A Procedural DTD Project for Dictionary Entry Parsing Described with Parameterized Grammars

Neculai CURTEANU[1] *Alex MORUZ*[1,2]
(1) INSTITUTE of COMPUTER SCIENCE, ROMANIAN ACADEMY, IAŞI Branch;
(2) FACULTY of COMPUTER SCIENCE, UNIV. "AL. I. CUZA", IAŞI, ROMANIA
ncurteanu@yahoo.com, mmoruz@info.uaic.ro

ABSTRACT

The present paper continues the successful parsing experiments with the method of *Segmentation-Cohesion-Dependency* (SCD) *configurations*, a breadth-first, formal grammar-free, and optimal approach to dictionary entry parsing, proposed in the previous CogALex Workshops and applied to the following *five* very large thesaurus-dictionaries: **DLR** (The Romanian Thesaurus – new format), **DAR** (The Romanian Thesaurus – old format), **TLF** (Le Trésor de la Langue Française), **DWB** (Deutsches Wörterbuch – GRIMM), and **GWB** (Göthe-Wörterbuch). In this work we report new results: **(a)** The lexicographic modeling and parsing experiments of the *sixth* large **DMLRL** (Dictionary of Modern Literary Russian Language); **(b)** Outlining the *Enumeration Closing Condition* (ECC) for solving the recursive calls between sense marker classes situated on different nodes of a sense dependency hypergraph (SCD-configuration, *i.e.* parsing level); **(c)** The central result we report here is the project of a new, *procedural* DTD (Document Type Description) for dictionaries, based on the formalization of the SCD parsing method, providing *parameterized grammars* to describe the dependency hypergraphs that correspond to the main parsing levels in a dictionary entry. Here we give two parameterized grammars for **DLR**, as a small sample from a larger package of combined grammars for the above mentioned dictionaries. This package is constructed as the "least common multiple" of the parameterized grammars written for the parsed dictionaries; it represents the DTD description of a general *parser* for large dictionary entries, and thoroughly extends the current DTD in the XCES TEI P5 standard.

KEYWORDS: SCD dictionary parsing method; procedural DTD for dictionary entry parsing.

1 Lexicographic Modeling of DMLRL

A pre-processing parsing stage can be added to the SCD (*Segmentation-Cohesion-Dependency*) *configurations* for **DMLRL** *homonymic entries*, which are discriminated by indexing each of the homonyms with Arabic numerals followed by dot, all in *Arial font*, *Regular* and *Bold* format. These indexes are positioned in front of each homonym-word lemma, enumerating increasingly all the homonyms of the same word-lemma. An example of *four* homonymic entries of the word "**БЫЧОК**" is present in (DMLRL, :860-861), exposed in (Curteanu et al., 2012a :45).

The *first SCD configuration* has to recognize the *lexicographic segments* of a **DMLRL** entry. **DMLRL** comprises (at least) five types of lexicographic packages / segments (Curteanu et al., 2012a): **(1)** a *morpho-lexical* package / segment; **(2)** the *sense description* segment; **(3)** a ***TildaDef*** *package* or *segment of definitions*; **(4)** the *morpho-syntactic variant* segment; and **(5)** the *etymology* segment of the word-lemma. The morpho-lexical definition package is obligatorily present at the beginning of each entry, immediately after the word-lemma. The morpho-lexical package may occur also at the sense lower-levels of the entry sense tree. The ***TildaDef*** package can be attributed not only to any (sub)sense description level of the entry but also to the root-sense (zero-level sense hierarchy), when this package / segment begins at *New_Paragraph*.

The *primary sense* markers in **DMLRL** pointed out so far by the lexicographic analysis are: Latin capital numerals followed by a dot (**I.**, **II.**, **III.**,... etc.), in bold (*LatCapNumb_Mark*), and Arabic numerals followed by a dot (**1.**, **2.**, **3.**,... etc.), in bold (*ArabNumb_Mark*). The markers of these classes are positioned at the beginning of the text row, in fact, at *New_Paragraph* (*NewPrg*) marker, except for the *first sense markers* (**I.**, **1.**), which usually do not occur at *NewPrg*.

The sense markers of the class denoting Latin capital numerals followed by a dot (**I.**, **II.**, **III.**,...etc. or simply, *LatCapLet_Enum*) represent the top of the sense hierarchy in **DMLRL**. These markers establish the lexicographic limits for the *most general senses* of the word-lemma. They are the lexical-semantic equivalent of the sense marker class containing bolded Latin capital letters **A.**, **B.**, etc. (abbreviated as *LatCapLet_Enum*) in **DLR** (Curteanu et al., 2008).

