# 5. Algorithm for processing the surface frames

In this chapter the automatic processing of the source data will be described. The format of the source data was described in Chapter 3. The desired content of the lexicon was described in Chapter 4. The steps which have to be done to achieve this are

1. identifying single frames
2. merging all variants of a single frame
3. marking the obligatority of frame members
4. assigning the functors to members
5. marking the possible diatheses

In the next sections these single steps will be described in detail.

## 5.1. Identifying and merging frames, marking the obligatority

In the source lexicon, every lemma is listed only once, even if it has several valency frames. A single valency frame, on the other hand, can have several variants (e.g. *učit koho co*$_{Acc}$, *učit koho čemu*$_{Dat}$—teach sb st). The variants of one frame are mixed with other frames and thus the first task is to separate the different frames and merge the variants. Let us show it with an example. The verb *bránit* has the following format in the source lexicon:

(128) `bránit <v>hTc3,sI,hPc3-sUeN,hPc3-hTc6r{v},` (protect, prevent)
`hPTc4,hPTc4-hPTc3r{proti},hPTc4-hPTc7r{před}`

Now, we arrange the members of all its frames into a table (see Table 5.1): the rows are single "frames" from the original dictionary and the columns are single members of the frames. If there are more than one + in a column, then two or more frames share

| | A<br>hTc3 | B<br>sI | C<br>hPc3 | D<br>sUeN | E<br>hTc6r{v} | F<br>hPTc4 | G<br>hPTc3r{proti} | H<br>hPTc7r{před} |
|---|---|---|---|---|---|---|---|---|
| 1 | + | | | | | | | |
| 2 | | + | | | | | | |
| 3 | | | + | + | | | | |
| 4 | | | + | | + | | | |
| 5 | | | | | | + | | |
| 6 | | | | | | + | + | |
| 7 | | | | | | + | | + |

Table 5.1.: Identifying single frames

that member. Now, we try to find maximal non-intersecting parts. In Table 5.1 they are marked by the gray background. These gray parts represent real frames. Their members which never occur in one frame together can be declared with high probability as variants of one member (in Table 5.1) we can see that items D and E are variants of one member and items G and H are variants of another member). Now, we can merge the variants, which is shown in Table 5.2: the frames 3 and 4 were merged into 3′ and the frames 5 and 6 into 6′.

| | A | B | C | D\|E | F | G\|H |
|---|---|---|---|---|---|---|
| 1 | + | | | | | |
| 2 | | + | | | | |
| 3′ | | | + | + | | |
| 5 | | | | | + | |
| 6′ | | | | | + | + |

Table 5.2.: Merging frame variants

There is a small problem with single-member frames (frames 1 and 2 in our example). They can be understood as separate frames, as in the case of *mířit kam* (head somewhere), *mířit na koho* (aim at sb), or as variants of one frame, as in the case of *bádat nad čím*, *bádat o čem* (research into st). We had to make a decision whether we wanted to merge all such frames, or whether we wanted to keep them separate. We decided to "merge as much as possible" because of an easier assignment of the functors, which will be explained in the next section. In our table, we then also merge the frames 1 and 2 into a frame with one member A|B.[1]

[1] A careful reader notices that the second frame should also contain Dative (hPc3) and it should in fact be merged with the third frame into one frame: `bránit [hPc3] [sI|sUeN]`. We showed here

In the above table we can also see how we identify obligatory members of a frame. In lines 5 and 6′, the member F is always present, while the other member G|H may be missing. Unfortunately, we are not able to say whether G|H is a general inner participant, or optional participant, or obligatory and deletable free modification, or even non-obligatory free modification, but at least the information about obligatory members of the frame should be correct. We use the square brackets for obligatory members of a frame (as was described in Chapter 4), and for now, we will use the parentheses for all other cases. The entry from example (128) now can be recorded as follows:

(129) a. `bránit [hTc3|sI]` (*bránit čemu/něco udělat*) (prevent st/doing st)

b. `bránit [hPc3] [sUeN]` (*bránit komu, aby něco neudělal*) (prevent sb from doing st)

c. `bránit [hPTc4] (hPTc3r{proti}|hPTc7r{před})` (*bránit koho/co {proti komu/čemu/před kým/čím}*) (protect sb/st {against sb/st|from sb/st})

As we said above, the source dictionary does not contain the so-called "left valency", i.e. subjects. This information is usually missing in printed dictionaries, as readers are able to fill the missing information, but in an electronic dictionary which is meant for language processing, such information must be included. We will describe the process of adding the subjects in the next section.

