Transfer rules examples

En français

This page is intended to supplement the page A long introduction to transfer rules. Examples used are taken from apertium-eo-fr pair. It is (at the beginning of 2013) a released pair for translating French to Esperanto. But Esperanto → French translation direction had not been implemented by the initial developer. It is another developer, full beginner for writing transfer rules who chose to do that. The examples given are the first rules written to translate a group of one, two or three Esperanto words into a group of two or three French words.

This page is only about writing the file with the suffix `.t1x` with rules intended to be used by the tool *apertium-transfer*. Writing tags used for chunking in a 3-stage transfer is not approached there.

Contents

Different steps for a translation with apertium

Let start by making a list of the different operations done for a translation.

Operation	Role	Concerned languages
Deformatting	Allows to mark zones of the source text not to be translated. For example, HTML tags must not be translated in another language, but only the text of the Web page.	The same software are used for every language pairs. It is the format of the data to be translated which will take to use a particular deformatter.
Analysis	Each word of the source text is decomposed into a lemma followed by the type of the word and its attributes (gender, number, person and tense for a verb ...). For some words, several analyses are possible. In this case, they all are sent on output.	Valid for every languages, it uses the morphological dictionary of the source language.
Disambiguation	When there are several analysis for a word, this step permits to keep only one.	Valid for every languages, it uses a file with `.prob` suffix For non ambiguous languages as Esperanto, this step stays necessary to take off the surface form of each analysed word (pre-formatting for the transfer step).
Pre-transfer	Processing multiwords before transfer step.	All languages. Does not require a particular data file.
Transfer	Transforms analyses from the source language into their translated version in the target language.	Valid for every languages, it uses the bilingual dictionary and the transfer file with `.t1x` suffix.
Interchunk processing	Allows processing on groups of words (the subject, a complement ...) As indicated above, we will not deal with this step (nor of the following).	Used a priori to make the transfer step more simple, it needs to add several tags during the transfer step. It uses a file with `.t2x` suffix and eventually other files if several pass of this kind are done.
Postchunk	End of interchunk processing(s)	Needed if one or more interchunk processing were done. It uses a file with `.t3x` suffix.
Generation	Generate the surface forms of the target language words from the decomposition in lemma + attributes obtained from the previous steps.	Valid for every languages, it uses the morphological dictionary of the target language.
Post-generation	Allows spelling corrections between following words when particular cases are not processed by the generation.	Used in a lot of target languages (including French), may be not for all.
Reformatting	Put the translated data back to the format of the source document.	The same software are used for every language pairs. There is a reformatter for each available deformatter even in every reformatter do a similar work.

The page Preparing to use apertium-transfer-tools gives an example about how a Spanish sentence is changed at every step of the process to lead finally to an English translation.

How to find what must be done

Basically, the transfer step starts from a disambiguated analysis of the source language text to provide an equivalent in the target language. The generation step then does the inverse processing as the analysis. It has a consequence : data given to generator must be exactly what a new analysis of the text translated in the target language would give. Otherwise, the generation will be only partial with some # appearing at the beginning of some words that will be written as lemmas.

Example :

We want to translate in French the 3 Esperanto words :

```
la aŭtomata traduko
```

After **analysis and disambiguation**, we get :

```
^la<det><def><sp>$ ^aŭtomata<adj><sg><nom>$ ^traduko<n><sg><nom>$
```

A **lexical transfer** step (using only the bilingual dictionary) will give :

```
^le<det><def><sp>$ ^automatique<adj><sg><nom>$ ^traduction<n><f><sg><nom>$
```

The **part of sentence we want to get in French** is :

```
la traduction automatique
```

When analysing this part of sentence, we get :

```
^le<det><def><f><sg>$ ^traduction<n><f><sg>$ ^automatique<adj><mf><sg>$
```

which is the text we must give to the generator to get the desired translation.

So, during the **structural transfer** step, we will have to do the following changes :

Origin :

```
^le<det><def><sp>$ ^automatique<adj><sg><nom>$ ^traduction<n><f><sg><nom>$
```

Result :

```
^le<det><def><f><sg>$ ^traduction<n><f><sg>$ ^automatique<adj><mf><sg>$
```

For this, we write the transfer rules. Their goal is to add or remove several tags in words descriptions, and possibly to change the order of certain words.

Structure of a .t1x file

The file containing transfer rules has the suffix `.t1x` . This file is made of several mandatory sections and can also contain other optional sections. Each section will have to contain at least one element.

```
<?xml version="1.0" encoding="UTF-8"?>
<transfer>
   <section-def-cats>
      ..........
   </section-def-cats>

   <section-def-attrs>
      ..........
   </section-def-attrs>

   <section-def-vars>
      ..........
   </section-def-vars>

   <section-def-macros>
      ..........
   </section-def-macros>

   <section-rules>
      ..........
   </section-rules>
</transfer>
```

def-cats section

The **def-cats** section is mandatory. It allows to declare **categories** of word that we will fetch to apply a particular transfer rule. It can be simple words (a determinant, a noun, an adjective, a verb, ...) or a little more complicated things as a noun with in its description the tag <nom> (nominative) meaning it is part of the subject of the sentence.

This section contains one or more element with the following structure :

```
    <def-cat n="name_of_what_we_want_to_describe">
      <cat-item tags="its_description"/>
      .... (there can be one or more <cat-item .../> tags)
    </def-cat>
```

def-attrs section

The **def-attrs** section is mandatory. It allows to put together by functionality **attribute** names for words defined in the section **sdefs** of a morphological dictionary. For example, we will put together in this section every tag corresponding to the :

- gender of a word
- number of a word (singular, plural, ...)
- person of a verb
- tense of a verb
- ...

This section contains one or more element with the following structure :

```
    <def-attr n="name_of_a_list_of_attributes_with_a_common_rule">
      <attr-item tags="an_attribute_of_the_sdef_section_of_a_dictionary"/>
      .... (we have several tags <attr-item .../> as many as possible
            values for the attribute)
    </def-attr>
```

def-vars section

The **def-vars** section is mandatory and must contain at least one element with the following syntax `<def-var n="..."/>` . It lists the global variables used in the transfer rules. However, for the rules described in this page, we will not need any of these variables.

def-macros section

The **def-macros** section is optional. Nevertheless, it will be very useful to write shorter transfer files avoiding to duplicate identical (or almost) operations done in several transfer rules.

This section contains one or more element with the following structure :

```
    <def-macro n="name_of_the_macro" npar="number_of_parameters">
      .... (the code of the macro)
    </def-macro>
```

rules section

Finally, the **rules** section is mandatory. It is the longest of the transfer file and the one that justifies its existence. It indeed makes it possible to define the operations to be performed to translate groups of words (or sometimes single words, as we will see).

