Complex Natural Language Processing System Architecture

Stefan Diaconescu
sdiaconescu@softwin.ro

Ionut Dumitrascu
iodumitrascu@softwin.ro
5. Linguistic Applications

4. XML LKB (Linguistic Knowledge Bases)

3. Linguistic tools

2. GRAALAN - Grammar Abstract Language

1. Linguistic theoretical background
1. Linguistic Theoretical Background

1.1 DT - Dependency Tree
1.2 GDG - Generative Dependency Grammar
1.3 AVT - Attribute Value Tree
DT = \{N, T, P, A, SR, CR\} where:
- \(N\) - set of non-terminals \(n\): \(n(i_1, i_2, \ldots)\), \(ij \in SR\)
- \(T\) - set of the terminals \(t\): \(t(i_1, i_2, \ldots)\), \(ij \in SR\)
- \(P\) - set of pseudo-terminals \(p\): \(p(i_1, i_2, \ldots)\), \(ij \in SR\)
- \(A\) - set of procedural actions \(a\): \(a(i_1, i_2, \ldots)\), \(ij \in SR\)
- \(SR\) - set of subord. rel. \(sr\): \(sr(i_1)\), \(i_1 \in N \cup T \cup P \cup A \cup CR\)
- \(CR\) - set of the coordinate relations \(cr\): \(cr(f_1, f_2 / s_1, s_2, \ldots)\), \(fi \in N \cup T \cup P \cup A \cup CR\), \(si \in SR\)
“The teacher and the disciple came and saw that the shepherds and the strangers enjoy and dance.”
1.1 DT - Dependency Tree: Example (b)

```
a
@r6@
@r1@  @r2@
           ↙  ↙
  1     2
“strangers” “shepherds”

“enjoy”
“dance”

“and”
“the”
```

- "enjoy" and "dance" are related to "shepherds" and "strangers" by dependency links.
- "and" connects "strangers" and "shepherds".
- "the" is associated with "enjoy" and "dance".

Diagram represents the dependency structure of the sentence: "The shepherds and strangers enjoy dance."
GDG = \{N, T, P, A, SR, CR, nt0, R\} where:

- **N** - set of non-terminals \( n: n(i1, i2, ...), ij \in SR \)
- **T** - set of the terminals \( t: t(i1, i2, ...), ij \in SR \)
- **P** - set of pseudo-terminals \( p: p(i1, i2, ...), ij \in SR \)
- **A** - set of procedural actions \( a: a(i1, i2, ...) , ij \in SR \)
- **SR** - set of subord. rel. \( sr(i1), i1 \in N \cup T \cup P \cup A \cup CR \)
- **CR** - set of the coordinate relations \( cr: cr(f1, f2, ... / s1, s2, ...), fi \in N \cup T \cup P \cup A \cup CR , si \in SR \)
- **nt0** - belongs to N and is named root symbol.
- **R** - set of numbered rules of the form \((i) ni ->(si, qi), ni \in N, si is a sequence of elements from N \cup T \cup P \cup A, qi is a dependency tree having nodes from si and oriented links (relations) from SR \cup CR\)
Example of a grammar that can generate a phrase like the following:
“We provide practical advice for system and working group administrators.”

(1) <phrase> -> ( <nominal group> <verbal group>,
          <nominal group>( r1 (<verbal group>())))
(2) <nominal group> -> ( “we”,
                          “we”())
(3) <verbal group> -> ( <verb> <complement> <complement’>,
                        <verb>( r2( <complement>() ), r3( <complement’>())))
(4) <complement> -> ( <attribute> <noun>,
                      <noun>( r4( <attribute>())))
(5) <complement> -> ( “for” <coordination>,
                      “for”( <coordination>()))
(6) <coordination> -> ( <member> “and” <member’>,
                        r5( <member>(), <member’>() / r6( “and”())))
(7) <member> -> ( <noun>,
                   <noun>())
(8) <member> -> ( <attribute> <noun>,
                   <noun>( r7( <attribute>())))
(9) <attribute> -> ( <attribute> <noun>,
                   <noun>( r8( <attribute>())))

