Second Workshop on the Apertium free/open-source machine translation platform: transferring structures from one language to another.

Gema Ramírez-Sánchez

[1]Prompsit Language Engineering, S.L.,
Av. Universitat, s/n. Edifici Quorum III. 03202 Elx, Spain

22nd May 2015

Intro

Apertium

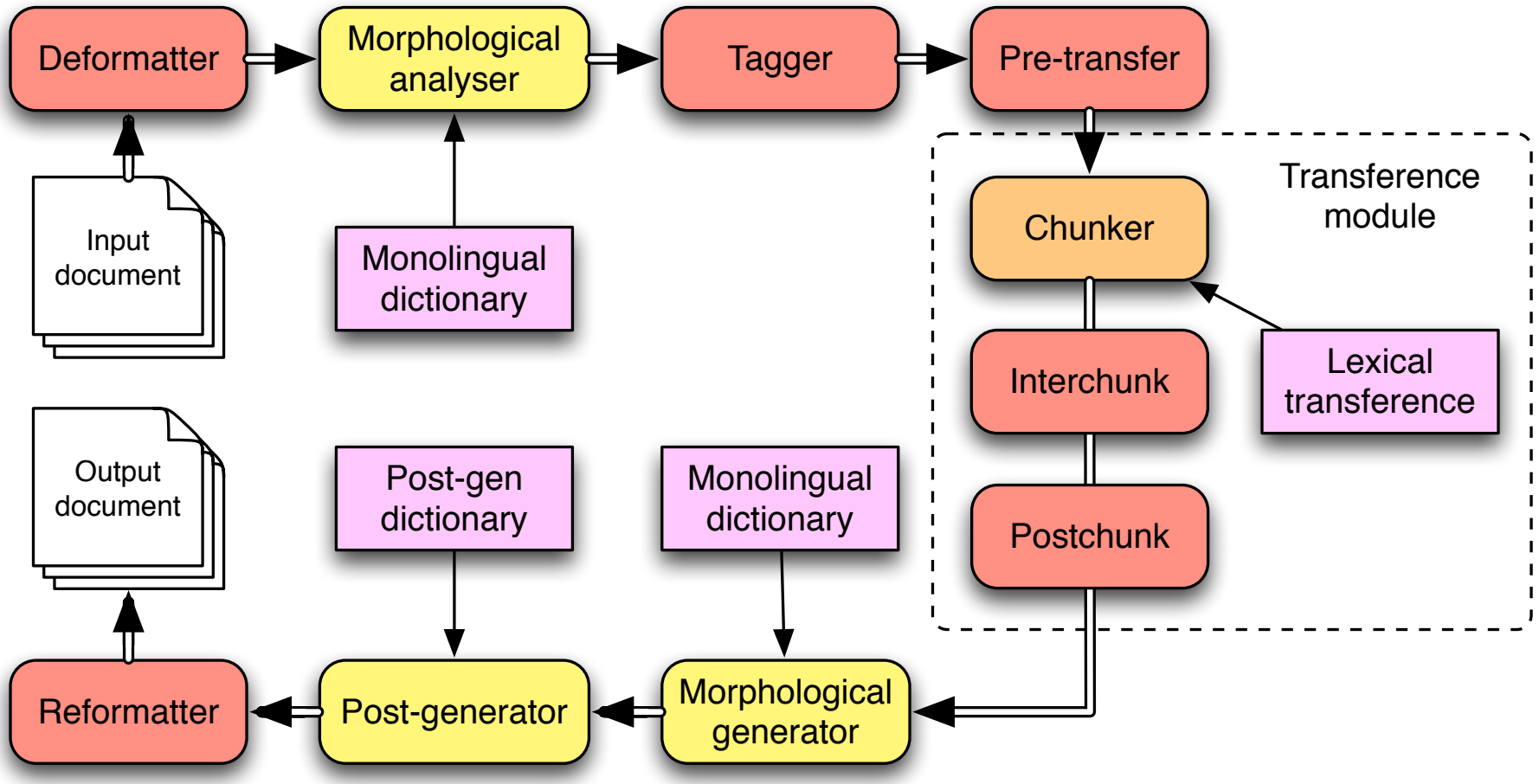
What is Apertium?