The sense marker class of *Arabic numerals* followed by dot (**1.**, **2.**, **3.**,... etc.), in bold (*ArabNumb_Enum*), stands for the second level of primary sense representation in **DMLRL**. The place of these two sense marker classes is displayed within the left side hypergraph of Fig. 1. The sense marker classes *LatCapNumb_Enum* and *ArabNumb_Enum* are considered as **DMLRL** *primary senses*, similarly to **DLR-DAR** lexicographic modeling (Curteanu et al., 2010).

We placed the *two-oblique-bars* ”//” sense marker, which is specific to **DMLRL**, on the *third level* of the hierarchical dependency structure of **DMLRL** senses. At the same time, the sense marker ”//” is considered to be the first element of the two-markers set {//, ◊} denoting the *secondary senses* in **DMLRL**. The sense marked by ”//” is in lexical-semantics subordination to (or subsumed by) any other primary sense marked by an element in the marker classes {*LatCapNumb_Enum*, *ArabNumb_Enum*}, when they exist in the entry text. Otherwise (when a primary super-ordinated sense is missing), the secondary sense marker ”//” may occur immediately under the topmost level of the **DMLRL** sense hierarchy. The marker ”//” is embodied explicitly into the entry text, even for the case when this sense level is unique.

We notice that *autonomous* definitions in the //-marked subsenses to the primary senses can be refined by the so-called *DictExem*, *i.e. examples-to-definitions* given by **DMLRL** authors. Usually, *DictExems* are separated from *DefExems* that follows through the **DMLRL**-specific marker ”□” called *traverse*. By analogy with the **DLR** hypergraph of sense dependencies, we associate the **DMLRL** ”//” marker with the **DLR** ”♦” sense marker: they are both secondary sense markers and subsume the similar secondary sense marker denoted in both dictionaries by the *empty-diamond* ”◊” (DMLRL, 1994), (Curteanu et al., 2012a, 2010).

The problem of literal enumeration in **DMLRL** is a challenging problem because one may find entry samples that display a recursion between the *literal enumeration* and the *secondary senses* “//” and ”◊” (at least these markers), a typical sample of this situation being the entry **БЫ** (DMLRL, :844). The same type of recursion can occur actually between primary, secondary, and atomic senses, on one hand, and literal enumeration senses, on the other hand. The solution of reducing these recursions to a finite number of cycles should be the consistent control of the monotonic and sound closing of the literal enumeration development on higher or lower levels of the **DMLRL** pre-established hypergraphs of sense marker class dependencies (Fig. 1 below) (Curteanu et al., 2012a, 2012c).

The reverse situation, of the sense levels that could refine the *literal enumeration* sense description is illustrated by the thick-tail arrows, oriented upwards and intersecting the thin-tail arrows, in the first dependency hypergraph, SCD-config2 parsing level (Fig. 1).

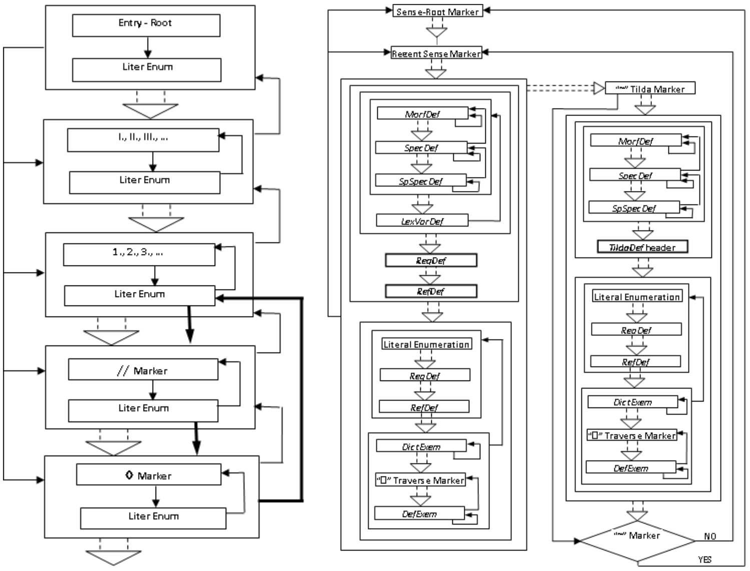


FIGURE 1 – The first dependency hypergraph (SCD-config2 of primary and secondary senses) calling the second dependency hypergraph (SCD-config3 of atomic senses) in **DMLRL**

The lower-level parsing (SCD-config3) is represented in the right side hypergraph of Fig. 1, for the *atomic* sense / definition markers in **DMLRL**. This hypergraph is *interconnected* with the higher-sense parsing level represented by the left side (*i.e.* SCD-config2) hypergraph in Fig. 1. This SCD-config2 hypergraph gives the dependency relationships among the higher-order sense marker classes, handing down from the root-sense, through primary and secondary senses, to the lower and atomic senses / definitions. When structurally accomplished, **DMLRL** lower-level senses are raising up, called by higher-level sense markers, until the structure of the entry sense tree is completed.