1.2 GDG – Generative Dependency Grammar: Example (a)
1.2 GDG – Generative Dependency Grammar: Example (b)

(10) <attribute> -> ( <noun>,
    <noun>())
(11) <attribute> -> ( <adjective>,
    <adjective>())
(12) <attribute> -> ( “practical”,
    “practical”())
(13) <verb> -> ( “provide”,
    “provide”())
(14) <noun> -> ( “advice”,
    “advice”())
(15) <noun> -> ( “system”,
    “system”())
(16) <noun> -> ( “administrator”,
    “administrator”())
(17) <noun> -> ( “group”,
    “group”())
(18) <adjective> -> ( “working”,
    “working”())
1.2 GDG – Generative Dependency Grammar: Example – generation (c)

(19) (1, 2) <phrase> -> ( “we” <verbal group>,
   “we” ( r1(<verbal group>( )))

(20) (19, 3) <phrase> -> ( “we” <verb> <complement> <complement’>,
   “we” ( r1( <verb> ( r2( <complement>( )),
   r3 ( <complement’>( )))

(21) (20, 13) <phrase> -> ( “we” “provide” <complement> <complement’>,
   “we” ( r1(“provide” ( r2( <complement>( )),
   r3( <complement’>( )))

(22) (21, 4) <phrase> -> ( “we” “provide” <attribute> <noun> <complement’>,
   “we” ( r1(“provide” ( r2( <noun>(r4( <attribute>( ))),
   r3(<complement’>( ))

(23) (22, 5) <phrase> -> ( “we” “provide” <attribute> <noun> “for” <coordination>,
   “we” ( r1(“provide” ( r2( <noun>(r4( <attribute>( ))),
   r3(“for”( <coordination>( )))

(24) (23, 12) <phrase> -> ( “we” “provide” “practical <noun> “for” <coordination>,
   “we” ( r1( “provide” ( r2( <noun>(r4( “practical”( ))),
   r3( “for”(<coordination> ( ))))


1.2 GDG – Generative Dependency Grammar: Example – generation (d)

(25)(24,14) <phrase> -> ( “we” “provide” “practical” “advice” “for” <coordination>,
             “we”( r1( “provide”( r2( “advice”(r4( “practical”( )))),
             r3( “for”( <coordination> ( ))))))
(26)(25,6) <phrase> -> ( “we” “provide” “practical” “advice” “for” <member> “and”
              <member’>,
             “we”( r1( “provide”( r2( “advice”(r4( “practical”( )))),
             r3( “for”( r5 ( <member>( ),
              <member’>( ) / r6( “and”( )))))))
(27)(26,7) <phrase> -> ( “we” “provide” “practical” “advice” “for” <noun> “and”
              <member’>,
             “we”( r1( “provide”( r2( “advice”(r4( “practical”( )))),
             r3( “for”( r5 ( <noun>( ),
              <member’>( ) / r6( “and” ( )))))))
(28)(27,15) <phrase> -> ( “we” “provide” “practical” “advice” “for” “system” “and”
               <member’>,
             “we”( r1( “provide”( r2( “advice”(r4( “practical”( )))),
             r3( “for”( r5 ( “system”( ),
              <member’>( ) / r6( “and”( ))))))
1.2 GDG – Generative Dependency
Grammar: Example – generation (e)

\[
(29)(28,8) \text{ <phrase> } \rightarrow \text{ ( “we” “provide” “practical” “advice” “for” “system” “and” }
\text{ <attribute> <noun>,}
\text{ “we”( r1( “provide”( r2( “advice”(r4( “practical”( ))))),}
\text{ r3( “for”( r5( “system”( ),}
\text{ <noun>(r7( <attribute>( )( ) / r6( “and”( ))))))})
\]