## 5.2. Assigning functors

It was shown by many authors that there is no straightforward correspondence between the deep frame and its surface realization. One can, however, try to find some regularities or tendencies, and then formulate rules for assigning the functors to the surface frames. The mappings between the tectogrammatical and morphemic levels (in active voice) is shown in Figure 5.1.

We can see that this picture does not help much—nearly everything is possible. It is necessary to add, however, that this picture also covers all marginal frames like líbit RSEs[i2]1(hPRc3)2[hPTc1]@ (like, appeal) and ubývat R--1[hTc2]@ (dwindle).[2]

Among all correspondences, there are some which are considered as typical. In the direction from the tectogrammatical level to the morphemic one these are Actor → Nominative, Patient → Accusative, Addressee → Dative, Effect → Instrumental, Origin →

a real example from the source lexicon, where some information was missing. The correction of this type of mistake is left for the post-editor.

[2] When we speak about marginal frames we do not say that the verbs with those frames are marginal, but the frames themselves are rather rare, and the lexicon contains only a few such frames. The verbs which have those frames may be in quite frequent use.

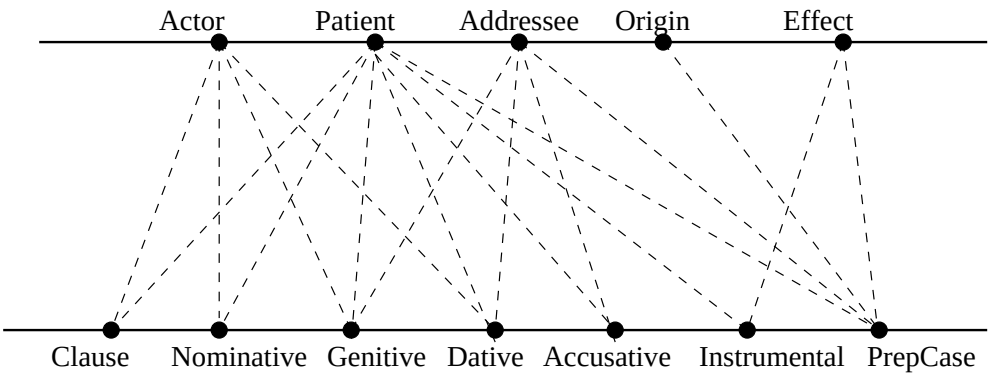


Figure 5.1.: Mapping between TL and ML in active voice

Genitive+Prep{*z*} (from) or Origin → Genitive+Prep{*od*} (from). In the opposite direction the correspondences are not so clear because of free modifications, which have a very broad repertory of the surface realizations. Thus Accusative can represent Patient or Temporal modification, Instrumental can represent Patient (*stát se*—become), Effect (*zvolit*—elect), Means (*zaplavit*—flood), Manner (*kopat*—dig); Genitive with the preposition *od* can represent Patient (e.g. *distancovat se*—dissociate), Origin (*dostat*—get), Direction from (*odejít*—leave), Temporal modification *how long* (*spát*–sleep), Cause (*opuchnout*—swell).

If we consider only frames with at least three actants[3] we get another picture shown in Figure 5.2.