2 The *Enumeration Closing Condition*: A Solution to Preserve the Soundness of Sense Structure Definitions

The *Enumeration Closing Condition* (ECC) represents a deterministic, computational constraint devoted to check the sound termination (*i.e.* in a deterministic, finite number of steps) of the literal or numeral enumeration marker list, when higher-level sense markers break into the list. When this happens, contextual look-ahead verifications are needed to obtain the correct closing of the enumeration list. More precisely, ECC means that whether after a certain (let us say, *current*) *letter* or *numeral* in the sense enumeration marker list occur higher-level sense markers (on the dependency hypergraph), then one should look forward in the sense marker sequence until the *next* letter or numeral of the same enumeration type occurs. If such an item does exist and follows *monotonously* (*i.e.* in lexicographic ordering) the current one in the enumeration list, then the enumeration should continue. Otherwise, thus if the (letter or numeral) item does not exist or it begins another enumeration, of the same or higher dependency level as the current one, then the ECC holds and the literal enumeration must be closed. For instance, in the Romanian **DLR**, with the filled and empty diamonds ♦, ◊ as secondary sense markers, the enumeration list **a) b) c)** ◊ ♦ ◊ ◊♦ ◊ **d)…** should continue, while the marker sequence **a) b) c)** ◊ ♦ ◊ ◊♦ ◊ **a)…** should close the first literal enumeration (Curteanu et al., 2012a, 2012c). The same is true if non-enumerable sense markers (such as ♦, ◊ in **DLR**) are replaced by another enumeration of sense markers, be it of another numeral or literal type. Two different enumerations, a standard, *literal* one, and a *numeral* one coming from transforming the *New_Paragraphs* into sense markers, are illustrated by the following special entries bearing recursive-calls: "**CAL**" in **DAR**, "**LUMÍNĂ**" in **DLR**, "**БЫ**" in **DMLRL** (DMLRL :844) (Curteanu et al., 2012c, 2012a).

Parsing with SCD configurations, we discovered the special role that the *New_Paragraph* (*NewPrg*) typographic marker is playing in the disambiguation process, either for the lexicographic segment recognition or to ECC verification. For an efficient use of *NewPrg*(*s*) as lexicographic markers, in a preprocessing phase for parsing a dictionary entry, we decided to transform *all* the *NewPrg*(*s*) occurring in the entry into *Latin small numerals* (*LatSmaNumb*) as lexicographic segment markers or sense enumeration markers.

3 Parsing Experiments with DMLRL Entries on the SCD-config2 Level

We outline here the parsing results on the six thesauri: **DLR**, completely parsed (175,000 entries; 15,000 pages; 37 volumes) at 98.01% accuracy, **DAR** completely parsed (25,000 entries, 3000 pages, 5 volumes), but no gold standard was available for automatic evaluation, while for **TLF**, **DWB**, **GWB**, and **DMLRL**, around 50 significant (including very large) entries have been parsed with very sound outcomes but not gold standard available for the parsing evaluation. Also, using ECC, we solved the following two difficult parsing problems, met not only in **DMLRL**, but also in **DLR**, **DAR** (as specified above), and other thesauri: (*a*) sense dependencies of the SCD *second configuration* (left side hypergraph in Fig. 1), and (*b*) the mutual calling between literal enumeration and secondary senses. For 50 **DMLRL** entries (of all sizes, including very large ones), the parser provided a really sound parsing percentage, at this level (Curteanu et al., 2012b).

The special entry **БЫ** (DMLRL, :844) contains the marker subsequence "3. а) ◊ ◊ // ◊ ◊ б) ◊ ◊ ◊ в) г) ◊ ◊", which shows (in the partial excerpt below) the occurrence of **DMLRL** secondary senses under the literal enumeration, whose sound parsing is based on ECC to hold (Curteanu et al. (2012a, 2012c).

