Dictionary Encoding Based on CSS and XML/HTML Parsers

1. Source File Structure

The source file has been obtained as a result of pdf-to-doc-to-odt-to-xhtml conversion. Resulting xhtml file preserved almost all typographic features except correct column rendering. The reason of the conversion is to perform annotation on the text's source level using OHCO model, not as we did previously in WYSIWYG editors. CSS and XHTML parsing is based on the following tag hierarchy:

Listing 1. Object hierarchy of the source file

```
<html>
        <head>
        <! - - … - - >
        <body>
                <div>
                        <p>+
                                <span>text</>+
                        <div>+
                                <p>+
                                        <span>text</>+
```

2. Parser Output Structure

Listing 2. Outline of the structure in XHTML

```
<dict>
        <entry>+
                <form>
                        <orth>
                        <formt type ={“variant”|”inflected”|”compound”}>?
                                <orth>
                <senseList>
                        <sense n={“1.”|”2.”|”3.”|”4.”}>
                                <transList>
                                        <trans>+
                                <example>?
                                        <form>
                                        <sense>
                <subentryList>?
                        <subentry>+
                                <form>
                                <sense
```

Listing 3. RELAX NG version of schema

1. dict_p = element dict{(entry_p)+}
2. entry_p = element entry{form_p, senseList_p, subEntryList_p?}
3. form_p = element form{orth_p, element form {attribute type{“variant”|”inflected”|”compound”},orth_p}?}
4. orth_p = element orth{text}
5. senseList_p = element senseList{sense_p+}
6. sense_p = element sense{translation_p+, element example{form_p, sense_p}?}
7. translation_p =element translation{text}
8. subEntry_p = element subEntry{form_p, senseList_p}

3. Interpretation of Text Features

Lexicographic structure of the dictionary is presented by means of typography (font features, layout) and syntax (predefined indicators and punctuation). Here below we provide some interpretation rules, which we used for parsing, in [text feature] : [interpretation] format:

Listing 4. Dictionary interpretation rules

1. Font Features
   - 1.1. [bold] AND [NOT[enumeration, reference]]: [form element]
   - 1.2. [non-bold] AND [non-italic]: [sense element]
   - 1.3. [non-bold italic] AND [Latin Encoding]: [international names of flora and fauna]
2. Indicators/Punctuation
   - 2.1. [‘1.’, ‘2.’ etc.]: [entry sense number]
   - 2.2. [‘1)’, ‘2)’ etc.]: [sub-entry sense number]
   - 2.3. [Rome Digit]: [homograph is to the left]
   - 2.4. [‘:’]: [delimits <orth> element from it’s compound form]
   - 2.5. [‘~’]: [headword placeholder]=[<oRef/> in TEI]
   - 2.6. [‘, -’]: [delimits <orth> element from it’s inflected form]
   - 2.7. [’,’ between bold words]: [delimits <orth> element from it’s variant form]
   - 2.8. [non-latin text between parenthesis]:[definition of sense] OR [context] OR [directing information] etc.

4. Parsing Workflow

The parser pursues a simple schema to provide clear logic and minimum complexity.

The main principle: at first some structural tokens are defined. After that key tokens are used to identify desirable elements or their boundaries.

Listing 5. The workflow steps

1. Retrieve lexical data from the source file
2. Markup dictionary entries
3. Markup <form> and <sense> elements
4. Markup <orth> element and text to the right of it as <form> with @type attribute
5. Markup <senseList/> and <subEntryList/> elements
6. Markup sense items inside <senseList>
7. Markup and move sense examples, if so, into <sense> constructions

8. Markup all sub-entries inside <subEntryList>
9. After general schema is achieved indicated(<lbl>, <usg>, <def> etc.) constructions should be marked up and moved, if needed, to appropriate elements.

5. Structure According to TEI P5

Listing 6. Outline of the structure in TEI P5

```
<dict>
        <superEntry>?
        <entry>+
                <form>
                        <orth>
                        <formt type ={“variant”|”inflected”|”compound”}>?
                                <orth>
                <sense>
                        <sense n={“1.”|”2.”|”3.”|”4.”}>+
                                <cit type = “translation”>
                                        <quote>+
                                        <cit type = “example”>?
                                                <quote>
                                                <cit type = “translation”>
                                                        <quote>
                <re>*
                        <form>
                                <orth>
                        <sense n={“1)”|”2)”|”3)”|”4)”}>+
                                <cit type = “translation”>
                                        <quote>+
```

Listing 7. RELAX NG version of structure XML/TEI (<xr>, <usg>, <lbl> and <def> elements are not shown).

```
dict_p = element dict{(superEntry_p | entry_p)+}
        superEntry_p = element superEntry{entry_p+}
        entry_p = element entry{form_p, sense_1, re_p*}
                form_p = element form{orth_p, element form{attribute att.type
                {”compound”|”inflected”|”variant”}, orth_p}?}
                        orth_p = element orth{text}
                sense_1 = element sense{sense_2+}
                        sense_2 = element sense{attribute att.n{“1.”|”2.”|”3.”|”5.”}, cit_1}
                                cit_1 = element cit{attribute att.type{“translation”}, quote_1+,
                                cit_ex?}
                                        quote_1 = element quote {text}
```

	cit_ex = element cit{attribute att.type{“example”},quote_2, cit_2}
	quote_2 = element quote{text}
	cit_2 = element cit{attribute att.type{“translation”}, quote_3}
	quote_3 = element quote{text}
	re_p = element re{element form{orth_p}, sense_3}
	orth_p =element orth{text}
	sense_3 = element sense{sense_4+}
	sense_4 = element sense{attribute att.n{“1)”|”2)”|”3)”|”5)”}, cit_3}
	cit_3 = element cit{attribute att.type{“translation”}, quote_4+}
	quote_4 = element quote{text}

6. Mapping Parser Output to TEI P5

Resuming the structure of the dictionary it can be said that it’s recursive. Generally saying a main entry contains a list of senses and after it an optional list of recursive entries which we called sub-entries. The only structural difference between main entry and sub-entry that senses of latter cannot contain examples. Listing 2. and Listing 6. illustrate that sense elements are presented differently: TEI P5 use <cit> elements and our XHTML doesn’t. Additionally our shema doesn’t deal with homographs.