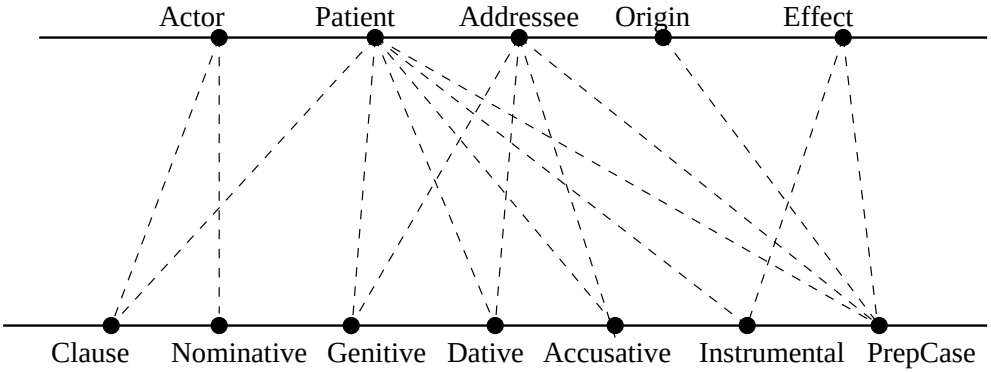


Figure 5.2.: Mapping between TL and ML for verbs with at least three actants

Though some joins disappeared, we still cannot find a unique mapping between the

[3]Frames with one or two actants are "uninteresting" as the functors are assigned after the rules listed in (6) in Chapter 2: if the frame has only one actant it is Actor, if there are two actants in the frame, they are Actor and Patient. In most cases, Actor is realized as Nominative and Patient as the "remaining" surface realization. There are some exceptional frames, as líbit RSEs[i2]1[hPRc3]2[hPTc1]@ (like, appeal) or zželet RSE1[hPc3]2[hPTRc2]@ (take pity on sb/st) which have to be edited manually.

| | Actor | Patient | Addressee | Origin | Effect | |
|---|---|---|---|---|---|---|
| *dát* | (Nom) | Acc | Dat | | | give |
| *dostat* | Nom | Acc | | <Gen+*od*> | | get |
| *šít* | (Nom) | (Acc) | <Dat> | <Gen+*z*> | | sew |
| *předělat* | (Nom) | Acc | <Dat> | <Gen+*z*> | <Acc+*na*> | remake |
| *žádat* | (Nom) | Acc | | (Gen+*od*) | | ask |

Table 5.3.: Prototypical frames

tectogrammatical and morphemic level. However, we can observe that frames can be split in two groups. The first group contains verbs whose actants are only realized by typical surface forms; we call these frames *prototypical* (several examples are listed in Table 5.3). The other group contains verbs with *non-prototypical* frames, where at least one member is realized by a non-typical surface form (examples are in Table 5.4). This observation was done by J. Panevová, and an experimental algorithm for assigning the functors to surface realizations was created (see Panevová and Skoumalová, 1992). The algorithm checks whether a frame contains only prototypical surface forms, and if so it assigns them the corresponding functors. In Table 5.4, we can see that there is a possible source of problems in frames with surface forms in Accusative and Dative—their functors can be assigned the other way round than we expect. In this case we have to add one more criterion, and it is that Addressee must be "more animate" than Patient.[4] From this reason we only assume *animate* Dative as the typical realization of Addressee (hPc3 or hPTc3).

In the experiment, it was supposed that we worked only with inner participants (free modifications were filtered out), which made the task easier. In BRIEF lexicon, however, we cannot rely on getting actants only in surface frames, but on the other hand, the repertory of free modifications occurring in the lexicon is not as wide as in

[4]The scale of animacy (in BRIEF notation) is hT < hPT < hP.

| | Actor | Patient | Addressee | Origin | Effect | |
|---|---|---|---|---|---|---|
| *zvolit* | (Nom) | Acc | | | Ins | elect |
| *hrozit* | (Nom) | Ins | (Dat) | | | threaten |
| *vystavit* | (Nom) | Dat | Acc | | | subject |
| *dědit* | (Nom) | (Acc) | | (Loc+*po*) | | inherit |
| *hovořit* | (Nom) | <Loc+*o*> | (Ins+*s*) | | | speak |
| *psát* | (Nom) | <Loc+*o*> | <Dat> | | (Acc) | write |
| *zeptat se* | Nom | Acc+*na* | (Gen) | | | ask |

Table 5.4.: Non-prototypical frames

the language as a whole (for example, a free modification of condition hardly occurs in a lexical entry). For this reason, we adopted a slightly different approach in the processing of BRIEF lexicon.

First, it was necessary to add the missing subjects. We did this automatically, and all frames got a subject in Nominative which was assigned the role of Actor: s[i1]1[hPTc1].[5]

The second step was assigning the roles to other members of the frame. Some preparation for this was done already while merging the frames: there is a list of possible functors for every surface realization, and this list was attached to every member of the original frame.[6] When we merged two members of a frame together we also made an intersection of the attached lists. An empty intersection prevented the two members from being merged. This process is shown in Table 5.5 on a frame of the verb *čertit se* (be angry). In BRIEF lexicon, the entry of this verb had the following form:

(130) `čertit se <v>hPTc4r{na},hTc4r{pro},hTc7r{nad},hTc3r{kvůli}`

| | hPTc4r{na} | hTc4r{pro} | hTc7r{nad} | hTc3r{kvůli} |
|---|---|---|---|---|
| (ACTANT) | + | + | + | |
| DIR.WHERE | + | | | |
| CAUSE | | + | + | + |
| PURPOSE | | | + | + |
| WHERE | | + | | |