БЫ (сокращенно **Б**), *частица.* В сочетании с глаголами в форме прошедшего времени образует сослагательное наклонение. **1.** Употр. для обозначения предположительной …

… …

3. Обозначает различные оттенки желаемости действия; **а)** Собственно желаемость. *Учился бы сын. Были бы дети здоровы.* ◊ Если бы, когда бы, хоть бы и т. п. О, *если бы когда-нибудь Сбылась поэта сновиденья!* Пушк. Посл. к Юдину. [Николка:] *Хоть бы дивизион наш был скорее готов.* Булгаков, Дни Турб. ◊ С неопр. ф. глаг. *Полететь бы пташечке К синю морю; Убежать бы молодцу в лес дремучий.* Дельв. Пела, пела пташечка.. [Настя:] *Ах, тетенька, голубок! Вот бы поймать!* А. Остр.

Не было ни гроша…— *Жара, дедушка Лодыжкин .. Нет никакого терпения! Искупаться бы!* Купр. Бел. пудель. / / Употр. для выражения опасения по поводу какого-л. нежелательного действия (с отрицанием). *Не заболел бы он.* ◊ С неопр. ф. глаг., имеющей перед собой отрицание. — *Гляди, — говорю, — бабочка, не кусать бы тебе локтя! Так-таки оно все на мое вышло.* Леск. Воительница. ◊ Только бы (б) не…. **б)** Пожелание. *Условие я бы предпочел не подписывать.* Л. Толст. Письмо А. Ф. Марксу, 27 марта 1899. ◊ С неопр. ф. глаг. *Поохотиться бы по-настоящему, на коня бы денег добыть, — мечтал старик.* Г. Марков, Строговы. ◊ В сочетании с предикативными наречиями со знач. долженствования, необходимости, возможности. … … ◊ Только бы (б), лишь бы, Употр. со знач. желательности действия. [Скалозуб:] *Мне только бы досталось в генералы.* Гриб. Горе от ума. **в)** Желание-просьба, совет или предложение (обычно при мест. 2л.). [Марина:] *И чего засуетился? Сидел бы:* Чех. Дядя Ваня.

The beginning of the parsing output shows the correct assignment of sense dependencies:

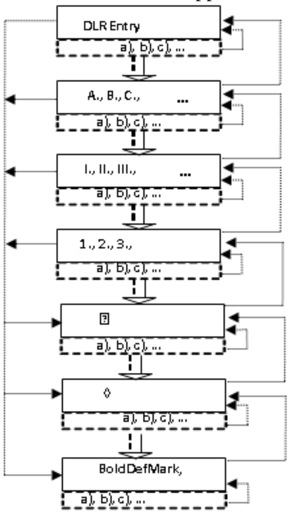
```
<entry>
 <list>БЫ 1. ◊ ◊ ◊ ◊ ◊ 2. 3. а) ◊ ◊ // ◊ ◊ б) ◊ ◊ ◊ в) г) ◊ ◊ n-23</list>
 <sense value="БЫ" class="0">
  <definition> (сокращенно <b>Б</b>), частица. В сочетании с глаголами в форме прошедшего
времени образует сослагательное наклонение. </definition>
  <sense value="1." class="4">
… … … … … … … … … … … … … … … … … … … … … … … … … … … … … … … … … …
  <sense value="3." class="4">
   <definition> Обозначает различные оттенки желаемости действия; </definition>
   <sense value="а)" class="5">
    <definition> Собственно желаемость. Учился бы сын. Были бы дети здоровы. </definition>
    <sense value="◊" class="8">
     <definition> Если <spaced> б ы</spaced>, когда <spaced> б ы</spaced>, хоть <spaced> б
ы</spaced><spaced> и</spaced> т. п. О, если бы когда-нибудь Сбылась поэта сновиденья! Пушк.
Посл. к Юдину. [Николка:] Хоть бы дивизион наш был скорее готов. Булгаков, Дни Турб.
</definition>
    </sense>
    <sense value="◊" class="8">
     <definition> С неопр. ф. глаг. Полететь бы пташечке К синю морю; Убежать бы молодцу в лес
дремучий. Дельв. Пела, пела пташечка.. … … </definition>
    </sense>
    <sense value="//" class="6">
     <definition> Употр. для выражения опасения по поводу … … … </definition>
     <sense value="◊" class="8">
      <definition> С неопр. ф. глаг., имеющей перед собой отрицание. <b>- </b>Гляди, - говорю, -
бабочка, не кусать бы тебе локтя! Так-таки оно все на мое вышло. Леск. Воительница. </definition>
     </sense>
… … … … … … … … … … … … … … … … … … … … … … … … … … … … … … … … … …
    </sense>
   </sense>
   <sense value="б)" class="5">
    <definition> Пожелание. Условие я бы предпочел не подписывать. Л. Толст. Письмо А. Ф.
Марксу, 27 марта 1899. </definition>
    <sense value="◊" class="8">
     <definition> С неопр. ф. глаг. Поохотиться бы по-настоящему, на коня бы денег добыть, -
мечтал старик. Г. Марков, Строговы. </definition>
    </sense>
    <sense value="◊" class="8">
… … … … … … … … … … … … … … … … … … … … … … … … … … … … … … … … … …
```

```
<sense value="в)" class="5">
  <definition> Желание-просьба, совет или предложение…. … … … </definition>
</sense>
<sense value="г)" class="5">
  <definition> Желаемость целесообразного и полезного действия. </definition>
  <sense value="◊" class="8">
… … … … … … … … … … … … … … … … … … … … … … … … … … … … … … … … … … … … …
```

