Lexical Functions and Homonymy

Sasha Andreyeva

Department of Theoretical and Applied Linguistics, Philological Faculty –
Moscow State University
Moscow, Vorobjevi Gori, 1st Building of the Humanities
andreyevs@mtu-net.ru

Abstract

The paper deals with the homonymy of lexical-functional constructions, i.e. word-combinations consisting of an argument of a lexical function and a value of this lexical function for this argument. The paper describes different types of homonymy of lexical-functional constructions and proposes the ways of their processing. The processing decisions are made for the system of machine translation ETAP-3.

Keywords

Lexical Functions, Natural Language Processing, Machine Translation.

1. Introduction

The concept of lexical function (LF) was proposed in Igor Mel’čuk's “Meaning ⇔ Text Theory” (Mel’čuk, 1974, Mel’čuk et al., 1984) as the means of description of certain types of lexeme meaning correlations. According to (Mel’čuk, 1974, 78), “Lexical function f describes the dependence that determines for the certain word or word-combination such a multitude of words or word-combinations \( \{Y_i\}=f(X) \), that for every \( X_1, X_2 \) the following statement is true: if \( f(X_1) \) and \( f(X_2) \) exist, then there is always the same semantic correlation between \( f(X_1) \) and \( X_1 \), on the one hand, and between \( f(X_2) \) and \( X_2 \), on the other hand”.

Soon lexical-functional description turned out to be of great value for the systems of natural language processing. Different ways the LF description can be used in NLP system are described in (Apresjan et al., 2003, 279-288). As far as machine translation is concerned, lexical functions can be used, in particular, to resolve syntactic and lexical ambiguity and to provide translation equivalents. In the system of machine translation ETAP-3 there are special mechanisms enabling the use of LFs for these purposes. But in real processing of homonymous LF constructions (i.e. word-combinations consisting of an argument of a lexical function and a value of this lexical function for this argument) these mechanisms do not always work properly.
The goal of this paper is to show how the homonymy interacts with lexical-functional description and to propose the mechanisms for the text processing in cases of such an interaction.

There exists a certain number of papers describing the correlation of lexical functions and homonymy (for example, Apresjan et al., 2003, 279-288). But in these papers lexical functions are used as a disambiguation tool in parsing. I.e. the papers deal with non-homonymous LF constructions consisting of words having homonyms, and the LF link between the words in these word-combinations is used to choose the right homonyms of these words. For example, in (1) both words are homonymous, but the whole phrase has only one meaning.

(1)  CausFunc0 (argument) = cause  \(\rightarrow\) cause an argument

This paper deals with two general types of homonymy in connection with lexical functions:

1. The cases when a word-combination has several meanings, and at least in one of these meanings it is a lexical-functional construction.

2. The cases when a word (or a group of words) is a value of two or more lexical functions for the same argument.

(2)  a) IncepOper1 (disease) = develop  \(\rightarrow\) Mary developed this disease.
    b) IncepFunc0 (disease) = develop  \(\rightarrow\) The disease developed quickly.
    c) Func0 (disease) = develop  \(\rightarrow\) The disease developed quickly.

The first type of homonymy is illustrated in (2b) and (2c) – the word-combination disease developed has two meanings, and at least one of them is lexical-functional (in this particular case both meanings are lexical-functional). The second type of homonymy is illustrated in all the examples from the group (2), especially with (2a) in comparison with the other examples from this group.

The paper is based on the vocabulary material used in the system of Russian-English and English-Russian machine translation ETAP-3. Thus, the examples in the paper are taken from the Russian and the English languages.

The constructions we deal with can be divided into the following classes:

0. False homonymy.

1. Homonymy of values of lexical functions - the cases when a word (or a group of words) is a value of several lexical functions for the same argument, but the word-combinations formed by this argument and these values are not homonymous (as in (2a) and (2b)).

2. Homonymy of word-combinations – the cases when a word-combination has several meanings, and at least in one of these meanings it is a lexical-functional construction.
2. Types of Homonymy

2.0. False Homonymy (the Problem of Definition)

This section of the paper is given number 2.0 not by mistake or by chance. It describes word-combinations that resemble the homonymy of (2b) – (2c) but that are not really homonymous.