\[
(30)(29,16) \text{ <phrase> } \rightarrow \text{ ( “we” “provide” “practical” “advice” “for” “system” “and” }
\text{ <attribute> “administrator”,}
\text{ “we”( r1( “provide”( r2( “advice”(r4( “practical”( ))))),}
\text{ r3( “for”( r5( “system”( ),}
\text{ “administrator”( r7( <attribute>( )( ) / r6( “and”( ))))))})
\]

\[
(31)(29,9) \text{ <phrase> } \rightarrow \text{ ( “we” “provide” “practical” “advice” “for” “system” “and” }
\text{ <attribute> <noun> “administrator”,}
\text{ “we”( r1( “provide”( r2( “advice”(r4( “practical”( ))))),}
\text{ r3( “for”( r5( “system”( ),}
\text{ “administrator”( r7( <noun>( r8( <attribute>( )( ) ) / r6( “and”( ))))))})
\]
(32)(31,17) <phrase> -> ( "we" "provide" "practical" "advice" "for" "system" "and"
 <attribute> "group" "administrator",
 "we"( r1( "provide"( r2( "advice"( r4( "practical"( ))))),
 r3( "for"( r5( "system"( ))),
 "administrator"( r7( "group"( r8( <attribute>( )))) / r6( "and"( )))))))
(33)(32,11) <phrase> -> ( "we" "provide" "practical" "advice" "for" "system" "and"
 <adjective> "group" "administrator",
 "we"( r1( "provide"( r2( "advice"( r4( "practical"( ))))),
 r3( "for"( r5( "system"( ))),
 "administrator"( r7( "group"( r8( <adjective>( )))) / r6( "and"( )))))))
(34)(33,18) <phrase> -> ( "we" "provide" "practical" "advice" "for" "system" "and"
 "working" "group" "administrator",
 "we"( r1( "provide"( r2( "advice"( r4( "practical"( ))))),
 r3( "for"( r5( "system"( ))),
 "administrator"( r7( "group"( r8( "working"( )))) / r6( "and"( ))))))
1.2 GDG – Generative Dependency Grammar: Example – generated DT (g)

- "we"
- "provide"
- "for"
- "admin."
- "working"
- "group"
- "system"
- "advice"
- "practical"
- "and"
### 1.3 AVT – Attribute Value Tree:
(simplified, syntactic) Definition

<table>
<thead>
<tr>
<th>Rule</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td><code>&lt;AVT&gt;</code></td>
</tr>
<tr>
<td>2.</td>
<td><code>&lt;attribute list&gt;</code></td>
</tr>
<tr>
<td>3.</td>
<td><code>&lt;attribute&gt;</code></td>
</tr>
<tr>
<td>4.</td>
<td><code>&lt;attribute content&gt;</code></td>
</tr>
<tr>
<td>5.</td>
<td><code>&lt;feature content&gt;</code></td>
</tr>
<tr>
<td>6.</td>
<td><code>&lt;attribute value list&gt;</code></td>
</tr>
<tr>
<td>7.</td>
<td><code>&lt;attribute value element&gt;</code></td>
</tr>
</tbody>
</table>
1.3 AVT – Attribute Value Tree: Example (a)

- Complex group type
  - Correlative
  - Distributive
  - Logical
- Complex group role
  - Subjective-predicative
  - Nominal-attributive
  - Verbal-completive
- Case
  - Nominative
  - Genitive
  - Dative
  - Accusative
  - Vocative
- Sequence type
  - g = governor
  - s = subordinate
  - gsg = governor subordinate governor
  - gs = governor subordinate
  - sgs = subordinate governor
  - sg = subordinate governor
  - g = subordinate governor

- Articulate
  - Not articulate
- Position
  - Left
  - Right

Example (a)
1.3 AVT – Attribute Value Tree: Example (b)