4 The Parameterized Grammar of Dependency Hypergraph for the Second Parsing Level (SCD-config2) of DLR

In the course of modeling the SCD configurations for the 6 large dictionaries discussed previously, *i.e.* **DLR**, **DAR**, **TLF**, **DWB**, **GWB**, and **DMLRL** (Curteanu et al., 2008, 2010, 2012a, 2012b), we attempted to use the XCES TEI P5 (2007) dictionary standard for *encoding* the SCD *parsing method*. While, at the basic level, this encoding is sufficient, a detailed description of the sense marker classes and their dependency hypergraphs was not possible. We have therefore examined ways to extend the dictionary encoding standard towards the "*least common multiple*" for the representation of lexicographic structures and dependencies of the dictionaries we have parsed. Since this extension is carried out incrementally, the addition of further information, possibly absent from these dictionaries but present in others, is straightforward. Since the XML format can be described with a DTD, which is, at heart, a grammar representation, we have proposed the extensions given below as *parameterized grammars* for the task at hand.

For the sake of simplicity and ease of understanding, we propose the creation of three grammar types, one for each of the SCD configurations used for parsing. The current form of the SCD-config1 grammar for lexicographic segments is given below. The rules are split into groups, according to the reason for their presence in the grammar: general rules, which are applied to all dictionaries, rules added to represent **DLR** structures, rules added to represent **DAR** structures, etc. (the order of the added rules is that of analyzing the dictionaries). If a rule is already present, it is not added again, as the desired result is a seamless package for the SCD-config1 of all the six dictionaries. We use the attribute type for the *Body_sense* variable in order to link the segment marker type to the actual rule for expanding the body of a segment (the link between SCD-config1 and SCD-config2, as the SCD-config2 is *specific* to each segment it belongs to). In this grammar we also show how we can reuse large parts of the existing XCES TEI P5 dictionary encoding standard. The sense marker class dependency hypergraph of SCD-config2 is (Curteanu et al.; 2010, p. 41):



General
Entry → Root_sense Seg S
S → Seg S
S → ""
Seg → Mrk Root_sense Body_sense Tail_sense, type(Body_sense) = type(Mrk)
Mrk → ""

Root_sense → ""
Body_sense → ""
Tail_sense → ""

DLR
Mrk → newPrg, type(newPrg) = senseSeg
Mrk → newPrgDash, type(newPrgDash) = morphSeg
Root_sense → MorfDef
Body_sense → entry, if(type(Body_sense) = senseSeg)
Body_sense → MorphologicalPart, if(type(Body_sense) = morphSeg)
MorphologicalPart → Gram Etym
Gram → Gram TEI P5
Etym → Etym TEI P5

We propose the following parameterized grammar to describe the above dependency hypergraph functioning (*i.e.* SCD-Config2 parsing level for **DLR**). The rules are grouped in packages according to the direction of generation: *descending rules* go towards less general senses (*e.g.* **A.** to **I.**), *ascending rules* return to superior senses (*e.g.* **I.** to **A.**), describing the closing condition, while *splitting rules* are calls to the enumeration partitioning. The *enumeration* procedure is given in the *enumeration package*, as, although enumeration items are subsenses of their parents, they must meet certain restrictions described in the production rule attributes.