This section contains one or more element with the following structure :

```
<rule>
  <pattern>
    <pattern-item n="name_defined_in_def-cat_corresponding_to_the_first_word_to_process"/>
    .... (as many tags <pattern-item ..../> as words we want to process together)
  </pattern>
  <action>
    .... (description of the transfer rule)
  </action>
</rule>
```

Examples of transfer rules

Transferring two words making them agree

We will start to translate to French the Esperanto determinant **la** followed by a common noun.

Search for modifications

In Esperanto, the definite determinant **la** is invariant, while in French, it has three forms: **le**, **la**, **les** according to gender and number of the noun to which it agrees.

For the common noun, there are two forms in Esperanto depending on whether it belongs to the subject or to the object complement in the sentence. In French, it is written the same way in both cases.

Examples :

Esperanto	Esperanto analyses	French	French analyses
la tago la tagon	^la<det><def><sp>$ ^tago<n><sg><nom>$ ^la<det><def><sp>$ ^tago<n><sg><acc>$	le jour	^le<det><def><m><sg>$ ^jour<n><m><sg>$
la nokto la nokton	^la<det><def><sp>$ ^nokto<n><sg><nom>$ ^la<det><def><sp>$ ^nokto<n><sg><acc>$	la nuit	^le<det><def><f><sg>$ ^nuit<n><f><sg>$
la tagoj la tagojn	^la<det><def><sp>$ ^tago<n><pl><nom>$ ^la<det><def><sp>$ ^tago<n><pl><acc>$	les jours	^le<det><def><mf><pl>$ ^jour<n><m><pl>$
la noktoj la noktojn	^la<det><def><sp>$ ^nokto<n><pl><nom>$ ^la<det><def><sp>$ ^nokto<n><pl><acc>$	les nuits	^le<det><def><mf><pl>$ ^nuit<n><f><pl>$

Let examine what the lexical translation of the Esperanto analysis gives and compare it to the analysis in French we wants to submit to the generator:

Esperanto analyses	Esperanto analyses translated in French	The analyses in French of what we want to get
^la<det><def><sp>$ ^tago<n><sg><nom>$ ^la<det><def><sp>$ ^tago<n><sg><acc>$	^le<det><def><sp>$ ^jour<n><m><sg><nom>$ ^le<det><def><sp>$ ^jour<n><m><sg><acc>$	^le<det><def><m><sg>$ ^jour<n><m><sg>$
^la<det><def><sp>$ ^nokto<n><sg><nom>$ ^la<det><def><sp>$ ^nokto<n><sg><acc>$	^le<det><def><sp>$ ^nuit<n><f><sg><nom>$ ^le<det><def><sp>$ ^nuit<n><f><sg><acc>$	^le<det><def><f><sg>$ ^nuit<n><f><sg>$
^la<det><def><sp>$ ^tago<n><pl><nom>$ ^la<det><def><sp>$ ^tago<n><pl><acc>$	^le<det><def><sp>$ ^jour<n><m><pl><nom>$ ^le<det><def><sp>$ ^jour<n><m><pl><acc>$	^le<det><def><m><pl>$ ^jour<n><m><pl>$
^la<det><def><sp>$ ^nokto<n><pl><nom>$ ^la<det><def><sp>$ ^nokto<n><pl><acc>$	^le<det><def><sp>$ ^nuit<n><f><sg><nom>$ ^le<det><def><sp>$ ^nuit<n><f><pl><acc>$	^le<det><def><f><pl>$ ^nuit<n><f><pl>$

We can note :

- for the determinant, the lexical translation always gives ^le<det><def><sp>$. It will be necessary to replace the last tag <sp> (singular or plural) by tags used by the common noun giving its gender and number.
- for the common noun, the lexical translation found (in the bilingual dictionary) the gender of the noun translated to French. To know if this noun is singular or plural, it kept the number attribute of the original language. But the attribute <nom> or <acc> which is not needed in French was also kept and it can prevent to generate the word. So, this attribute will have to be removed by the transfer rule.

Writing the transfer rule

For this first rule, we start from a "empty" file with `.t1x` suffix having the structure described here.

As the **def-macros** section is optional and not used for the first transfer rules described in this page, we will not put it for the present.

The **def-vars** section is mandatory. Although it will never be used in the examples this page, we will just put a minimum content so that the file `.t1x` can be compiled:

```
<section-def-vars>
  <def-var n="aucune_variable"/>
</section-def-vars>
```

The other sections may contain useful information for our first transfer rule.

def-cats section

In this section, we will define 2 word categories :

- determinants written as **det** which are identified in analysis by the tag **<det>** followed by anything.
- common noun written as **nom_commun** which are identified in analysis by the tag **<n>** followed by anything.

The **def-cats** section will be written as follow :

```
<section-def-cats>
  <def-cat n="det">
    <cat-item tags="det.*"/>
  </def-cat>

  <def-cat n="nom_commun">
    <cat-item tags="n.*"/>
  </def-cat>
</section-def-cats>
```

- names of word categories are in the attribute **n** of **<def-cat n="...">** tags
- descriptions of what must be found into analysis to recognize the word category are in the attribute **tags** of **<cat-item tags="..."/>** tags.

def-attrs section

Now we will define possible attributes to the various tags of words

```
<section-def-attrs>
  <def-attr n="type_mot">
    <attr-item tags="n"/>
    <attr-item tags="det"/>
  </def-attr>

  <def-attr n="genre">
    <attr-item tags="m"/>
    <attr-item tags="f"/>
    <attr-item tags="mf"/>
  </def-attr>

  <def-attr n="nombre">
    <attr-item tags="sg"/>
    <attr-item tags="pl"/>
    <attr-item tags="sp"/>
  </def-attr>
</section-def-attrs>
```

- In the **n** attribute of tags **<def-attr n="...">**, we give a name to the various characteristics of the words we want to process
- for each of these characteristics, **<attr-item tags="..."/>** tags indicate the different possible values of this characteristic.

For the rule we want to write, we defined 3 characteristics :

- **type_mot** (may be mandatory, but there is no documented alternative solution). Presently, the available types are:
  - n (common noun)
  - det (determinant)

We will add some others later when we will write other rules.