- animation
  - not animated
  - animated

- gender
  - masculine
  - feminine
  - neuter

- number
  - singular
  - plural

- person
  - I
  - II
  - III
<subordinated complex group>
{[type = correlative, distributive, logical]
[complex group role =
    nominal-attributive
    {NominalAttributivSubjectivePredicative:
        [case = nominative, genitive, dative, accusative, vocative]
        [sequence type =
            governor {ArticulatePosition:
                [articulate = not articulate, definite]
                [Position: position against the accordant = left, right]},
            subordinate {ArticulatePosition},
            governor subordinate governor,
            governor subordinate,
            subordinate governor subordinate,
            subordonate governor] },
    subjective-predicative {NominalAttributivSubjectivePredicative},}
verbal-completive [sequence type =
  governor [Position],
  subordinate [Position],
  governor subordinate governor,
  governor subordinate,
  subordinate governor subordinate,
  subordinate governor]
[animation = not animated, animated]
[gender = masculine, feminine, neuter]
[number = singular, plural]
[person = I, II, III]
1.3 AVT – Attribute Value Tree: Properties

- Paths in the AVT
- EC – Exclusive combinations in the AVT
- AVT equivalence
- Well formed AVT
- AVT ordering
- AVT intersection
- AVT difference
- AVT union
- AVT factoring
- AVT normalization
- AVT unifiability
- AVT unification
GRAALAN is a language that allows the description of:

a) A natural language;

b) A correspondence between two natural languages.
2. GRAALAN: Sections

- Alphabet
- Lexicon
- Syllabification
- Morphology (Morphological configuration)
- Inflection rules
- Inflection forms
- Syntax
- Bilingual correspondences
Role: Defines the codes of the signs used to represent and describe the language.

Sign codes:
- a) Phonetic alphabet (IPA – International Phonetic Alphabet)
- b) Normal alphabet
- c) The special characters
- d) Groups of characters (diphthongs or triphthongs, etc.).
  - Normal + Phonetic.
- e) (Alphabetic) Classes (vowel class, consonant class etc.)

Other functions:
- All the characters used in GRAALAN use UNICODE.
- A sorting method can be defined
2. GRAALAN: Alphabet Section: how it looks like…

**Phonetic alphabet**
/* a */ character code = "&#x0061;" type = internal label = open_central_unrounded
/* ə */ character code = "&#x0259;" type = internal label = mid_central_unrounded

............... 

**Normal alphabet**
/* a */ character code = "a" type = internal label = a
/* A */ character code = "A" type = internal label = A

............... 

**Groups**
/* a, A */ group code = ("a", "A") ["&open_central_unrounded;"] label = a_group
/* iou */ group code = ("iI","oO","uU")
["&semivowel_i;&mid_back_rounded;&semivowel_u;"] label = triphthong_iou

............... 

**Classes**
class label = vowel elements = ("a", "A", "e", "E", "i", "I", "o", "O", "u", "U", "&abreve;", ...)
class label = diphthong
elements = ("&diphthong_ai;", "&diphthong_au;", "&diphthong_ei;", ...)

...............
Role: Defines the syllabification rules

Syllabification types
- a) Euphonic syllabification (for the word written with the normal or special alphabet).
- b) Phonetic syllabification (for the word written with the phonetic alphabet).
- c) Morphologic syllabification (it respects the morphological structure of the word).

The elements of a word that are „separated” by syllabification (or not)
- Type (a): The normal alphabet characters
- Type (b): Groups (diphthongs, triphthongs, etc) described in Alphabet section (phonetic groups).
- Type (c): Some special characters.
- Type (d): Other constitutive elements (morphemes) described in Lexicon section (morphological groups).

Syllabification rules
- Euphonic rules
- Phonetic rules
2. GRAALAN: Syllabification Section: how it looks like

Euphonic
/* 1 */    Rule "&vowel;" - "&vowel;";
/* 2 */    Rule "&vowel;" - "&diphthong;";
/* 3 */    Rule "&vowel;" - "&tripthong;";
/* 4 */    Rule "&diphthong;" - "&diphthong;";
/* 5 */    Rule "&vowel;" - "&consonant;" + "&vowel;";
/* 5.1 */   Rule "&vowel;" - "&che_cons_voc;";

