,other]))))

LITERATURE

[1] Aho. A. V., Ullman J.D. *The theory of parsing, translation and compiling*, vol. 1. *Parsing,* Prentice Hall,1972

[2] Jassem K. *Classification of Polish nouns for the flexional electronic dictionary,* to appear in: "Sprache und Datenverarbeitung. International Journal for Language Data Processing".

[3]. Jassem K. *Zale¿noœci miêdzy s³ownikiem elektronicznym i regu³ami translacji w t³umaczeniu automatycznym na przyk³adzie informatycznego systemu t³umaczenia zdañ polskich na jêzyk angielski* ("Interactions Between an Electronic Dictionary and Computerized Translation Rules in an Informatic System of Polish-to-English Text Translation"), in: C. Basztura, K. Dobrogowska (ed.): "Podstawowe problemy komputerowego t³umaczenia ró¿nojêzycznego dialogu w czasie rzeczywistym", Poznañ 1996.

[4]. Jassem K., Lison M., M¹czyñski R. *Implementacja elektronicznego s³ownika dwujêzycznego skonstruowanego dla potrzeb automatycznego t³umaczenia tekstu* ("The Implementation of a Bilingual Electronic Dictionary Designed for Computer Text Translation"), in: C.Basztura, K. Dobrogowska (ed.): "Podstawowe problemy komputerowego t³umaczenia ró¿nojêzycznego dialogu w czasie rzeczywistym", Poznañ 1996.

[5] Jassem K., Lison M., M¹czyñski R. *The implementation of a bilingual electronic dictionary amenable to on-line modification,* in „Speech and Language Technology", Wroc³aw 1997.

[6] Szpakowicz S. *Formalny opis sk³adniowy zdañ polskich*, Warszawa 1986

[7] Vetulani Z., Jassem K.: *Coding of nouns in the morphologic electronic dictionary of Polish - POLEX*, in: *Proceedings of the 29. Linguistic Colloquium, Aarhuus 1994.*