- **genre** with the possible values
  - m (masculine)
  - f (feminine)
  - mf (masculine or feminine)
- **nombre** with the possible values
  - sg (singular)
  - pl (plural)
  - sp (singular or plural)

rules section

A **rules section** containing only the rule we want to write will contain:

```
<section-rules>
  <rule>
    <pattern>
      <pattern-item n="det"/>
      <pattern-item n="nom_commun"/>
    </pattern>
    <action>
      <out>
        <lu>
          <clip pos="1" side="tl" part="lem"/>
          <clip pos="1" side="tl" part="type_mot"/>
          <lit-tag v="def"/>
          <clip pos="2" side="tl" part="genre"/>
          <clip pos="2" side="tl" part="nombre"/>
        </lu>
        <b />
        <lu>
          <clip pos="2" side="tl" part="lem"/>
          <clip pos="2" side="tl" part="type_mot"/>
          <clip pos="2" side="tl" part="genre"/>
          <clip pos="2" side="tl" part="nombre"/>
        </lu>
      </out>
    </action>
  </rule>
</section-rules>
```

The rule is made of 2 sections :

```
<pattern>
  <pattern-item n="det"/>
  <pattern-item n="nom_commun"/>
</pattern>
```

In this part, we specifies which are the successive categories of words that must be found in the analysis of the source text so that the rule can apply. In this case, we will have to find a determinant, followed of a common noun. The attributes of **<pattern-item n="..."/>** tags must all have been defined in the **def-cats** section, otherwise the rule could never be applied.

The most interesting part of the rule is starting from the **<action>** tag. It has the following structure:

```
<action>
  <out>
    <lu>
      ... (generation of the lexical unit for the first word)
    </lu>
    <b />
    <lu>
      ... (generation of the lexical unit for the second word)
    </lu>
  </out>
</action>
```

In this rule, we only generate data that we send on output. The contents of **<action>** tag is therefore limited to the generation of the text that is indicated in **<out>** tag.

We will have to generate the analysis of 2 words in the target language. Analysis of each word is a lexical unit]] (**<lu>** tag) which on output will be symbolized by the characters **^...$** where the description of the lexical unit will replace the dotted lines.

Between the two lexical units, we will leave a space (tag) otherwise, the two words generated would be stick.

Let us examine how lexical units are written :

The first tag **<clip pos="1" side="tl" part="lem"/>** has element by element the following meaning:

Part	Meaning
clip	This is a keyword which can be translated by "get"
pos="1"	It is the number of the **pattern-item** in the list **<pattern>...</pattern>** of the rule. Here, pos="1" corresponds to the analysis of the determinant
side="tl"	We get the information from the target language. To access to the source language, we would write **side="sl"**
part="lem"	This is a reserved keyword corresponding to the lemma.

The third **<lit-tag v="def"/>** tag has element by element the following meaning:

Part	Meaning
lit-tag	This is a keyword which can be translated by "generate a tag"
v="def"	Here we specify the contents of the tag. In this case, **<def>** will be generated.

The 5 instruction necessary to generate the analysis of the determinant have the following meaning:

Instruction	Meaning
<clip pos="1" side="tl" part="lem"/>	Get the lemma of the first word of the pattern in the target language. It will always be French article "le".
<clip pos="1" side="tl" part="type_mot"/>	Get the type of the first word of the pattern in the target language. It will be **det**.
<lit-tag v="def"/>	Generate a **def** tag, that is the text **<def>** which specifies that the determinant is *defined*.
<clip pos="2" side="tl" part="genre"/>	Get the gender of the second word of the pattern in the target language, that is the gender of the common noun.
<clip pos="2" side="tl" part="nombre"/>	Get the number of the second word of the pattern in the target language, that is the number of the common noun.

The 5 elements we got constitute constitutes the lexical unit **<lu>...</lu>** that will be sent on output using the tag **<out>...</out>**

For the second lexical unit corresponding to the common noun translation, we can notice that we have on each line : **pos="2" side="tl"** meaning that we will simply copy several tags of the common noun (2nd word of the rule).

Detailed explanation of the four instructions:

Instruction	Meaning
<clip pos="2" side="tl" part="lem"/>	Get the lemma of the second word of the pattern in the target language (the common noun in French).
<clip pos="2" side="tl" part="type_mot"/>	Get the type of the second word. That will be **n**.
<clip pos="2" side="tl" part="genre"/>	Get the gender of the common noun.
<clip pos="2" side="tl" part="nombre"/>	Get the number of the common noun.

Note

If we send to the generator the result of the of the transfer, we don't get exactly what is needed :

French analysis	Result of the generation	What is needed
^le<det><def><m><sg>$ ^jour<n><m><sg>$	~le jour	le jour
^le<det><def><f><sg>$ ^nuit<n><f><sg>$	~la nuit	la nuit
^le<det><def><mf><pl>$ ^jour<n><m><pl>$	~les jours	les jours
^le<det><def><mf><pl>$ ^nuit<n><f><pl>$	~les nuits	les nuits
^le<det><def><m><sg>$ ^arbre<n><m><sg>$	~le arbre	l'arbre
^le<det><def><f><sg>$ ^histoire<n><f><sg>$	~la histoire	l'histoire
^le<det><def><m><pl>$ ^arbre<n><m><pl>$	~les arbres	les arbres
^le<det><def><f><pl>$ ^histoire<n><f><pl>$	~les histoires	les histoires

The replacement of the determinant **le/la** by **l'** according to the first letter of the following word is not done during the generation but just after during the post-generation step which process the words marked by a ~ . This remark being done, the post-generation will not be mentioned again in this page.

Adding a word in the target language text

Esperanto does not have any indefinite determinant. To translate **un**, **une**, **des**, we simply do not put the definite determinant *la* before the common noun. A common noun written alone in Esperanto will have to be preceded by the correct indefinite determinant **un**, **une** or **des**, if it is translated in French.

Our second rule will make this transformation.

Let examine what gives the lexical translation of the Esperanto analysis and compare it to the analysis in French we want to submit to the generator:

Esperanto analysis	Esperanto analysis translated to French	French analysis that we want to get
^tago<n><sg><nom>$ ^tago<n><sg><acc>$	^jour<n><m><sg><nom>$ ^jour<n><m><sg><acc>$	^un<det><ind><m><sg>$ ^jour<n><m><sg>$
^nokto<n><sg><nom>$ ^nokto<n><sg><acc>$	^nuit<n><f><sg><nom>$ ^nuit<n><f><sg><acc>$	^un<det><ind><f><sg>$ ^nuit<n><f><sg>$
^tago<n><pl><nom>$ ^tago<n><pl><acc>$	^jour<n><m><pl><nom>$ ^jour<n><m><pl><acc>$	^un<det><ind><m><pl>$ ^jour<n><m><pl>$
^nokto<n><pl><nom>$ ^nokto<n><pl><acc>$	^nuit<n><f><sg><nom>$ ^nuit<n><f><pl><acc>$	^un<det><ind><f><pl>$ ^nuit<n><f><pl>$

Compared to the previous rule, instead of generating **^le<det><def><*gender*><*number*>$** we will generate **^un<det><ind><*gender*><*number*>$**. Everything else is unchanged.