- Apertium je **platforma** za računalniško prevajanje. (sl)
- Apertium je **platforma** za računarsko prevođenje. (bs)
- Apertium je **platforma** za računalno prevođenje. (hr)
- Apertium je **platforma** za kompjutersko prevođenje. (sr)
- Apertium é um **sistema** de tradução automática. (pt)
- Apertium es un **sistema** de traducción automática.(es)
- Apertium ei un **sistèma** de traduccion automatica. (oc)
- Apertium és un **sistema** de traducció automàtica. (ca)
- Apertium este o **platformă** de traducere automată. (ro)

How does Apertium work, module by module [10 min.]

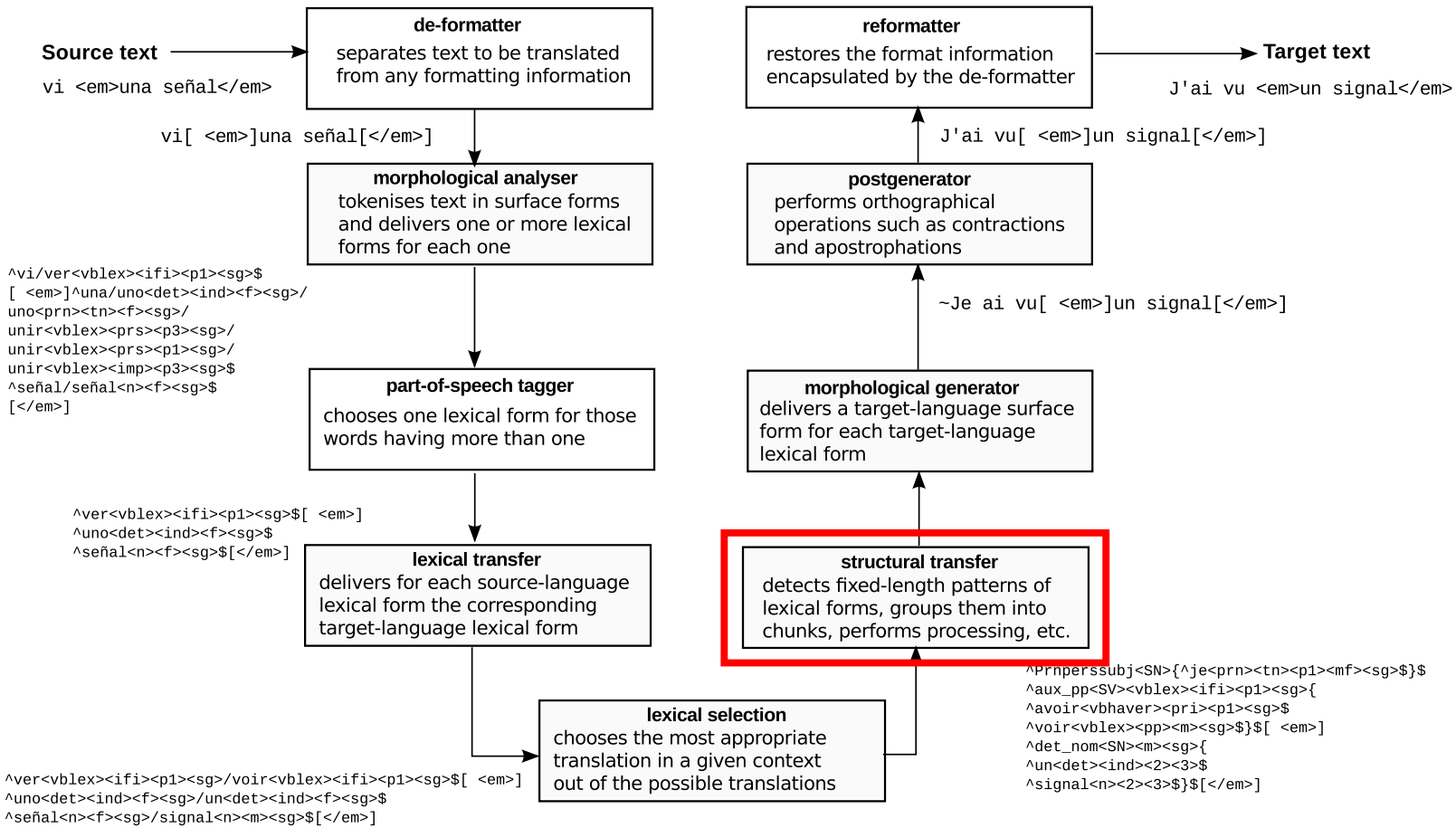
Apertium modules

- Classic shallow-transfer system
- Pipeline made by 8 independent modules:



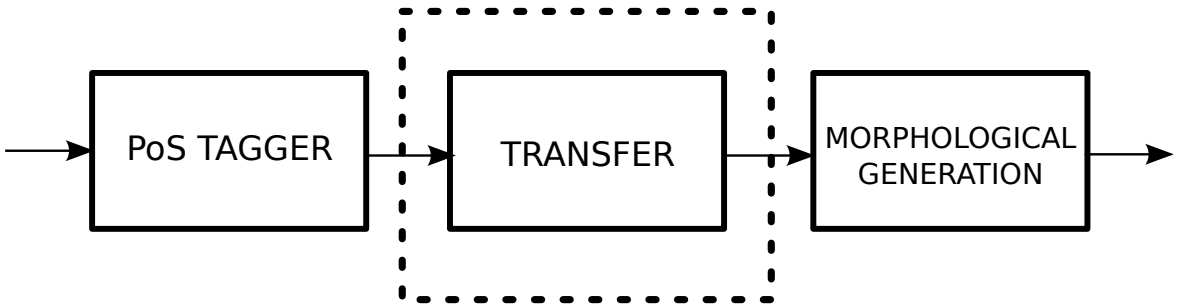
Apertium, module by module

- (es) Vi una señal
- (fr) J'ai vu un signal



Structural transfer in Apertium

Where are we in this workshop?



Practice 1: Taking a look to Apertium with apertium-viewer [30 min.]

Contrastive analysis for machines [10 min]

Contrastive analysis

Contrastive analysis is the process of examining two or more languages together to find out what kind of features they share, and how they are distinguished.

Morphological contrasts

We call morphological contrasts those differences between languages which are expressed at the level of a single word. Some examples might be:

- How possession is expressed: suffix, or separate word
- Case inventories
- Gender (or word class) inventories
- Number (in all words, or just in nouns/pronouns)
- "Synthetic" verb tenses
- Person inventory
- Formality
- Morpheme order (e.g. possessive before/after number)

Syntactic contrasts

Syntactic contrasts on the other hand are differences between words

- Existence or not of separate articles
- Existence or not of compulsory subject pronouns
- Word order (both within phrases, and between phrases)
- “Analytic” verb tenses

Contrastive analysis for machines

Apertium can handle easily systematic contrasts:

- local reorderings
- local agreement (gender, number, case)
- verb tenses
- numerical structures (dates, hours, etc.)

But is not able to make big changes: active to passive voice is out of scope.

Practice 2: Contrastive analysis for Serbian and Croatian [30 min.]

Formalising rules [10 min.]

Creating rules

- In Apertium, our intermediate representation is based on lemmas, parts-of-speech and morphological tags. Syntactic tags may also be used if available.
- So, when writing rules, they should be based on this level of intermediate representation

Operations

When thinking about implementing transfer rules, it is often worth trying to think in terms of some very basic operations such as:

- **Insertion**: Add a new word or tag,
  - e.g., debata će biti održana → **the** *debate will be held*
  - e.g., Sue<np><ant><f><sg> → Sue<np><ant><f><sg>**<nom>**
- **Deletion**: Removing a word or a tag,
  - e.g., give **it** → dati
  - e.g., debata <n>**<f>**<sg>**<nom>** → debate <n><sg>

The following operations are also basic operations, but can be considered to be permutations of the first two:

- **Substitution**: Replacing one tag/word, with another tag/word
  - e.g. vocative in Ukranian is replaced with nominative in Russian
- **Reordering**: Changing the order of words, or tags
  - e.g. the order of *adjective + verb* **to be** in Croatian to English: *odgovoran* **je** → **is** *responsible*

Combining operations

Most actual transfer rules will require more than a single operation. Consider the following example from English into Croatian. The rule:

- noun phrases acting as indirect object in English are formed by preposition *to* and the object often preceded by a determiner in English
- In Croatian, the object (a noun or pronoun) carries the dativ case to indicate that it is the indirect object of the sentence

Formalised:

- **Input**: *to the dog* = `to<pr> the<det><def><sp> dog<n><sg>`
  - (del) Remove preposition *to*
  - (del) Remove determiner *the*
  - (ins) Set the gender (ma), number (sg, from the determiner) and case (dativ, from the preposition) for the noun, dative in this case
- **Output**: `pas<n><ma><sg><dat>` = *psu*

Practice 3: Formalising Serbian to Croatian rules [30 min.]

Structural transfer in Apertium: 20 min.