Table 5.5.: Merging frame of the verb *čertit se* (be angry)

Every surface realization is assigned a list of functors, as shown in the table. However, the functor ACTANT which denotes any actant is only taken in consideration if the surface realization has no variants.[7] As we first try to merge all the prepositional cases into one member of the frame, we exclude ACTANT from the list. In the rest of the table, we can see that the first prepositional case (hPTc4r{na}) has an empty intersection of functors with other prepositional cases which means that it cannot be taken as their

[5] Some Czech verbs do not have a subject at all, e.g. *pršet* (rain), in some frames the subject is realized by a clause or by an infinitive, e.g. *znamenat* (mean), *zdát se* (seem), but the vast majority of Czech verbs have a nominal subject in Nominative. The exceptions will be treated by a post-editor, again.

[6] These lists were created manually. The original lexicon was first divided into classes of frames containing a certain surface realization. These classes were analyzed and the surface realization was assigned a list of functors. Similar lists were also created for the Prague Dependency Treebank (Hajičová et al., 2000). These lists are longer because they contain all functors found in texts, not only in a lexicon. Beside it, they also contain more prepositional cases than the BRIEF lexicon.

[7] We do not try to assign single inner participants (Actor, Patient, etc.) in this step, we only mark whether a certain surface form can possibly represent an inner participants. Because of technical reasons we mark all potential inner participants as Patients—in a case that that there is only one actant beside Actor we get Patient "for free". In a case that there are more actants further processing is necessary.

variant inside one member of a frame. The remaining surface realizations have a non empty intersection of functors containing the value CAUSE. In the resulting frame, the first prepositional case will be assigned the functors ACTANT and DIR.WHERE. Other prepositional cases will be merged into one frame member which will be assigned the functor CAUSE:[8]

(131) `čertit_se s[i1]1[hPTc1]2A[hPTc4r{na}] \`
`C[hTc4r{pro}|hTc7r{nad}|hTc3r{kvůli}]`

After the merging of actants, we get three sorts of frames: frames where every member has only one functor assigned, frames where actants are distinguished from free modifications, but some of the free modifications are ambiguous, and frames where at least one member is ambiguous between an actant and a free modification. Approximately one third of all merged frames fall in the first category and another thousand into the second one. These frames are candidates for further processing with help of the above mentioned algorithm, and therefore they will be separated from the rest which must be left for post-editing.

Now, we will describe the process of assigning functors in the categories where actants are distinguished from free modifications. These frames fall into two subcategories: frames with at most two inner participants (i.e. Actor and Patient) and frames with at least three inner participants. The former are done already and we do not need to process them any further. The latter will be processed by the algorithm for assigning functors, but let us first resume the starting conditions:

- We have at least three inner participants.
- Actor is already assigned to the subject.
- We have to decide which of the actants is Patient and what are functors of the remaining inner participants.

We will not describe the algorithm in detail, we only sketch the overall strategy. More details and a flow chart can be found in Appendix D.

- A rule (following from the actant shifting) which must be observed after every step of the algorithm is that Patient slot must be filled. If there is only one unassigned member and the Patient slot has not been filled yet then the last member of the frame is assigned the Patient functor.
- We start with searching for Origin as Origin has the narrowest set of possible surface realizations, which in addition are not "polysemous".

[8] For the list of abbreviations used for functors see Appendix B.4, for lists of functors attached to every surface realization see Appendix C.2.

- Addressee assignment is ruled by the animacy of surface forms rather than the morphological cases. Animate Accusative or an animate prepositional case are realizations of Addressee rather than inanimate Dative.
- The decision about Effect can be quite difficult. Beside the typical prepositional cases also Instrumental can be a surface form of Effect. We then have to take into consideration the remaining unassigned members of the frame and make a decisions about pairs of surface forms.

As was said above, approximately 7500 frames are processed by this algorithm and the program ends successfully in all cases. The remaining ca 11,000 frames must be edited manually, with help of an editor prepared by Z. Žabokrtský (see Skoumalová et al., prep). The editor's work should be easier as s/he gets a (small) set of possible functors which can be assigned to every member of a frame and s/he does not have to choose from all 47 possibilities.

## 5.3. Marking diatheses

We made a simple assumption that

- reflexive verbs cannot have any diatheses (the exception with the periphrastic passive of the verb *tázat se* was discussed above), and so they get the mark `@`.
- intransitive verbs[9] can form reflexive passive; they get the mark `$`.
- a verb with a member in Accusative or in an indirect case (without preposition) can form both periphrastic and reflexive passive; it gets marks `%$`
- a verb whose all objects are prepositional cases can form the reflexive passive; it gets the mark `$`.