To write the new rule, we already have all what we need in **def-cats** and **def-attrs** sections . So, we will just have to add the new rule in the **rules** section that will become:

```
<section-rules>
  <rule>
    <pattern>
      <pattern-item n="det"/>
      <pattern-item n="nom_commun"/>
    </pattern>
    <action>
      ... (see the contents in the preceding paragraph)
    </action>
  </rule>

  <rule>
    <pattern>
      <pattern-item n="nom_commun"/>
    </pattern>
    <action>
      <out>
        <lu>
          <lit v="un"/>
          <lit-tag v="det.ind"/>
          <clip pos="1" side="tl" part="genre"/>
          <clip pos="1" side="tl" part="nombre"/>
        </lu>
        <b />
        <lu>
          <clip pos="1" side="tl" part="lem"/>
          <clip pos="1" side="tl" part="type_mot"/>
          <clip pos="1" side="tl" part="genre"/>
          <clip pos="1" side="tl" part="nombre"/>
        </lu>
      </out>
    </action>
  </rule>
```

In this new rule, we find for the first time the instruction **lit** that will generate a string, contrarily to **lit-tag** which includes the generated string inside < > so that it becomes a tag.

As in the text of the source language to be transferred, there is only one word (the common noun mentioned in the pattern), we can access its attributes by **pos="1"** whereas it was **pos="2"** in the first rule.

The 4 instructions needed to generate the analysis of the indefinite determinant have the following meaning:

Instruction	Meaning
<lit v="un"/>	Generate the lemma "un".
<lit-tag v="det.ind"/>	Generate a **det** tag followed by a **ind** tag, that is the text **<det><ind>** which makes it possible to specify that we generate a *indefinite determinant*.
<clip pos="1" side="tl" part="genre"/>	Get the gender of the common noun.
<clip pos="1" side="tl" part="nombre"/>	Get the number of the common noun.

The instructions to generate the translation in French of the common noun are the same ones as for the previous rule, except that now **pos="1"**.

Interchange two words

Now we will see a rule to change the order of two words during a translation.

In Esperanto, it is recommended to put the adjective before the noun but it is not mandatory. The Apertium Spanish -> Esperanto translator preserves the word order of the Spanish sentence whereas The Apertium French -> Esperanto translator puts the adjective before the noun.

In French, most of the adjectives are placed after the noun they qualify, but some adjectives are placed before.

The complete solution would process all the possible cases in Esperanto as in French. We will limit ourselves to the most frequent case by writing a rule which starting from a form "la" + adjective + noun in Esperanto, provides a translation such as "le/la/les" + noun + adjective in French.

To be added in the def-cats section

In this section, we will add a category for the adjectives:

```
    <def-cat n="adj">
      <cat-item tags="adj.*"/>
    </def-cat>
```

To be added in the def-attrs section

In the words type list (type_mot), we add adjectives:

```
    <def-attr n="type_mot">
      <attr-item tags="n"/>
      <attr-item tags="det"/>
      <attr-item tags="adj"/>
    </def-attr>
```

Adding the rule which will invert the adjective and the noun

```
    <rule>
      <pattern>
        <pattern-item n="det"/>
        <pattern-item n="adj"/>
        <pattern-item n="nom_commun"/>
      </pattern>

      <action>
        <out>
          <lu>
            <clip pos="1" side="tl" part="lem"/>
            <clip pos="1" side="tl" part="type_mot"/>
            <lit-tag v="def"/>
            <clip pos="3" side="tl" part="genre"/>
            <clip pos="3" side="tl" part="nombre"/>
          </lu>
          <b />
          <lu>
            <clip pos="3" side="tl" part="lem"/>
            <clip pos="3" side="tl" part="type_mot"/>
            <clip pos="3" side="tl" part="genre"/>
            <clip pos="3" side="tl" part="nombre"/>
          </lu>
          <b />
          <lu>
            <clip pos="2" side="tl" part="lem"/>
            <clip pos="2" side="tl" part="type_mot"/>
            <clip pos="3" side="tl" part="genre"/>
            <clip pos="3" side="tl" part="nombre"/>
          </lu>
        </out>
      </action>
    </rule>
```

We can note that in this rule we generate first the determinant (pos = 1), then the noun (pos = 3 in the pattern) and finally the adjective (pos = 2 in the pattern). To swap two words, we only needed to generate the lexical units **<lu>...</lu>** in a different order.

In this rule, the determinant and the adjective agree in gender and number with the noun.

Changing attributes according to conditions

Now, we will examine a rule to translate a personal pronoun followed by a verb applying the conjugation rules.

Searching modifications to be made

- In Esperanto, the verb is invariant according to the personal pronoun witch is just before ((or more generally according to the subject).
- In French, the verb agrees with the person and the number of the personal pronoun (but not with its gender).

In addition, some of the French personal pronouns have no specific equivalent in Esperanto which is for this point like English:

- **tu** (second person singular) and **vous** (second person plural) in French are both translated by **vi** in Esperanto.
- **ils** and **elles** (masculine and feminine forms of the 3rd person plural) are translated by **ili** in Esperanto.

To translate from Esperanto to French, we will then have to make choices:

- **vi → vous** second person plural or polite form to speak to a single person
- **ili → ils** we choose the masculine for the 3rd person plural in French.

Similarly, Esperanto has only one tense for the past where French has four. In addition, in an analysis, Esperanto and French dictionaries do not use the same abbreviation for the present indicative. It will thus be necessary to change all that during the translation.

We will see what all this gives for the verb **kanti → chanter** conjugated in the present indicative.