The structural transfer module

Two types of transfer systems:

- single level transfer:

  Rule file: *.t1x

- multiple level transfer (3+):

  Rule files: *.t1x *.t2x *.t3x

Single level transfer: how does it work

- Rules detect patterns of words (generally specified as parts of speech):
  - DET NOUN
    The table
  - DET ADJ NOUN
    The black table
- Transformations are applied to these patterns:
  `EN` $\rightarrow$ `ES`

DET ADJ NOUN	$\rightarrow$ DET NOUN ADJ
The black table	$\rightarrow$ *La mesa negra*
sp sp sg	f.sg f.sg f.sg
The black book	$\rightarrow$ *El libro negro*
sp sp sg	m.sg m.sg m.sg

Single level transfer: how does it work /2

Some transformations:

- syntactic/structural changes: *They like dancing* (en) → *Les gusta bailar* (es)
- reorderings: *an interesting book* (en) → *un libro interesante* (es)
- agreement: *le vélo rouge* (fr) → *la bicicleta roja* (es)
- lexical changes: *They are tired* (en) → *Están cansados* (es)

How the rules work

- words can only be processed by ONE rule: rules can not be overlapped
- patterns are matched following the LRLM principle (left-to-right-longest match)
  Example:
  - Rule 1: DET NOUN
  - Rule 2: NOUN ADJ CONJ ADJ
  - Input text (fr): *Le vélo rouge et vert*

  Which rule(s) will be applied?

How the rules work

- words can only be processed by ONE rule: rules can not be overlapped
- patterns are matched following the LRLM principle (left-to-right-longest match)
  Example:
  - Rule 1: DET NOUN
  - Rule 2: NOUN ADJ CONJ ADJ
  - Input text (fr): *Le vélo rouge et vert*

  Which rule(s) will be applied?

Le	vélo	rouge	et	vert
Det	Noun	Adj	Conj	Adj

How the rules work

- words can only be processed by ONE rule: rules can not be overlapped
- patterns are matched following the LRLM principle (left-to-right-longest match)
  Example:
  - Rule 1: DET NOUN
  - Rule 2: NOUN ADJ CONJ ADJ
  - Input text (fr): *Le vélo rouge et vert*

  Which rule(s) will be applied?

	Le	*vélo*	*rouge*	*et*	*vert*
	Det	Noun	Adj	Conj	Adj
left-to-right	DET	NOUN			

Structure of the rule file

Structure of the rule file:

- `<section-def-cats>`
- `<section-def-attrs>`
- `<section-def-vars>`
- `<section-def-lists>`
- `<section-def-macros>`
- `<section-def-rules>`

Structure of the rule file /2

- `<section-def-cats>` → definition of categories. Categories will be used to define the pattern detected by a rule.

```
<def-cat n="noun">
  <cat-item tags="n.*">
</def-cat>
<def-cat n="nounplural">
   <cat-item tags="n.pl"/>
</def-cat>
```

Categories can be lexicalised:

```
<def-cat n="ago">
   <cat-item lemma="ago" tags="adv"/>
</def-cat>
```

Structure of the rule file /3

- `<section-def-attrs>` → definition of attributes. Attributes are possible values of a certain feature.

```
<def-attr n="nbr">
  <attr-item tags="sg">
  <attr-item tags="pl">
  <attr-item tags="sp"/>
  <attr-item tags="ND"/>
</def-attr>

<def-attr n="gen">
  <attr-item tags="m">
  <attr-item tags="f">
  <attr-item tags="mf">
  <attr-item tags="GD">
</def-attr>
```

- `<section-def-vars>` → definition of variables.

Structure of the rule file /4

- `<section-def-lists>` → definition of lists of lemmas which will be used inside of the rules.

```
<def-list n="days">
   <list-item v="Monday"/>
   <list-item v="Tuesday"/>
   <list-item v="Wednesday"/>
(...)
```

- `<section-def-macros>` → definition of macroinstructions (parts of code to be reused in the rules).
- `<section-def-rules>` → definition of the transfer rules.