.........................
Role: Defines the lexical-morphological categories and values

- It is an AVT (Attribute Value Tree) with:
  - Attribute nodes = lexical-morphological categories
    - the category name
    - the abbreviation of the category name
    - the indication if the category is inflected or not
    - (eventually) the name of a procedural program
  - Value nodes = lexical-morphological category values
    - the category value name
    - the abbreviation of the category value name
    - belongs to a lemma (or not)
    - belongs to a lexicon entry (or not)
    - (eventually) the name of a procedural program.
2. GRAALAN: Morphological Configurator
Section: how it looks like

| clasa / name = Clasa, abbreviation =Cls, inflection = no / |
| = substantiv / name = Substantiv, abbreviation = Subst, lemma = yes, lexicon = input / |
| [tip substantiv / name = TipSubstantiv, abbreviation = TipSubst, inflection = no / |
| = comun / name = Comun, abbreviation = Com, lemma = yes, lexicon = input / |
| , propriu / name = Propriu, abbreviation = Pr, lemma = yes, lexicon = input / ] |
| animatie / name = Animatie, abbreviation = Animat, inflection = no / |
| = animat / name = Animat, abbreviation = Anim, lemma = yes, lexicon = input / |
| , inanimat / name = Inanimat, abbreviation = Inanim, lemma = yes, lexicon = input / ] |
| [GEN_NEFLEXIONAT: gen / name = Gen, abbreviation = Gen, inflection = no / |
| = masculin / name = Masculin, abbreviation = Masc, lemma = yes, lexicon = input / |
| , feminin / name = Feminin, abbreviation = Fem, lemma = yes, lexicon = input / |
| , neutru / name = Neutru, abbreviation = Neu, lemma = yes, lexicon = input / ] |

........]
Role: defines words, expressions and lexical / syntactic structures

Types of entries
• a) Morphemes (roots, prefixes, suffixes, prefixoids, suffixoids, etc.)
• b) Words:
  – Lemma is the basic form (canonical form) of a word.
  – Supplementary input: is an inflected form of a word that accompanies the lemmas in an ordinary dictionary (for example plural form of a noun).
  – Wordform is an inflected form of another word (therefore not a lemma) but that usually appears in a dictionary (for example "you" that can be considered an inflected form of "I").
• c) Multi Word Expression (MWE): groups of words represented as a DT (Dependency Tree)
• d) Morphological analytical structures – as MWEs
• e) Syntactic structures – as MWEs
Types of informations attached to different entries:

- a) Semantic information (gloss, synonyms, antonyms, paronyms, hiperonyms, hyponyms, connotations, homonyms, meronyms)
- b) Etymology (original language, original form, transliteration of the original form)
- c) Syllabification (euphonic, phonetic and morphologic)
- d) Morphology (inflection situation, inflection rule identification, segmentation)
- e) Sort (how the word must be put in a list of ordered words)
- f) etc.
Role: Defines the rules that can be used to generate the inflected forms.

Lemma (from the lexicon) indicates a Compound rule.

- a) Compound rules: list of basic rules.
- b) Basic rule: an AVT where each leaf has one or more associated elementary inflection rules.
- c) Elementary inflection rule contains:
  - a condition (logical expression) that indicates when the transformation sequence must be used.
  - a transformation sequence (insert, delete, replace word or chars) acting on normal alphabet
  - a transformation sequence (insert, delete, replace word or chars) acting on phonetic alphabet form
  - an AVT for analytic forms
  - relations in a DT (dependency tree) for analytic forms
Lexicon

Inflection rules

Compound inflection rule

Basic inflection rule

AVT

Elementary inflection rule

Condition

Transformation seq. (normal alph.)

Transformation seq. (phonetic alph.)