Esperanto	Esperanto analyses	Esperanto analyses translated	The analysis we would like to get	French
mi kantas	^prpers<prn><subj><p1><mf><sg>$ ^kanti<vbtr_ntr><pres>$	^prpers<prn><p1><mf><sg>$ ^chanter<vblex><pres>$	^prpers<prn><p1><mf><sg>$ ^chanter<vblex><pri><p1><sg>$	je chante
vi kantas	^prpers<prn><subj><p2><mf><sp>$ ^kanti<vbtr_ntr><pres>$	^prpers<prn><p2><mf><sp>$ ^chanter<vblex><pres>$	^prpers<prn><p2><mf><pl>$ ^chanter<vblex><pri><p2><pl>$	tu chantes → vous chantez
li kantas	^prpers<prn><subj><p3><m><sg>$ ^kanti<vbtr_ntr><pres>$	^prpers<prn><p3><m><sg>$ ^chanter<vblex><pres>$	^prpers<prn><p3><m><sg>$ ^chanter<vblex><pri><p3><sg>$	il chante
ŝi kantas	^prpers<prn><subj><p3><f><sg>$ ^kanti<vbtr_ntr><pres>$	^prpers<prn><p3><f><sg>$ ^chanter<vblex><pres>$	^prpers<prn><p3><f><sg>$ ^chanter<vblex><pri><p3><sg>$	elle chante
ni kantas	^prpers<prn><subj><p1><mf><pl>$ ^kanti<vbtr_ntr><pres>$	^prpers<prn><p1><mf><pl>$ ^chanter<vblex><pres>$	^prpers<prn><p1><mf><pl>$ ^chanter<vblex><pri><p1><pl>$	nous chantons
vi kantas	^prpers<prn><subj><p2><mf><sp>$ ^kanti<vbtr_ntr><pres>$	^prpers<prn><p2><mf><sp>$ ^chanter<vblex><pres>$	^prpers<prn><p2><mf><pl>$ ^chanter<vblex><pri><p2><pl>$	vous chantez
ili kantas	^prpers<prn><subj><p3><mf><pl>$ ^kanti<vbtr_ntr><pres>$	^prpers<prn><p3><mf><pl>$ ^chanter<vblex><pres>$	^prpers<prn><p3><m><pl>$ ^chanter<vblex><pri><p3><pl>$	ils chantent (elles chantent)

Writing the transfer rule

To be added in the def-cats section

In this section, we will add a category for pronouns and a category for verbs:

```
    <def-cat n="prn">
      <cat-item tags="prn.*"/>
    </def-cat>

    <def-cat n="verbe">
      <cat-item tags="vbser.*"/>
      <cat-item tags="vblex.*"/>
      <cat-item tags="vbtr.*"/>
      <cat-item tags="vbntr.*"/>
      <cat-item tags="vbtr_ntr.*"/>
    </def-cat>
```

As there are in Esperanto many forms for verbs, we put several **cat-item** to list all of them.

To be added in the def-attrs section

According to verbs, different keywords are used in Esperanto, whereas in French, almost all the verbs are classified vblex.

In the words type list (type_mot), we add verbs (several possibilities) and pronouns:

```
    <def-attr n="type_mot">
      .......... (what there was before)
      <attr-item tags="prn"/>
      <attr-item tags="vblex"/>
      <attr-item tags="vbmod"/>
      <attr-item tags="vbser"/>
      <attr-item tags="vbhaver"/>
    </def-attr>
```

We also add the two categories *personne* and *temps* for the conjugation of verbs:

```
    <def-attr n="personne">
      <attr-item tags="p1"/>
      <attr-item tags="p2"/>
      <attr-item tags="p3"/>
    </def-attr>

    <def-attr n="temps">
      <attr-item tags="pres"/>
      <attr-item tags="past"/>
      <attr-item tags="pri"/>
      <attr-item tags="pii"/>
      <attr-item tags="fti"/>
    </def-attr>
```

Before writing the rules section, some changes are needed for the verb tenses and for the gender and number of pronouns.

Transformation for the tense

For this example, we will limit to the indicative tenses.

In Esperanto, there are 3 indicative tenses:

- the past : past
- the present : pres
- the future : fti

In French, there are 6 more or less common tenses for the indicative:

- the *imparfait* :pii
- the *passé simple* (simple past) : ifi
- the *passé composé* (compound past) that should be made with the verb *avoir* (to have) + the past participle.
- the *plus que parfait* (plus perfect) (same problem as for the passé composé)
- the present : pri
- the future : fti

For verbs at the future, the attribute **fti** can be kept unchanged

For verbs at the present, it will be necessary to replace the attribute **pres** used in Esperanto by **pri**.

For verbs at the past, compound past should be nice for a translation, but less easy to generate. For this example we will replace the **past** attribute used in Esperanto by **pii** (imparfait).

In algorithmic form, that makes the following conditional transformations:

```
IF temps = "pres" THEN
    temps <- "pri"
ELSE IF temps = "past" THEN
    temps <- "pii"
END IF
```

Transformation of the pronoun attributes

For the pronoun, we will do the following changes:

```
IF personne = "p2" THEN
    nombre <- "pl"
ELSE IF (personne = "p3" AND nombre = "pl" THEN
    genre <- "m"
END IF
```

rules section

The new rule has the following contents:

```
    <rule>
      <pattern>
        <pattern-item n="prn"/>
        <pattern-item n="verbe"/>
      </pattern>

      <action>

        <choose>
          <when>
            <test>
               <equal>
                  <clip pos="2" side="sl" part="temps"/>
                  <lit-tag v="pres"/>
                </equal>
            </test>
            <let>
              <clip pos="2" side="tl" part="temps"/>
              <lit-tag v="pri"/>
            </let>
          </when>
          <when>
            <test>
               <equal>
                  <clip pos="2" side="sl" part="temps"/>
                  <lit-tag v="past"/>
                </equal>
            </test>
            <let>
              <clip pos="2" side="tl" part="temps"/>
              <lit-tag v="pii"/>
            </let>
          </when>
        </choose>

        <choose>    <!--  special cases for pronouns transfers  -->
          <when>    <!--  2nd person always plural : vi -> vous  -->
            <test>
               <equal>
                  <clip pos="1" side="sl" part="personne"/>
```

```
                <lit-tag v="p2"/>
              </equal>
            </test>
            <let>
              <clip pos="1" side="tl" part="nombre"/>
              <lit-tag v="pl"/>
            </let>
          </when>
          <when>    <!--  3rd person plural always masculine : ili -> ils  -->
            <test>
               <and>
                 <equal>
                    <clip pos="1" side="sl" part="personne"/>
                    <lit-tag v="p3"/>
                  </equal>
                 <equal>
                    <clip pos="1" side="sl" part="nombre"/>
                    <lit-tag v="pl"/>
                  </equal>
                </and>
            </test>
            <let>
              <clip pos="1" side="tl" part="genre"/>
              <lit-tag v="m"/>
            </let>
          </when>
        </choose>

        <out>
          <lu>
            <clip pos="1" side="tl" part="lem"/>
            <clip pos="1" side="tl" part="type_mot"/>
            <clip pos="1" side="tl" part="personne"/>
            <clip pos="1" side="tl" part="genre"/>
            <clip pos="1" side="tl" part="nombre"/>
          </lu>
          <b />
          <lu>
            <clip pos="2" side="tl" part="lem"/>
            <clip pos="2" side="tl" part="type_mot"/>
            <clip pos="2" side="tl" part="temps"/>
            <clip pos="1" side="tl" part="personne"/>
            <clip pos="1" side="tl" part="nombre"/>
          </lu>
        </out>
      </action>
    </rule>
```