(3) *to come to / to conclude an agreement*

(3) is a lexical-functional construction. Trying to determine which LF it represents we find two suitable candidates: IncepOper1 and CausFunc0.

At first sight (3) is the same as (2b) and (2c). In both cases there is a word-combination consisting of an argument of a lexical function and its value, and the value turns out to be the value of two different LFs.

But there is an important difference between (2) and (3). (2b) and (2c) have different meanings. (2b) interpreted in IncepFunc0 terms means that the moment somebody fell ill came quickly after some other event, (2c) interpreted in Func0 terms means that after the person had fallen ill his disease went quickly through all of its stages.

(3) has only one meaning irrespective of the function used to describe it. “To begin to have an agreement” (IncepOper1) or “to cause an agreement to take place” (CausFunc0) are two descriptions of the same situation, not descriptions of two different situations.

(3) is not unique. There are quite large groups of word-combinations that, not being homonymous, can be described with the help of two lexical functions. For example, there are a lot of words being arguments of LF IncepOper1 with the value *begin* (*to begin an argument, a battle, a struggle* and so on). All these words can be described as arguments of the LF CausFunc0 with the same value.

There exists a much larger group of words denoting different objects that can be created this or that way. All these words are arguments of LF CausFunc0 with the corresponding values (*to grow plants, to write music* etc). All these words can also be described as arguments of the LF Oper1 with the same values.

The possibility of non-homonymous word-combinations to be described with the help of two lexical functions can be called false homonymy. There are several reasons for its existence.

First, the descriptions of lexical functions are quite general and approximate. When the system of lexical function was created, the authors had no aim to divide all the possible situations into non-crossing classes, the aim was to describe the main prototypical semantic correspondences.

Second, the notion of lexical function was created for the description of situations. Being used for the description of objects, it produced additional cases of false homonymy of lexical functions.
Each lexical function has its meaning. As indicated above, there are big groups of words for which two of these meanings turn out to be the same. It proves that there are language domains where these pairs of meanings unite. To sum it up, these pairs of meanings are:

- IncepOper1 (X) – CausFunc0 (X)
- Oper1 (X) – CausFunc0 (X)
- FinOper1 (X) – LiquFunc0 (X) *(to stop the battle, for example)*

As far as processing of the described word-combinations is concerned, the main and the most important question that must be answered is with how many lexical functions we are going to describe these constructions. To my mind, the answer must be “one”. It seems that the description with the help of both LFs will not improve the system but will only make it more complicated.

The second question is which function must be chosen. This question must be answered separately for specific arguments or groups of arguments and is of no great importance for the problems under discussion.

### 2.1. Homonymy of Values of Lexical Functions

This section is devoted to the cases when a word (or a group of words) is a value of several lexical functions for the same argument, but the word-combinations formed by this argument and this value are not homonymous.

(4)  
\[
\begin{align*}
\text{a) Func0 (carrot) = grow} & \quad \text{Carrot grows.} \\
\text{b) CausFunc0 (carrot) = grow} & \quad \text{Pete grows carrot.}
\end{align*}
\]

The LF constructions in (4) are not homonymous, but one word is the value of two lexical functions for the same argument. In the ETAP-3 system the values of lexical functions are written in the dictionary entry of the argument. Thus, the dictionary entry of the word *carrot* has the following fragment:

- Func0: grow
- CausFunc0: grow

The processing problem of constructions of this kind consists in the correct choice of the lexical function at the stage of analysis. This problem is already solved in ETAP-3 system. There are several rules that are responsible for the establishment of an LF link between an argument and a value of LF in the sentence. Every rule is responsible for a certain group of lexical functions with the same type of syntactic link between a value and an argument of LF. In (4a) the argument of the LF is the predicate of its value, and in (4b) it is the primary object. So, at the stage of analysis the information about the syntactic structure allows to establish the correct LF.