DT/AVT (for analytic forms)
2. GRAALAN: Inflection Rules Section: how it looks like

**Basic Rule**

Subst_masc1: [clasa = substantiv] [tip substantiv = comun] [tip substantiv comun = animat, inanimat] [gen = masculin] [numar = singular] [caz = nominativ] [articulare = nearticulat (EtS0: alphabetic -), hotarat (EtS11: if(&consonant;) alphabetic insert "ul" if("i") alphabetic insert "ul" if("u") alphabetic insert "l" if("e") alphabetic insert "le"), nehotarat (EtS12: alphabetic insert word left "un" [clasa = articol] [tip articol = nehotarat] [caz = nominativ] [gen = masculin] [numar = singular] @acord_gen-numar-caz@)

], genitiv ......] ]
Role: Describes the inflected forms of the language

- Contains an entry for an inflected form. An entry contains:
  - The inflected form written using the normal alphabet
  - The inflected form written using the phonetic alphabet
  - The reference of the word in the lexicon whose inflected form is the current entry.
  - The characterizing of the inflection situation (i.e. an AVT with lexical categories and lexical categories values).
  - How the inflected form is syllabified in different situations:
    - euphonic
    - phonetic
    - Morphologic
    - at the end of the line (hyphenation).
  - How the sorting is done
    - the word from the inflection form that is used
    - the sort from left to right, or from right to left
Role: Defines the syntax rules.

It is described as a sequence of labeled rules
- The left part contains a non terminal and an AVT
- The right part contains one or more alternants.

An alternant is formed by a set of subsections:
- a) The syntactic subsection: a sequence of NTPAs (N = non-terminals; T = terminals; P = pseudo-terminals; A = procedural actions)
  - The name of the NTPA
  - the associated relations with other NTPAs
  - The associated AVT
- b) The dependency subsection:
  - The subordinate relations SR
  - The coordinate relations CR
- c) The agreement section contains a list of agreement rules [24]:
  "if(conditional expression)
   true (actions)
   false (actions)
   not applicable (actions)
   not determinate (actions)"
2. GRAALAN: Syntax Section (b)

Grammar Rule

Left part

<non terminal>

[AVT] ::=  

Alternant sequence

Alternant 1

Syntactic subsection

Alternant 2

Dependency subsection

Alternant 3

Agreement subsection

Right part

Sequence of

<N non terminal>[AVT]CR/SR/GR

“T terminal” [AVT]CR/SR/GR

%A procedural action#[AVT]CR/SR/GR

@CR coordinate relations@ CR/SR/GR

@SR subordinate relations@

- agreement rules with condition expressions and actions
### Rule

\[\text{RegentSubordonatNominalAtributivNominitivAcuzativ:} \]
\[\langle\text{regent/subordonat}\rangle\]
\[\quad \text{[rol grup complex = nominal-atributiv]}\]
\[\quad \text{[caz = nominativ, acuzativ] [animatie = neanimat, animat]} \ldots ::=\]

\[\vdots\]

### Alternant

**Alternant2:**

**Syntax**

\[\text{Eticheta1:} \langle\text{articol}\rangle\]
\[\quad \text{[tip articol = nehotarat]}\]
\[\quad \text{(gen = masculin, feminin, neutru)}\]
\[\quad \text{(numar = singular, plural)} \ldots\]

**Subordinate** Eticheta6

\[\text{Eticheta2:} \langle\text{secventa grup complex subordonat}\rangle\]
\[\quad \text{[rol grup complex = nominal-atributiv]}\]
\[\quad \text{[tip = corelativ, distributiv, logic]}\ldots\]

**Subordinate** Eticheta7

\[\text{Eticheta3:} \langle\text{regent}\rangle\]
\[\quad \text{[rol grup complex = nominal-atributiv]}\]
\[\quad \text{(exprimat prin = substantiv, pronume, \ldots)}\ldots\]

**Governor** Eticheta5, Eticheta6, Eticheta7
Eticheta4: <atribut> [rol grup complex = nominal-atributiv] 
[exprimat prin = adjectiv propriuzis] …

**Subordinate** Eticheta5

**Dependencies**
- Eticheta5: @relatie atribut / regent@
- Eticheta6: @relatie articol nehotarat@
- Eticheta7: @relatie atribut / regent@