For the first time, the **action** part of the rule does not limit to a block **<out>...</out>**, but starts with two **choose** blocks each having the following structure:

```
        <choose>
          <when>
            <test>
```

```
            .... (a condition)
          </test>
          <let>
            .... (action if this condition is true)
          </let>
        </when>
        <when>
          <test>
            .... (alternative to the previous condition)
          </test>
          <let>
            .... (action if the alternative condition is true)
          </let>
        </when>
      </choose>
```

Let us examine in detail the first block **<when>...</when>**

```
        <when>
          <test>
            <equal>
              <clip pos="2" side="sl" part="temps"/>
              <lit-tag v="pres"/>
             </equal>
          </test>
          <let>
            <clip pos="2" side="tl" part="temps"/>
            <lit-tag v="pri"/>
          </let>
        </when>
```

We start from inside the tags, then we will go up towards the including tags.

Instruction	Meaning
<clip pos="2" side="sl" part="temps"/>	Get the "temps" attribute of the 2nd word concerned with the rule (that is the verb) from the source language side
<lit-tag v="pres"/>	Generate a **pres** tag
<equal>...</equal>	Check if the 2 preceding values are equal
<test>...</test>	Decide if the block of instruction just afterwards must be executed.

Then, here is what is done when the test condition is true :

Instruction	Meaning
<clip pos="2" side="tl" part="temps"/>	Get (or access to) the "temps" attribute of the 2nd word concerned with the rule (that is the verb) from the target language side
<lit-tag v="pri"/>	Generate a **pri** tag
<let>...</let>	Seems to be an assignment of the second value into the first one

By the same way, the second block **<when>...</when>**

```
          <when>
            <test>
               <equal>
                  <clip pos="2" side="sl" part="temps"/>
                  <lit-tag v="past"/>
                </equal>
            </test>
            <let>
              <clip pos="2" side="tl" part="temps"/>
              <lit-tag v="pii"/>
            </let>
          </when>
```

tests whether the tense of the verb is "past" and in this case gives it the value "pii" for the target language.

Inside the conditional instructions for the pronoun, there is a more complicated **test** block:

```
            <test>
               <and>
                 <equal>
                    <clip pos="1" side="sl" part="personne"/>
                    <lit-tag v="p3"/>
                  </equal>
                 <equal>
                    <clip pos="1" side="sl" part="nombre"/>
                    <lit-tag v="pl"/>
                  </equal>
               </and>
            </test>
```

inside the block **<and>...</and>**, there are two blocks **<equal>...</equal>** (there could be more) and the condition is true if the two equalities are simultaneously verified : in this case "p3" for the attribute *personne* **and** "pl" for the attribute *nombre*.

In other rules, we could also find **<or>...</or>** blocks for which the condition is true if at least one of the conditions inside the block is.

In the same way, there are **<not>** and **</not>** tags to take the opposite of a condition. If two things we compare must be different, we will write:

```
      <not>
        <equal>
           ......
        </equal>
      </not>
```

To finish, we could wonder whether the two **choose** blocks of the rule we just studied could be combined in only one.

A try shows that the answer is no. When inside a **<choose>...</choose>** block we find several **<when>...</when>** blocks, the first of these blocks for which the condition is true makes the instructions of **<let>...</let>** block executed, and then the other following **<when>...</when>** blocks are not processed. The various tests inside the **<when>...</when>** blocks relate to exclusive conditions that we translate into algorithmic language by *ELSE IF*. There is also the possibility to put a **<otherwise>... </otherwise>** block to specify what must be done when none of the conditions of the various **<when>...</when>** blocks is true. It corresponds in algorithmic language to *ELSE* keyword.

The end of the rule:

```
       ......
        <out>
          <lu>
            <clip pos="1" side="tl" part="lem"/>
            <clip pos="1" side="tl" part="type_mot"/>
            <clip pos="1" side="tl" part="personne"/>
            <clip pos="1" side="tl" part="genre"/>
            <clip pos="1" side="tl" part="nombre"/>
          </lu>
          <b />
          <lu>
            <clip pos="2" side="tl" part="lem"/>
            <clip pos="2" side="tl" part="type_mot"/>
            <clip pos="2" side="tl" part="temps"/>
            <clip pos="1" side="tl" part="personne"/>
            <clip pos="1" side="tl" part="nombre"/>
          </lu>
        </out>
      </action>
    </rule>
```

do not present any new difficulty for understanding. We will send on output two lexical units each corresponding to the translation of the word, and to do this, we will use the new values of attributes we just modified.

Writing only once instructions common to several rules

After writing a rule for a personal pronoun followed by a verb, we will add two others for a noun (subject in the sentence) followed by a verb and for a determinant, followed by a noun (subject), followed by a verb.

A first innovation is that we will not only seek word groups (determinant, noun, verb, adjective, ...) but we add a constraint : the noun must belong to the subject of the sentence. In Esperanto, a noun used as the subject is not finished by letter *n* and in its analysis, we will find the **<nom>** (nominative) tag whereas for an object complement, we have the **<acc>** (accusative) tag.

In addition, the two new rules have something in common with the previous rule: we will have to make changes to the tense of the verb which is not written the same in all cases in Esperanto and French. But this change will be the same one in every rule including a conjugated verb. So, better is to write in one place and to use it as often as necessary. Besides saving code, a single copy will be easier to complete to add tenses for conditional and subjunctive or any other correction. When programming, we use *functions* to define pieces of codes used in several places of the program. For transfer rules, these are *macros*.

Define a word type with attributes

To define a noun having the attribute **<nom>** in its tags, we just have to add a category:

```
    <def-cat n="nom_sujet">
      <cat-item tags="n.*.nom"/>
    </def-cat>
```

The page A long introduction to transfer rules specifies that the .* when not placed at the end means "only one tag". This is the case for the analysis of most Esperanto nouns which do not have gender. However, it seems this definition also works with 2 tags between the **n** and the **<nom>**. Otherwise, at worst, for nouns having a gender (humans and animals), we could add a second **cat-item** :

```
      <cat-item tags="n.*.*.nom"/>
```

to specify 2 intermediate tags.