---

1 In the ETAP-3 system verbs that can be both causative (*Mary grows carrot*) and non-causative (*Carrot grows*) are listed as one vocabulary entry, i.e. as one lexeme.
2.2. Homonymy of Word-Combinations

This section deals with what can be called homonymy proper – word-combinations that have several meanings among which at least one represents a lexical-functional construction.

Homonymous word combinations, at least one meaning of which represents a lexical function, can be divided into two types:

1. Homonymy of LF constructions - cases when a word-combination has two or more homonyms, and all of them are LF constructions representing different LF.

2. Homonymy of LF constructions and free word-combinations - cases when a word-combination has two or more homonyms, among which at least one is a lexical-functional construction and at least one is a free word-combination.

2.2.1. Homonymy of LF constructions

This section of the paper deals with the homonymous word-combinations, every meaning of which consists of an argument of an LF and a value of this LF for this argument. (2b) and (2c) represent this type of homonymy.

The task of the NLP system while processing homonymous constructions of any kind is to disambiguate them, if possible. Below the homonymous LF constructions are listed and the algorithms of their disambiguation are described.

No possible algorithm

There are some constructions which did not lend themselves to be resolved algorithmically.

(5) break-up of the fight
   a) S0_IncepFunc0 (fight) = break-up
   b) S0_FinFunc0 (fight) = break-up
   c) S0_LiquFunc0 (fight) = break-up

Of course, in the majority of real contexts a human can identify the proper meaning of a word-combination from (5), but the possible algorithms of disambiguation include extralinguistic information that cannot be used in NLP system.

Syntactic algorithm

There is a group of homonymous LF constructions with two possible meanings, one of which represents LF IncepOper1, and the other represents LF CausOper1. In this pair of lexical functions the syntactic links between the value and the argument of LF differ: X is the primary object of the value of IncepOper1 and the secondary object of the value of CausOper1. This difference enables the creation of disambiguation algorithms based on the analysis of syntactic structure.

Though the principle of the algorithm is the same, the algorithms themselves are of course unique. They are described below.
engage + different types of confrontations, both physical and verbal
a) IncepOper1 (battle) = engage  
He engaged in the battle.

b) CausOper1 (battle) = engage  
He engaged John in the battle.

These are some of the words that can be arguments of LF in this construction: argument, battle, combat, debate, discussion, dispute, fight, game, struggle, war, play, confrontation.

The disambiguation algorithm for this construction is the following:

1. Step one. If the sentence includes “engage oneself in”, i.e. if the value of LF takes a reflexive pronoun as its primary object, the whole phrase represents IncepOper1, and the reflexive pronoun must be removed.

2. Step two. If the value of LF takes something except “in + the argument of the LF” as its primary object, the phrase represents CausOper1.

3. Step three. If the value of LF takes “in + the argument of the LF” as its primary object, the phrase represents IncepOper1.

engagement in + different types of confrontations, both physical and verbal
a) S0_IncepOper1 (combat) = engagement  
engagement in the combat

b) S0_CausOper1 (combat) = engagement  
engagement of smb in the combat

Unlike the previous example, in this case the algorithm cannot disambiguate all the contexts. If the value of LF takes “in + the argument of the LF” as its primary object, the word-combination represents IncepOper1, in other cases the disambiguation is impossible.

zarazhenije + bolezn’ (disease) / names of diseases
a) S0_IncepOper1 (bolezn’) = zarazhenije  
zarazhenije bolezn’ju development of a disease

b) S0_CausOper1 (bolezn’) = zarazhenije  
zarazhenije bolezn’ju infection with a disease

The algorithm in this case is the following: if the value of LF takes “ot + Gen” (from + noun) as its secondary object, the phrase represents IncepOper1, in other cases the disambiguation is impossible.

Non-syntactic algorithm

Along with the homonymous LF constructions representing lexical functions with different syntactic links between the value and the argument of LF there exist homonymous LF constructions with the same syntactic link between the value and the argument of LF. Syntactic disambiguation algorithms cannot be used for these constructions, so mechanisms of another kind are necessary.

vxodit’ v komitet
a) Oper1 (komitet) = vxodit’ v  
vxodit’ v komitet to be on the committee

b) IncepOper1 (komitet) = vxodit’ v  
vxodit’ v komitet to join the committee
The following arguments can be used in this construction: administration, coalition, committee, commission, government, group, society, union; team, troop, leaders.