**Agreement**
- $\text{regula simpla a numarului SN } ("Eticheta1", "Eticheta3")$
- $\text{regula simpla a genului SG } ("Eticheta1", "Eticheta3")$
- $\text{regula simpla a numarului SN } ("Eticheta2", "Eticheta3")$
- $\text{regula simpla a genului SG } ("Eticheta2", "Eticheta3")$
- $\text{regula simpla a numarului SN } ("Eticheta4", "Eticheta3")$
- $\text{regula simpla a genului SG } ("Eticheta4", "Eticheta3")$
Role: Defines the correspondences between two languages

- **a) MWE (Multi Word Expression) correspondences**: contains transformation rules based on DT form of MWE where nodes can be:
  - Invariable elements.
  - Partial variable elements
  - Total variable elements
- **b) Word correspondences**: particular cases of the MWE correspondences (both MWEs have only one word).
- **c) Syntactic structure correspondences**: a particular case of MWE correspondences where the nodes can be non-terminals.
- **d) Morphological analytic structure correspondences**: a particular case of MWE (the correspondences between analytic inflection forms)
- **e) Morphological sub-tree correspondences**: a particular case of MWE (i.e. express the correspondences between a source morphological sub-tree and a target morphological sub-tree).
2. GRAALAN: Other features

- *Messages section*
- *Macros*
3. Linguistic Tools (a)

3.1 GRAALAN Compiler
3.2 GRAALAN Macroprocessor
3.3 LINK
3.4 Inflection Forms Tool
3.5 Lexicon Tool
3. Linguistic Tools (b)

- Linguist
  - Macro GRAALAN
  - Macro processor
  - GRAALAN text
  - GRAALAN compiler
  - XML LKB
  - Infl. Forms Tool
  - Lexicon Tool
  - GRAALAN Link

XML LKB
4. Linguistic Knowledge Bases (XML LKB)

- Alphabet XML LKB
- Syllabification XML LKB
- Morphology XML LKB
- Inflection rules XML LKB
- Inflection forms XML LKB
- Lexicon XML LKB
- Syntax XML LKB
- Bilingual correspondences XML LKB

- Alphabet DTD
- Syllabification DTD
- Morphology DTD
- Inflection rules DTD
- Inflection forms DTD
- Lexicon DTD
- Syntax DTD
- Bilingual correspondences DTD
5. Linguistic Applications

<table>
<thead>
<tr>
<th>Applications</th>
</tr>
</thead>
<tbody>
<tr>
<td>Morphological analyzer</td>
</tr>
<tr>
<td>Grammar checker</td>
</tr>
<tr>
<td>Inflection</td>
</tr>
<tr>
<td>Indexing/Searching</td>
</tr>
<tr>
<td>Lemmatiser</td>
</tr>
<tr>
<td>Speller</td>
</tr>
<tr>
<td>Hyphenating</td>
</tr>
<tr>
<td>Lexicon</td>
</tr>
<tr>
<td>Lexical dictionary</td>
</tr>
<tr>
<td>Human assisted machine translation</td>
</tr>
<tr>
<td>Computer assisted machine translation</td>
</tr>
<tr>
<td>Automatic machine translation</td>
</tr>
<tr>
<td>etc.</td>
</tr>
</tbody>
</table>
Conclusions

- NLPS Architecture presented is quite **general** and appropriate for any language.

- A system with this architecture is **in progress** at a Romanian company.

- The **language GRAALAN** is already defined.

- Some tools are already developed (**GRAALAN Macroprocessor**, **GRAALAN Compiler**, **Inflection Forms Tool**) some tools are currently in design / implementation stage (**Lexicon Tool**, **LINK**).

- Some Romanian linguistic knowledge bases are already defined (**Alphabet Section**, **Morphological Configurator Section**, **Syllabification Section**, **Inflection Rules Section**), some are partially developed (**Lexicon Section**, **Syntax Section**), some will be developed until the end of this year (**Inflection Forms Section**).
• Q&A