Writing a macro

Now, we will put inside a macro the operations necessary to the transfer of the tense of a verb. As it is our first macro, it will be necessary to create the **def-macros** section (which is an optional section) with the following contents:

```
  <section-def-macros>
    <def-macro n="set_temps" npar="1">    <!--  tenses concordance  -->
      <choose>
        <when>
          <test>
```

```
            <equal>
               <clip pos="1" side="sl" part="temps"/>
               <lit-tag v="pres"/>
             </equal>
          </test>
          <let>
            <clip pos="1" side="tl" part="temps"/>
            <lit-tag v="pri"/>
          </let>
        </when>
        <when>
          <test>
             <equal>
               <clip pos="1" side="sl" part="temps"/>
               <lit-tag v="past"/>
             </equal>
          </test>
          <let>
            <clip pos="1" side="tl" part="temps"/>
            <lit-tag v="pii"/>
          </let>
        </when>
      </choose>
    </def-macro>
  </section-def-macros>
```

The only true the innovation is the instruction: **<def-macro n="set_temps" npar="1">** :

It contains two informations:

Paramètre	Meaning
n="set_temps"	the name given to the macro
npar="1"	the number of parameters of the macro

Then, the code is identical to the one written for the rule personal pronoun + verb, except that in this rule, we specified **pos="2"** (the verb was the 2nd word of the pattern), whereas here, we have **pos="1"** which is the number of the parameter of the macro. And this macro only needs one parameter of verb type to work.

Transfer Rules using the macro

Thus let us see how the macro is used in the previous rule (changed) and the two new rules:

```
    <rule>
      <pattern>
        <pattern-item n="prn"/>
        <pattern-item n="verbe"/>
      </pattern>
```

```
      <action>
        <choose>    <!--  special cases for pronouns transfers  -->
          <when>    <!--  2nd person always plural : vi -> vous  -->
            <test>
               <equal>
                  <clip pos="1" side="sl" part="personne"/>
                  <lit-tag v="p2"/>
                </equal>
            </test>
            <let>
              <clip pos="1" side="tl" part="nombre"/>
              <lit-tag v="pl"/>
            </let>
          </when>
          <when>    <!--  3rd person plural always masculine : ili -> ils  -->
            <test>
               <and>
                 <equal>
                    <clip pos="1" side="sl" part="personne"/>
                    <lit-tag v="p3"/>
                  </equal>
                 <equal>
                    <clip pos="1" side="sl" part="nombre"/>
                    <lit-tag v="pl"/>
                  </equal>
                </and>
            </test>
            <let>
              <clip pos="1" side="tl" part="genre"/>
              <lit-tag v="m"/>
            </let>
          </when>
        </choose>

        <call-macro n="set_temps">
          <with-param pos="2"/>
        </call-macro>

        <out>
          <lu>
            <clip pos="1" side="tl" part="lem"/>
            <clip pos="1" side="tl" part="type_mot"/>
            <clip pos="1" side="tl" part="personne"/>
            <clip pos="1" side="tl" part="genre"/>
            <clip pos="1" side="tl" part="nombre"/>
          </lu>
          <b />
          <lu>
            <clip pos="2" side="tl" part="lem"/>
            <clip pos="2" side="tl" part="type_mot"/>
            <clip pos="2" side="tl" part="temps"/>
            <clip pos="1" side="tl" part="personne"/>
            <clip pos="1" side="tl" part="nombre"/>
          </lu>
        </out>
```

```
      </action>
    </rule>

    <rule>
      <pattern>
        <pattern-item n="nom_sujet"/>
        <pattern-item n="verbe"/>
      </pattern>

      <action>
        <call-macro n="set_temps">
          <with-param pos="2"/>
        </call-macro>

        <out>
          <lu>
            <lit v="un"/>
            <lit-tag v="det.ind"/>
            <clip pos="1" side="tl" part="genre"/>
            <clip pos="1" side="tl" part="nombre"/>
          </lu>
          <b />
          <lu>
            <clip pos="1" side="tl" part="lem"/>
            <clip pos="1" side="tl" part="type_mot"/>
            <clip pos="1" side="tl" part="genre"/>
            <clip pos="1" side="tl" part="nombre"/>
          </lu>
          <b />
          <lu>
            <clip pos="2" side="tl" part="lem"/>
            <clip pos="2" side="tl" part="type_mot"/>
            <clip pos="2" side="tl" part="temps"/>
            <lit-tag v="p3"/>
            <clip pos="1" side="tl" part="nombre"/>
          </lu>
        </out>
      </action>
    </rule>

    <rule>
      <pattern>
        <pattern-item n="det"/>
        <pattern-item n="nom_sujet"/>
        <pattern-item n="verbe"/>
      </pattern>

      <action>
        <call-macro n="set_temps">
          <with-param pos="3"/>
        </call-macro>

        <out>
          <lu>
            <clip pos="1" side="tl" part="lem"/>
```

```
            <clip pos="1" side="tl" part="type_mot"/>
            <lit-tag v="def"/>
            <clip pos="2" side="tl" part="genre"/>
            <clip pos="2" side="tl" part="nombre"/>
          </lu>
          <b />
          <lu>
            <clip pos="2" side="tl" part="lem"/>
            <clip pos="2" side="tl" part="type_mot"/>
            <clip pos="2" side="tl" part="genre"/>
            <clip pos="2" side="tl" part="nombre"/>
          </lu>
          <b />
          <lu>
            <clip pos="3" side="tl" part="lem"/>
            <clip pos="3" side="tl" part="type_mot"/>
            <clip pos="3" side="tl" part="temps"/>
            <lit-tag v="p3"/>
            <clip pos="2" side="tl" part="nombre"/>
          </lu>
        </out>
      </action>
    </rule>
```

In the two first rules corresponding to the following patterns:

```
      <pattern>
        <pattern-item n="prn"/>
        <pattern-item n="verbe"/>
      </pattern>
```

and

```
      <pattern>
        <pattern-item n="nom_sujet"/>
        <pattern-item n="verbe"/>
      </pattern>
```

we call the macro as follows:

```
        <call-macro n="set_temps">
          <with-param pos="2"/>
        </call-macro>
```

whereas for the last rule corresponding to the pattern:

```
      <pattern>
        <pattern-item n="det"/>
        <pattern-item n="nom_sujet"/>
        <pattern-item n="verbe"/>
      </pattern>
```

the macro call becomes:

```
        <call-macro n="set_temps">
          <with-param pos="3"/>
        </call-macro>
```

For each of the three cases, the value of **pos** of the tag **with-param** corresponds to the position of the verb in the pattern. Doing like that, we will send the macro all the information about the verb in the source language and the target language.

And if we wanted to make macro with several parameters, there would be as many **with-param** tags as parameters in the call for the new macro.

The rest of the two last transfer rules does not include a particular difficulty:

- we generate the analysis of a determinant which agrees with the noun
- then the one of the noun

as we did it in the rules without a verb.