In (9) the disambiguation tool is the aspect of the verb. If the aspect is perfective, it can be only IncepOper1. If the aspect is imperfective, the situation is not that simple. In the majority of cases and with the majority of arguments it can be only Oper1, but there exist constructions which can represent IncepOper1 in the imperfective aspect, that is why the imperfective characteristic itself cannot be used as a disambiguation tool. For example, in the following sentence the verb *vxodit’* is used in its imperfective form, but the whole word-combination represents LF IncepOper1.

(10) *Za poslednije 2 goda predstaviteli etoj partii trizhdy vxodili v komitet i trizhdy vyxodili iz nego.*

In the last 2 years the representatives of this party joined the cabinet three times and left it three times.

Disambiguation algorithms described in the section 2.2.1 cannot be implemented directly in the ETAP-3 system. The fact is that all these algorithms are “positive”: if the construction has certain characteristics, it represents this or that lexical function.

In the ETAP-3 the rules of identification of LF links between the words are general, and they cannot be changed for every concrete word-combination. I.e., if in the sentence there is a phrase *engage in the battle*, the system will establish both LF links between the words irrespective of the characteristics of the primary object.

To realize the above-described disambiguation algorithms in the system we must make them “negative”: first let the system establish all the possible LF links and then delete the links that are impossible. Thus, the algorithm for “engage in battle” will look the following way:

1. Step one. If the sentence includes “engage oneself in”, i.e. if the value of LF takes a reflexive pronoun as its primary object, delete the CausOper1 link and remove the reflexive pronoun from the syntactic structure.

2. Step two. If the value of LF takes something except “in + the argument of the LF” as its primary object, delete the IncepOper1 link.

3. Step three. If the value of LF takes “in + the argument of the LF” as its primary object, delete the CausOper1 link.

### 2.2.2. Homonymy of LF constructions and free word-combinations

This section describes the word-combinations having two or more meanings, among which at least one is a lexical-functional construction and at least one is a free word-combination. There exist two different types of such constructions.

---

2 To simplify the understanding of the paper Russian words are replaced with their English equivalents in this list.
1. The first type is the situation when the described word-combination either consists of non-homonymous words or includes the same homonyms of the words in all its meanings. In other words, if the described word-combination consists in all its meanings of the same lexemes, of the same vocabulary entries. This type of constructions is illustrated in (11).

(11) crisis breaks up
   a) IncepFunc0 (crisis) = break up  crisis breaks up
   b) non-LF meaning  crisis breaks up families

The verb *break up* is the value of LF IncepFunc0 for the argument *crisis* and this is the way it is listed in the dictionary. It means that if a word-combination *crisis breaks up* occurs in the text, it is analysed as an LF construction, and the LF link is established according to the general rule, which leads to the further processing of the verb *break up* as the value of LF. But in the word-combination *crisis breaks up families* the verb *break up* has its ordinary meaning that is not the value of LF. Thus, in order to provide proper processing of this phrase, we have to do something with this incorrect LF link.

As it was described above, this link cannot help being established, so our task is to get rid of it this or that way. There are two possibilities: to delete the link itself, using the same algorithm as described in the end of section 2.2.1 (create a rule: if in this word-combination the verb takes something as primary object, delete the LF link), or, if a system of machine translation is concerned, to translate the verb directly (to create a rule where the condition is the same and the action is “translate the verb”). The first decision seems preferable because it is more general and does not depend on the other language.

3. Mixed cases

In section 2 different types of homonymy of LF constructions were given. In section 3 some mixed cases of such homonymy are listed.

3.1. Mixed cases 2.1 and 2.2.2

This section describes the following case: a word (or a group of words) is a value of at least two LFs for the same argument, and the word-combinations formed by this argument and this value are not homonymous (2.1); at least one of these word-combinations is homonymous, and at least one of the homonyms is a free word-combination (2.2.2).