The attributes used are *parent* and *item*. The *parent* of a node is the sense from which that node is generated, and the *item* of an element is its position in the list of sister elements. In order to jump over sense levels, as most dictionaries do (*e.g.* **A.** to **1.**), we have used a *dummy node* for each skipped level, as the grammar is built so that it cannot generate a lower sense level without the superior level (this is a correctness restriction). The dummy nodes derivate to the empty string and are not itemized (the *item attribute* is not incremented for them).

entry → newPrg e LatCapLet; parent(LatCapLet) = e; item(LatCapLet) = 0
entry → e

LatCapLet → LatCapLet_Mrk LatCapNum; parent(LatCapLet_Mrk) = parent(LatCapLet);
    item(LatCapLet_Mrk) = item(LatCapLet) + 1; parent(LatCapNum) = LatCapLet_Mrk;
    item(LatCapNum) = 0
LatCapLet → LatCapLet_Dummy LatCapNum; parent(LatCapLet_Dummy) = parent(LatCapLet);
    item(LatCapLet_Dummy) = item(LatCapLet); parent(LatCapNum) = LatCapLet_Dummy;
    item(LatCapNum) = 0
LatCapLet → LatCapLet_Mrk; parent(LatCapLet_Mrk) = parent(LatCapLet);
    item(LatCapLet_Mrk) = item(LatCapLet) + 1

==**descending**==
LatCapNum → LatCapNum_Mrk ArabNum; parent(LatCapNum_Mrk) = parent(LatCapNum);
    item(LatCapNum_Mrk) = item(LatCapNum) + 1; parent(ArabNum) = LatCapNum_Mrk;
    item(ArabNum) = 0
LatCapNum → LatCapNum_Dummy ArabNum; parent(LatCapNum_Dummy) = parent(LatCapNum);
    item(LatCapNum_Dummy) = item(LatCapNum); parent(ArabNum) = LatCapNum_Dummy;
    item(ArabNum) = 0
LatCapNum → LatCapNum_Mrk; parent(LatCapNum_Mrk) = parent(LatCapNum);
    item(LatCapNum_Mrk) = item(LatCapNum) + 1
==**asending==c**

LatCapNum → LatCapLet; parent(LatCapLet) = parent(parent(LatCapNum));
    item(LatCapLet) = item(parent(LatCapNum))

==**descending**==
ArabNum → ArabNum_Mrk RombP; parent(ArabNum_Mrk) = parent(ArabNum); item(ArabNum_Mrk) =
    item(ArabNum) + 1; parent(RombP) = ArabNum_Mrk; item(rombP) = 0
ArabNum → ArabNum_Dummy RombP; parent(ArabNum_Dummy) = parent(ArabNum);
    item(ArabNum_Dummy) = item(ArabNum); parent(RombP) = ArabNum_Mrk; item(rombP) = 0
ArabNum → ArabNum_Mrk; parent(ArabNum_Mrk) = parent(ArabNum);
    item(ArabNum_Mrk) = item(ArabNum) + 1;
==**ascending**==
ArabNum → LatCapLet; parent(LatCapLet) = parent(parent (ArabNum));
    item(LatCapLet) = item(parent(ArabNum))

==splitting==
ArabNum → ArabNum_Mrk LatSmaLet; parent(ArabNum_Mrk) = parent(ArabNum);
    item(ArabNum_Mrk) = item(ArabNum) + 1; parent(LatSmaLet) = ArabNum_Mrk; item(LatSmaLet) = 0
ArabNum → ArabNum_Dummy RombP; parent(ArabNum_Dummy) = parent(ArabNum);
    item(ArabNum_Dummy) = item(ArabNum); parent(LatSmaLet) = ArabNum_Mrk; item(LatSmaLet) = 0

==descending==
RombP → ♦ RombG; parent(♦) = parent(rombP); item(♦) = item(RombP) + 1; parent(RombG) = ♦;
    item(RombG) = 0
RombP → RombP_Dummy RombG; parent(RombP_Dummy) = parent(rombP); item(RombP_Dummy) =
    item(RombP); parent(RombG) = RombP_Dummy; item(RombG) = 0
RombP → ♦; parent(♦) = parent(rombP); item(♦) = item(RombP) + 1
==ascending==
RombP → ArabNum; parent(ArabNum) = parent(parent(RombP)); item(ArabNum) = item(parent(RombP))

==splitting==
RombP → ♦ LatSmaLet; parent(♦) = parent(rombP); item(♦) = item(RombP) + 1; parent(LatSmaLet) = ♦;
    item(LatSmaLet) = 0
RombP → RombP_Dummy LatSmaLet; parent(RombP_Dummy) = parent(rombP); item(RombP_Dummy)
    = item(RombP); parent(LatSmaLet) = RombP_Dummy; item(LatSmaLet) = 0