Then, we generate the analysis of the verb, using the **temps** attribute updated in the macro. This verb is conjugated with the 3rd person with the number (singular or plural) of the subject noun in the sentence.

Using variables

To finish, we will examine a rule which requires to memorize a value into a variable.

This rule will translate a personal pronoun, followed by verb être (to be), followed by another verb to the past participle.

We already know how to process the pronoun followed by a verb, it was done in the paragraph Changing attributes according to conditions. It will remain to put in concordance the past participle with the personal pronoun. But there is a problem :

- with 1st and the 2nd person, the personal pronoun must have the gender **mf** (masculine/féminine) to be generated,
- for the past participle, the authorized genders are **m** and **f** (masculine or féminine, but only one of these).

Consequently, we will not be able to always use the same tag for the gender of the personal pronoun and the gender of the past participle. The idea to do that is to build the gender of the past participle from the one of the personal pronoun and to use a variable to memorize the result.

Calculation of the gender of the past participle is the following:

```
IF gender of pronoun = "mf" ALORS
    genre_pp <- "m"
ELSE
    genre_pp <- gender of pronoun
END IF
```

The variable which memorizes the gender of the past participle is called *genre_pp*. In the case of the personal pronoun used with 1st or 2nd person, it would be necessary to make a deep analysis to find (may be in a preceding sentence) the best gender to put the past participle in concordance. Apertium does not allow this kind of complex analysis. We will thus choose the masculine in this case. On the contrary, if the personal pronoun is used with the 3rd person, we will use its gender for the past participle.

A first thing to do is to declare the variable. For that, the **def-vars** section becomes :

```
  <section-def-vars>
    <def-var n="genre_pp"/>
  </section-def-vars>
```

We did not yet write any rule using the verb être (to be) conjugated or past participle. It will thus be necessary to complete the section **def-cats** by adding the two declarations :

```
    <def-cat n="etre_conj">
      <cat-item tags="vbser.pres"/>
      <cat-item tags="vbser.past"/>
      <cat-item tags="vbser.fti"/>
    </def-cat>

    <def-cat n="verbe_pp">
      <cat-item tags="vbser.pp.*"/>
      <cat-item tags="vblex.pp.*"/>
      <cat-item tags="vbtr.pp.*"/>
      <cat-item tags="vbntr.pp.*"/>
      <cat-item tags="vbtr_ntr.pp.*"/>
    </def-cat>
```

The rule doing the required work is the following :

```
    <rule>
      <pattern>
```

```
    <pattern-item n="prn"/>
    <pattern-item n="etre_conj"/>
    <pattern-item n="verbe_pp"/>
  </pattern>

  <action>
    <choose>    <!--  particular case for pronouns transfers  -->
      <when>    <!--  2nd person allways plural : vi -> vous  -->
        <test>
           <equal>
              <clip pos="1" side="sl" part="personne"/>
              <lit-tag v="p2"/>
            </equal>
        </test>
        <let>
          <clip pos="1" side="tl" part="nombre"/>
          <lit-tag v="pl"/>
        </let>
      </when>
      <when>    <!--  3rd person plural allways masculine : ili -> ils  -->
        <test>
           <and>
             <equal>
                <clip pos="1" side="sl" part="personne"/>
                <lit-tag v="p3"/>
              </equal>
              <equal>
                <clip pos="1" side="sl" part="nombre"/>
                <lit-tag v="pl"/>
              </equal>
            </and>
        </test>
        <let>
          <clip pos="1" side="tl" part="genre"/>
          <lit-tag v="m"/>
        </let>
      </when>
    </choose>

    <choose>    <!--  if gender of the pronoun is mf, gender of the past participle will be m  -->
      <when>
        <test>
           <equal>
              <clip pos="1" side="tl" part="genre"/>
              <lit-tag v="mf"/>
            </equal>
        </test>
        <let>
          <var n="genre_pp"/>
          <lit-tag v="m"/>
        </let>
      </when>
      <otherwise>
        <let>
          <var n="genre_pp"/>
```

```
            <clip pos="1" side="tl" part="genre"/>
          </let>
        </otherwise>
      </choose>

      <call-macro n="set_temps">
        <with-param pos="2"/>
      </call-macro>

      <out>
        <lu>
          <clip pos="1" side="tl" part="lem"/>
          <clip pos="1" side="tl" part="type_mot"/>
          <clip pos="1" side="tl" part="personne"/>
          <clip pos="1" side="tl" part="genre"/>
          <clip pos="1" side="tl" part="nombre"/>
        </lu>
        <b />
        <lu>
          <clip pos="2" side="tl" part="lem"/>
          <lit-tag v="vbser"/>
          <clip pos="2" side="tl" part="temps"/>
          <clip pos="1" side="tl" part="personne"/>
          <clip pos="1" side="tl" part="nombre"/>
        </lu>
        <b />
        <lu>
          <clip pos="3" side="tl" part="lem"/>
          <clip pos="3" side="tl" part="type_mot"/>
          <lit-tag v="pp"/>
          <var n="genre_pp"/>
          <clip pos="1" side="tl" part="nombre"/>
        </lu>
      </out>
    </action>
  </rule>
```

The really new part of the rule is this one :

```
  <choose>    <!--  if gender of the pronoun is mf, gender of the past participle will be m  -->
    <when>
      <test>
         <equal>
            <clip pos="1" side="tl" part="genre"/>
            <lit-tag v="mf"/>
          </equal>
      </test>
      <let>
        <var n="genre_pp"/>
        <lit-tag v="m"/>
      </let>
    </when>
    <otherwise>
```

```
        <let>
          <var n="genre_pp"/>
          <clip pos="1" side="tl" part="genre"/>
        </let>
      </otherwise>
    </choose>
```

It includes two assignments of values into the variable **genre_pp** :

```
        <let>
          <var n="genre_pp"/>
          <lit-tag v="m"/>
        </let>
```

allowing to put the tag **<m>** into **genre_pp**,

```
        <let>
          <var n="genre_pp"/>
          <clip pos="1" side="tl" part="genre"/>
        </let>
```

allowing to put the gender of the personal pronoun into **genre_pp**.

We can also notice that the conditional processing performed uses for the first time the tags **<otherwise>...</otherwise>** .

The last thing to do is to use the variable **genre_pp** to generate the lexical unit for the past participle :

```
    <lu>
      <clip pos="3" side="tl" part="lem"/>
      <clip pos="3" side="tl" part="type_mot"/>
      <lit-tag v="pp"/>
      <var n="genre_pp"/>
      <clip pos="1" side="tl" part="nombre"/>
    </lu>
```

It is the same instruction :

```
      <var n="genre_pp"/>
```

that allows to initialise the variable or to access the value it countains.