Structure of a rule

Rule = <pattern> + <action>

- Pattern: defined using the categories of the <section-def-cats>

```
<rule comment="REGLA: DET NOUN">
  <pattern>
   <pattern-item n="det"/>
   <pattern-item n="noun"/>
  </pattern>
(...)
```

- Action:
  - specifies the tests and transformations to be made on the detected pattern
  - outputs the resulting lexical forms in the desired order

Description of some tags

Some tags used in the <action> part of the rules:

- <choose>: start a test
  - <when>: if the conditions specified are met...
    - <test>: the condition to be tested
    - <otherwise>: if the condition is not met...
- <let>: assign a value
- <call-macro>: call a macroinstruction defined in <section-def-macros>
- <out>: output the lexical units
  - <lu>: lexical unit, specified by means of one or more <clip>
- <clip>: the basic unit of a lexical form.

```
<clip pos="2" side="tl" part="lem"/>
<clip pos="2" side="tl" part="temps"/>
<clip pos="2" side="tl" part="whole"/>
```

The clip element

Given the lexical form:

^take# into account<vblex><pres><p3><sg>$

- `<clip pos="1" side="sl" part="whole"/>` : whole content of lexical unit
- `<clip pos="1" side="sl" part="lem"/>` : lemma: *takes# into account*
- `<clip pos="1" side="sl" part="tags"/>` : `<vblex><pres><p3><sg>`
- `<clip pos="1" side="sl" part="lemh"/>` : lemma head : *takes*
- `<clip pos="1" side="sl" part="lemq"/>` : lemma queue : *# into account*

Example rule: pattern

```
<rule comment="Place clitic between adjective and noun">
 <pattern>
   <pattern-item n="CAT__clt_"/>
   <pattern-item n="CAT__adj_"/>
   <pattern-item n="CAT__n_"/>
 </pattern>
 <action>
   <choose>
    [...]
    </choose>
      <out>
        [..]
     </out>
 </action>
</rule>
```

Example rule: action>conditions

```
<rule comment="Place clitic between adjective and noun">
  <pattern>[...]</pattern>
  <action>
    <choose>
      <when><!--Check noun gender in source and target-->
        <test>
          <not>
            <equal>
              <clip pos="3" side="sl" part="gender"/>
              <clip pos="3" side="tl" part="gender"/>
             </equal>
           </not>
        <let> <!--Gender agreement for noun and ajective-->
          <clip pos="2" side="tl" part="gender"/>
          <clip pos="3" side="tl" part="gender"/>
        </let>
        </test>
      </when>
    </choose>
      [...]
```

Example rule: action>output

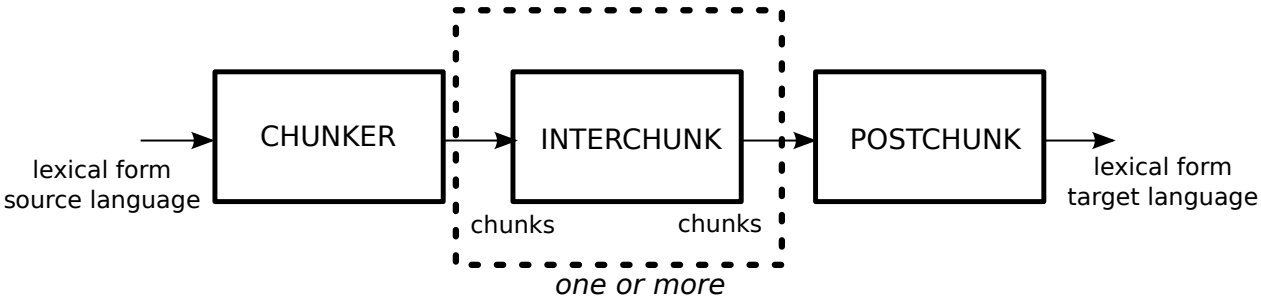
```
<rule comment="Place clitic between adjective and noun">
  <pattern>[...]</pattern>
  <action>
    [...]
    <out>
      <lu>
        <clip pos="2" side="tl" part="whole"/>
      </lu>
      <b pos="1"/>
      <lu>
        <clip pos="1" side="tl" part="whole"/>
      </lu>
      <b pos="2"/>
      <lu>
        <clip pos="3" side="tl" part="whole"/>
      </lu>
    </out>
  </action>
</rule>
```

Practice 4: Identifying rules and evaluating them [30 min.]

Multiple transfer level in Apertium: 30 min.

Multiple transfer level

Typically, we have 3 modules



t1x	t2x	t3x
LRLM	LRLM	
patterns	patterns of chunks	chunks chunks

Chunker and interchunk have the same file structure. Chunker is like the transfer module in the single level transfer. Main difference:

- Transfer outputs lexical units
- Chunker outputs chunks, to be processed by the interchunk

Chunks

Chunks = group of words that correspond roughly to a phrase.