==descending==
RombG → ◊ Atom; parent(◊) = parent(rombG); item(◊) = item(RombG) + 1; parent(Atom) = ◊;
    item(Atom) = 0
RombG → ◊; parent(◊) = parent(rombG); item(◊) = item(RombG) + 1
==ascending==
RombG → RombP; parent(RombP) = parent(parent(RombG)); item(RombP) = item(parent(RombG))

==splitting==
RombG → ◊ LatSmaLet; parent(◊) = parent(rombG); item(◊) = item(RombG) + 1; parent(LatSmaLet) = ◊;
    item(LatSmaLet) = 0
RombG → RombG_Dummy LatSmaLet; parent(RombG_Dummy) = parent(rombG);
    item(RombG_Dummy) = item(RombG); parent(LatSmaLet) = RombG_Dummy; item(LatSmaLet) = 0

==descending==
DefAtom → DefAtom_Mrk DefAtom; parent(DefAtom_Mrk) = parent(DefAtom); item(DefAtom_Mrk) =
    item(DefAtom) + 1; parent(DefAtom) = parent(DefAtomSt); item(DefAtom) = item(DefAtom_Mrk)
DefAtom → DefAtom_Mrk; parent(DefAtom_Mrk) = parent(DefAtom);
    item(DefAtom_Mrk) = item(DefAtom)+1

==ascending==
DefAtom → RombG; parent(RombG) = parent (parent(DefAtom)); item(RombG) = item(parent(DefAtom))
==splitting==
DefAtom → DefAtom_Mrk LatSmaLet; parent(DefAtom_Mrk) = parent(DefAtom); item(DefAtom_Mrk) = item(DefAtom) + 1; parent(LatSmaLet) = DefAtom_Mrk; item(LatSmaLet) = 0

==enumeration==
==descending==
LatSmaLet → LatSmaLet_Mrk RombP, if parent(LatSmaLet) > RombP; parent(LatSmaLet_Mrk) = parent(LatSmaLet); item(LatSmaLet_Mrk) = item(LatSmaLet) + 1; parent(RombP) = LatSmaLet_Mrk; item(RombP) = 0;
LatSmaLet → LatSmaLet_Mrk RombP, if parent(LatSmaLet) > RombG; parent(LatSmaLet_Mrk) = parent(LatSmaLet); item(LatSmaLet_Mrk) = item(LatSmaLet) + 1; parent(RombG) = LatSmaLet_Mrk; item(RombG) = 0;
LatSmaLet → LatSmaLet_Mrk LatSmaLet; parent(LatSmaLet_Mrk) = parent(LatSmaLet); item(LatSmaLet_Mrk) = item(LatSmaLet) + 1
==ascending==
LatSmaLet → ArabNum, if parent(parent(LatSmaLet)) > ArabNum; parent(ArabNum) = parent(parent(LatSmaLet)); item(ArabNum) = item(parent(LatSmaLet))
LatSmaLet → RombP, if parent(parent(LatSmaLet)) > RombP; parent(RombP) = parent(parent(LatSmaLet)); item(RombP) = item(parent(LatSmaLet))
LatSmaLet → RombG, if parent(parent(LatSmaLet)) > RombG; parent(RombG) = parent(parent(LatSmaLet)); item(RombG) = item(parent(LatSmaLet))
LatSmaLet → DefAtom, if parent(parent(LatSmaLet)) > DefAtom; parent(DefAtom) = parent(parent(LatSmaLet)); item(DefAtom) = item(parent(LatSmaLet))

We highlight the following particular and interesting new lexicographic units during the SCD lexicographic modeling: the intricate recognition and organization of **DWB** segments; the recursive "**Rem.**", "**Dér.**" (and other) segments in **TLF**; the "**TildaDef**" segment / package in **DMLRL**, with similar syntactic behavior as the "**Nest**" segment / package in **DAR**; the sense / definition inheritance "long-dash" marker and rules in **TLF** and **GWB**; the *dictionary authors*' *examples* in **DLR** and **DMLRL** (the latter, specially marked); the *indexed examples-to-definitions* package, specially marked and met only in **TLF**; various species of "*sigles*" (*i.e.* text source references) etc. While many lexicographic structures are similar or identical in their syntactic or semantic behavior over several dictionaries, the above mentioned examples should be integrated carefully within their appropriate SCD configuration, *i.e.* dependency hypergraph, described by corresponding parameterized grammars within the unitary procedural DTD.