During the automatic processing all frames are assigned these marks and corrections will be made by the post-editor. Actors, which were added automatically to all frames, are marked as general ((`hPTc1`)) in frames that allow for forming any passive, and they are marked as obligatory ([`hPTc1`]) in other frames.

## 5.4. Usage of the final lexicon

The final product can be used in NLP systems for parsing, tagging, grammar checking and similar purposes. In all these applications, however, all possible instances of single frames must be generated. In the next section, it will be shown how we obtain single sentence patterns from frames.

[9] The term intransitive verb here means a verb with only one actant realized as subject in Nominative.

### 5.4.1. Generating frame instances from frames

The frames shown so far represent in fact sets of frame instances in which only one of all variants for every member is chosen, optional members may or may not be present, surface realizations of deletable or general participants can be missing and the passive constructions can be formed. Let us take some frames from (23), which we repeat here and let us add some more complicated cases:

(132) a. `akumulovat` `R--s[i1]1(hPTc1)2[hTc4]%$` (accumulate st)

b. `kazit~2` `RDEs[i1]1[hTc1]@` (decay)

c. `přihlásit~1` `R--s[i1]1(hPc1)2[hPTSRc4]A[hTc2r{do}|hTc4r{na}]%$` (enroll sb/st where)

d. `slíbit~1` `R--s[i1]1(hPc1)2[hZc4|sD|sIq3d%]3[hPc3]%$#*` (promise)

The frame in (132a) represents the following set of instances:[10]

(133) a. `R--s[i1]1[hPTc1]2[hTc4]`
*Kamna akumulují teplo.*
Radiator accumulates heat.

b. `P--s[i2]1[hPTc7]2[hTc1]`
*Teplo je akumulováno kamny.*
Heat is accumulated by radiator.

c. `P--s[i2]1[hG]2[hTc1]`
*Teplo je akumulováno.*
Heat is accumulated.

d. `PSEs[i2]1[hG]2[hTc1]`
*Teplo SE akumuluje.*
Heat SE accumulates.

The frame in (132b) has no possible variants, so the only operation will be the deletion of the diathesis mark:

(134) `RDEs[i1]1[hTc1]`
*Potraviny se kazí.*
Eatables SE decay.

The frame in (132c), on the other hand, represents quite a large set of frame instances:

[10] In the frame instances, we use only one type of brackets, as they only serve as delimiters.

(135) a. `přihlásit~1 R--s[i1]1[hPc1]2[hPTSRc4]A[hTc2r{do}]`

b. `přihlásit~1 R--s[i1]1[hPc1]2[hPTSRc4]A[hTc4r{na}]`

c. `přihlásit~1 P--s[i2]1[hPc7]2[hPTc1]A[hTc2r{do}]`

d. `přihlásit~1 P--s[i2]1[hPc7]2[hPTc1]A[hTc4r{na}]`

e. `přihlásit~1 P--s[i2]1[hG]2[hPTc1]A[hTc2r{do}]`

f. `přihlásit~1 P--s[i2]1[hG]2[hPTc1]A[hTc4r{na}]`

g. `přihlásit~1 PSEs[i2]1[hG]2[hPTc1]A[hTc2r{do}]`

h. `přihlásit~1 PSEs[i2]1[hG]2[hPTc1]A[hTc4r{na}]`

So far, we only needed marks that have been already defined, but for all instances of the verb *slíbit* (promise) we will also need marks for "frames" with the support verbs *mít* and *dostat*. For this purpose, three new marks for a type of a frame were introduced:

`M` — construction with the support verb *mít*

`D` — construction with the support verb *dostat*

`T` — resultative construction with the verb *mít*

Now, we can generate all instances of the frame:

(136) a. `slíbit~1 R--s[i1]1[hPc1]2[sIq3d%]3[hPc3]`

b. `slíbit~1 R--s[i1]1[hPc1]2[sD]3[hPc3]`

c. `slíbit~1 R--s[i1]1[hPc1]2[hZc4]3[hPc3]`

d. `slíbit~1 P--s[i2]1[hPc7]2[sIq3d%]3[hPc3]`

e. `slíbit~1 P--s[i2]1[hPc7]2[sD]3[hPc3]`

f. `slíbit~1 P--s[i2]1[hPc7]2[hZc1]3[hPc3]`

g. `slíbit~1 P--s[i2]1[hG]2[sIq3d%]3[hPc3]`

h. `slíbit~1 P--s[i2]1[hG]2[sD]3[hPc3]`

i. `slíbit~1 P--s[i2]1[hG]2[hZc1]3[hPc3]`

j. `slíbit~1 PSEs[i2]1[hG]2[sIq3d%]3[hPc3]`

k. `slíbit~1 PSEs[i2]1[hG]2[sD]3[hPc3]`

l. `slíbit~1` `PSEs[i2]1[hG]2[hZc1]3[hPc3]`

m. `slíbit~1` `M--s[i3]1[hPc2r{od}]2[sIq3d%]3[hPc3]`

n. `slíbit~1` `M--s[i3]1[hPc2r{od}]2[sD]3[hPc3]`

o. `slíbit~1` `M--s[i3]1[hPc2r{od}]2[hZc4]3[hPc3]`

p. `slíbit~1` `M--s[i3]2[sIq3d%]3[hPc3]`

q. `slíbit~1` `M--s[i3]2[sD]3[hPc3]`

r. `slíbit~1` `M--s[i3]2[hZc4]3[hPc3]`

s. `slíbit~1` `D--s[i3]1[hPc2r{od}]2[sIq3d%]3[hPc3]`

t. `slíbit~1` `D--s[i3]1[hPc2r{od}]2[sD]3[hPc3]`

u. `slíbit~1` `D--s[i3]1[hPc2r{od}]2[hZc4]3[hPc3]`

v. `slíbit~1` `D--s[i3]2[sIq3d%]3[hPc3]`

w. `slíbit~1` `D--s[i3]2[sD]3[hPc3]`

x. `slíbit~1` `D--s[i3]2[hZc4]3[hPc3]`

### 5.4.2. Extracting subcat lists

For testing whether our lexicon can be used also in other theoretical frameworks we made a small experiment with LFG. The verbs frames were converted to *templates* which can be used in a lexicon. These templates are then processed by lexical rules which derive all sentence patterns.

Every template contains a *predicate* (i.e. lemma and a subcat list) on which the lexical rules will be applied. A template can also contain some constraint which apply for all verbs of a given category. We will show it on an example:

(137)
```
TRANSRFLPERPASS(P) =
        @(LR-TRANSRFLPERPASS (^ PRED)='P<(^ SUBJ)(^ OBJ)>').
TRANSRFLPERPASSDAT(P) =
        @(LR-TRANSRFLPERPASS {(^ PRED)='P<(^ SUBJ)(^ OBJ)(^ OBJ2)>'
                        (^ OBJ2 CASE)=DAT}).
```

P in parentheses and in the subcat list is a variable for the lemma. The template TRANSRFLPERPASS is used for transitive verbs which have only one object and they can be passivized by both ways. The template TRANSRFLPERPASSDAT is used for transitive verbs which have another object in Dative and which can also be passivized by both ways. Both the templates use the same set of lexical rule, namely LR-TRANSRFLPERPASS:

(138)
```
LR-TRANSRFLPERPASS(SCHEMATA) =
            { SCHEMATA
            (^ OBJ CASE)=ACC
            ~(^ REFL)
            |SCHEMATA
            (^ REFL)=c SE
            (^ OBJ)->(^ SUBJ)
            (^ OBJ CASE)=NOM
            (^ SUBJ)->NULL
            |SCHEMATA
            (^ OBJ)->(^ SUBJ)
            (^ OBJ CASE)=NOM
            (^ SUBJ)->NULL
            ~(^ REFL)
            ~(^ TENSE)
            (^ PARTICIPLE)=c PASS }.
```

Lexical rules work like functions on variables supplied by templates. SCHEMATA stands for the variable and it is filled either by a predicate, or by a predicate an further constraints.

In the above example, we can see that three construction are created by the lexical rule LR-TRANSRFLPERPASS. The first construction is an active sentence where the object is in Accusative and the reflexive form of the verb is prohibited. The second construction is a reflexive passive, the object takes the position of a subject and the original subject is deleted. The third construction is a periphrastic passive, where, again, the object takes the position of a subject and the original subject is deleted, and further, the verb must have a form of passive participle and no reflexive particle can be part of the verb construction.

An experimental grammar was written for testing the lexicon. The lexicon only contains verbs from regular morphological paradigms so that the morphological module would not to be to large. Results of processing testing sentences are shown in Appendix F.