(12) break up the conflict – the conflict broke up
   a) IncepFunc0 (conflict) = break up  conflict broke up
   b) not LF, + direct object  conflict broke up the kingdom
   c) not LF, + into  conflict broke up into riots
   d) LiquFunc0 (conflict) = break up  somebody broke up the conflict
In this case the disambiguation process is the following: (12d) is separated easily (see section 2.1); in all the other cases conflict is a subject of the verb, and to disambiguate these cases the following algorithm is implemented: if the verb takes something as its direct object, delete the LF-link – example (12b) (for details, see section 2.2.2); in the construction “conflict broke up into + noun” the verb has a special meaning, so, if the verb takes something beginning with into as its primary object, replace the verb with a special translation equivalent – example (12c) (for details, see section 2.2.2); the remaining example (12a) represents the LF IncepFunc0.

3.2. Mixed cases 2.2.1 and 2.2.2

This section describes homonymous word-combinations having at least three meanings, among which at least two meanings represent lexical-functional constructions (2.2.1) and at least one represents a free-word combination (2.2.2).

(13) vxodit’ v kabinet

a) Oper1 (kabinet2) = vxodit’2 vxodit’2 v kabinet2 be in the cabinet
b) IncepOper1 (kabinet2) = vxodit’2 vxodit’2 v kabinet2 join the cabinet
c) not LF, kabinet1 vxodit’1 v kabinet1 enter the office

Homonymy that takes place in (13) is the worst type of homonymy connected with lexical functions. As it was mentioned in introduction, one of the uses of LF description is lexical ambiguity resolution. Its mechanism is the following: if a system establishes an LF link between two lexemes, it can ignore all the homonyms of these lexemes. For example, in (1) both words are homonymous, and after the LF link between cause1 (a verb) and argument2 (=discussion) is established, the system stops working with cause2 (a noun) and argument1 (=reason).

But this mechanism cannot and must not be used for the processing of (13). If we use it, we will never get the translation version “to enter the cabinet”, and it is the most common translation variant of this expression.

Neither can we by any means stop using the mechanism of lexical ambiguity resolution with the help of LFs because it is of great value for the system.

It seems that in this particular case the following decision can be proposed. “To enter the office” is definitely a free word-combination, but we can describe it with the help of a lexical function. It is possible to establish the following link: IncepOper1 (kabinet1) = vxodit’ v. In this case the status of (13a) (13b) and (13c) will become equal, and the system will process all of them.

This decision can be taken to improve the processing of the certain construction at the present moment, but in future it is to be replaced with a better and a more correct one.

---

3 Kabinet1 – office, kabinet2 – cabinet; vxodit’1 – to enter, vxodit’2 – to be a part of.
3.3. Mixed cases 2.1, 2.2.1, and 2.2.2

This section deals with the cases when all the types of homonymy described in the paper are illustrated with the help of one pair of words. One of these words is a value of at least three lexical functions for the same argument – the other word; at least two LF constructions formed with the help of these functions are homonymous and one is not; besides, one of these LF constructions is homonymous to a free word-combination.

(14) to break up the fight / struggle – the fight / struggle broke up
   a) IncepFunc0 (fight) = break up  \textit{fight broke up}
   b) FinFunc0 (fight) = break up  \textit{fight broke up}
   c) not LF, + direct object  \textit{fight broke up their romance}
   d) not LF, + into  \textit{fight broke up into a series of single combats}
   e) LiquFunc0 (fight) = break up  \textit{teacher broke up the fight}

In spite of the large number of meanings, the processing of (14) is relatively easy. The algorithms for all the cases except (14b) can be taken from the explanation for (12), and as for the disambiguation of cases (14a) and (14b), this mechanism cannot be created.

(14) is worth being mentioned separately. The verb used in it managed not only to be used in three LF constructions with the same argument, but to have there antonymous meanings.

4. Conclusion

The paper described different types of homonymy of lexical-functional constructions. It proposed a classification of these types and gave the examples of pure and mixed cases of homonymy. It also described possible decisions and algorithms of processing of the given examples.

Acknowledgement

I am grateful to L.L. Iomdin for comments on the prefinal version of this paper.

Bibliography