5 Conclusion

The current DTD for dictionaries in the standard XCES TEI P5 (2007) represents dictionary entry data types, described with context-free grammars. For the recursive sense dependencies embodied into the dependency hypergraph of a parsing level, the challenge was to provide a formal tool for procedural description, and we delivered the parameterized grammars that describe the functioning of the first and second SCD-configurations (*i.e.* parsing levels) for **DLR**. This is the first phase of the newly proposed, procedural DTD, and there is still a lot of work to accomplish the project of such a general, procedural DTD. We will describe the dependency hypergraphs of SCD configurations for the *six* largest dictionaries we have already parsed (**DLR** completely, the other ones, partially) and augment the parameterized grammars on each level of SCD configuration, such that to obtain a *least common multiple* description for *all* the *six* considered dictionaries. The procedural DTD does not overlap the currently existing DTD for dictionaries in the TEI P5 standard, but effectively extends it from the detailed, static description of dictionary entry data types, to the procedural, hierarchically organized of *all* the component lexicographic structures, SCD-modeled, in the largest dictionaries.

We provided here the parameterized grammars for the first two SCD configurations of **DLR**, and achieved only one atomic sense dependency hypergraph (for **DMLRL**) from the six dictionaries involved. There are necessary (at least) 18 (6 dictionaries x 3 SCD-configurations) grammars, combined into their three *least common multiple* grammars on the synthesized *three* parsing levels of SCD configurations. Any new dictionary parsing experiments could possibly bring (or not!) novelties, incrementally integrated into the final version of the new procedural DTD for dictionaries, made up of (at least) three packages of parameterized grammars.

These are the dimensions of the project (which may also be called Document Structural Description – DSD, as the procedural completion to the current DTD for dictionaries). This project constitutes the formal (and incremental) description framework of a *general parser* for large thesaurus-dictionaries, proved to be *optimal*, *portable*, and *robust* (Curteanu *et al.*; 2010).

References

Curteanu, N., Moruz, A., Trandabăţ, D. (2008): *Extracting Sense Trees from the Romanian Thesaurus by Sense Segmentation & Dependency Parsing*, Proceedings of CogAlex-I Workshop, COLING-2008, Manchester, UK, pp. 55-63, http://aclweb.org/anthology/W/W08/W08-1908.pdf

Curteanu, N., Trandabăţ, D., Moruz, A. (2010): *An Optimal and Portable Parsing Method for Romanian, French, and German Large Dictionaries*, Proceedings of COGALEX-II Workshop, COLING-2010, Beijing, China, pp. 38-47, http://www.aclweb.org/anthology-new/W/W10/W10-3407.pdf

Curteanu, Neculai, Svetlana Cojocaru, Eugenia Burcă (2012a): *Parsing the Dictionary of Modern Literary Russian Language with the Method of SCD Configurations. The Lexicographic Modeling*. Comp. Science Journal of Moldova, Academy of Sciences of Moldova, Vol. 20, No.1(58), pp. 42-81, http://www.math.md/files/csjm/v20-n1/v20-n1-(pp42-82).pdf

Curteanu, Neculai, Svetlana Cojocaru, Alex Moruz (2012b): *Lexicographic Modeling and Parsing Experiments for the Dictionary of Modern Literary Russian Language*, ConsILR-2012 Proceedings, Bucharest, The Editorial House of ”Al. I. Cuza” University, Iaşi, pp. 189-198.

Curteanu, Neculai, Alex Moruz (2012c): *Toward the Soundness of Sense Structure Definitions in Thesaurus-Dictionaries. Parsing Problems and Solutions*. Computer Science Journal of Moldova, Academy of Sciences of Moldova, Vol. 20, No.3 (60), pp. 275-303, http://www.math.md/files/csjm/v20-n3/v20-n3-(pp275-303).pdf.

DWB (2010): Das Woerterbuch-Netz (2010): http://germazope.uni-trier.de/Projects/WBB/woerterbuecher/

DMLRL (1994): Dictionary of Modern Literary Russian Language (20 volumes, 1994): Словарь современного русского литературного языка. В 20 томах. Издательство: М.: Русский язык; Издание 2-е, перераб. и доп. 864 страниц; 1991-1994 г. ISBN: 5-200-01068-3 (in Russian).

TLF (2010): Le Trésor de la Langue Française informatisé (2010) : http://atilf.atilf.fr/tlf.htm

XCES TEI Standard, Variant P5, (2007): http://www.tei-c.org/Guidelines/P5/