Input pattern	**Example**	**Output chunk**	**Example**
nom	*car*	SN{nom}	auto
det nom	*the car*	SN{nom}	auto
num nom	*three cars*	SN{num nom}	tri automobila
adj nom	*red car*	SN{adj nom}	crveni auto
verb	*she is*	V{verb}	ona je
verb	*she isn't*	V{neg_adv verb}	ona nije
prn adv verb	*she just said*	V{prn verb adv pp}	ona je upravo rekla
prn aux inf	*she will cook*	V{prn aux verb}	ona će kuhati

The creation of chunks allows for a more powerful treatment of syntactic/structural divergences:

Example

interchunk rule:	NP	V
possible chunks:	noun	verb
	det noun	aux verb
	det adj noun	adv verb
	adj noun	aux adv verb
	adj adj noun	aux aux verb
	adj conj adj noun	...
	...	

The chunker

The **chunker** is like the **transfer** module of the single-level transfer, being the main difference:

- The transfer outputs lexical units
- The chunker outputs chunks

Structure of a chunk:

- Chunk = chunkname<tags>{chunk_content}
- In the chunker, lexical units are translated into the target language

"The teacher" (en → es)

```
^Det_noun<SN><DET><GD><sg>{^el<det><def><3><4>$
^profesor<n><3><4>$}$
```

The interchunk

The interchunk module:

- detects patterns of chunks in the same way that the chunker detects patterns of words
- applies the necessary transformations to these patterns: agreement, reorderings, syntactic or morphological changes

Example output:

"The teacher" (en → es)

```
^Det_noun<SN><DET><m><sg>{^el<det><def><3><4>$
^profesor<n><3><4>$}$
```

The main difference with the chunker rules: the <clip> element in the rules does not have a *side* attribute:

- chunker: `<clip pos="2" side="tl" part="lem"/>`
- interchunk: `<clip pos="2" part="lem"/>`

The interchunk /2

Examples of interchunk operations in `en-es`:

- Agreement SN-SA:
  - *The idea is not bad* → *La idea no es mala*
  - Rules: SN Vcop SA, SN Vcop ADV SA...
- Agreement SN-SN:
  - *Maria is the teacher* → *Maria es la profesora*
  - Rules: SN Vcop SN, SN Vcop ADV SN...
- Agreement SN-SV:
  - *The man came* → *El hombre vino*
  - *The men came* → *Los hombres vinieron*
  - Rules: SN SV, SN ADV SV, SN prep SN SV...
- Deletion of subject pronouns:
  - *They don't care* → *No se preocupan*
- Syntactic changes:
  - *They like cakes* → *Les gustan los pasteles*
- Generation of articles:
  - *Cats are intelligent* → *Los gatos son inteligentes*

The interchunk /3

```
<rule comment="REGLA: SN ser SN">
  <pattern>
    <pattern-item n="SN"/>
    <pattern-item n="vbcop"/>
    <pattern-item n="SN"/>
  </pattern>
  <action>
  (...)
   <out>
    <chunk>
     <clip pos="1" part="whole"/>
    </chunk>
    <b pos="1"/>
    <chunk>
     <clip pos="2" part="whole"/>
    </chunk>
(...)
```

The postchunk

The postchunk module:

- substitutes numbers in tags by the referenced values
- removes the chunk lemma and tags
- outputs the sequence of lexical units
- can operate only on a single chunk at a time
- can perform some final operations on the chunk content

Example output:

"The teacher" (en → es)

```
^el<det><def><m><sg>$ ^profesor<n><m><sg>$
```

Looking at three-stage transfer

Input text: *"Spiders are beautiful"*

apertium -d . en-es-pretransfer

```
^Spider<n><pl>$ ^be<vbser><pres>$ ^beautiful<adj>$
```

apertium -d . en-es-chunker

```
^Nom<SN><UNDET><f><pl>{^araña<n><3><4>$}$
^be<Vcop><vbser><pri><PD><ND>{^ser<vbser><3><4><5>$}$
^adj<SA><GD><ND>{^bonito<adj><2><3>$}$
```

apertium -d . en-es-interchunk

```
^Nom<SN><PDET><f><pl>{^araña<n><3><4>$}$
^be<Vcop><vbser><pri><PD><pl>{^ser<vbser><3><4><5>$}$
^adj<SA><f><pl>{^bonito<adj><2><3>$}$
```

apertium -d . en-es-postchunk

```
^El<det><def><f><pl>$ ^araña<n><f><pl>$
^ser<vbser><pri><p3><pl>$ ^bonito<adj><f><pl>$
```

Practice 5: Inspecting the multiple-level transfer [30 min.]

Hvala!