

Electronic dictionaries for the automatic analysis of texts: from INTEX to NooJ

Slim Mesfar

Université de Franche Comté
Besançon, France
mesfarslim@yahoo.fr

1 INTEX and DELA system dictionaries

The INTEX linguistic engine (Silberztein 1993) used two sets of dictionaries that were designed at Prof. Maurice Gross's LADL laboratory: on the one hand, DELA-type dictionaries describe simple and compound words (Courtois, Silberztein 1990) within four types of dictionaries:

- ✓ DELAS : Electronic Dictionary of LADL for Simple word forms
- ✓ DELAF : Electronic Dictionary of LADL for inFlected simple word forms
- ✓ DELAC : Electronic Dictionary of LADL for Compound word forms
- ✓ DELACF : Electronic Dictionary of LADL for inFlected compound word forms

On the other hand, lexicon-grammars describe frozen expressions (Gross M. 1993).

The INTEX lexical parser recognizes simple words by looking up DELAF-type dictionaries, whose entries are all the inflected word forms of a language (Courtois 1990). For instance, the following is a sample of a DELAF:

```
avais,avoir.V:I2s  
a,avoir.V:P3s  
avoir,avoir.V:INF  
ayant,avoir.V:ANT
```

Each entry (e.g. "avais") is associated with its lemma ("avoir"), a category ("V") and a series of inflectional codes ("I" = "Imperfect", "2" = "second person" and "s" = "singular").

DELAFs are themselves constructed automatically from DELAS-type dictionaries, whose entries are lemmas associated with an inflectional paradigm that allows INTEX to automatically generate the DELAFs.

In the same manner, INTEX recognizes compound words by a lookup of DELACF-type dictionaries, in which inflected forms of each compound word are explicitly listed (Silberztein 1990). For instance following is a sample of a DELACF dictionary:

```
station de sport,station de sport.N:s
stations de sport,station de sport.N:p
rendez-vous,rendez-vous.N:s
rendez-vous,rendez-vous.N:p
```

In INTEX, DELACF-type dictionaries are constructed semi-automatically from DELACs (similarly to the DELAS/DELAf system), through a complex series of INTEX PERL and manual editing operations.

Recognizing frozen expressions is more complex. Frozen expressions are described in Lexicon-grammars (Gross M. 1993). Lexicon-Grammars are displayed as tables, in which each line corresponds to a lexical entry, and each column corresponds to a property. Property values are either binary (“+” or “-”), or textual (e.g. **PREP** = “for”).

In order to automatically recognize frozen expressions in texts, INTEX uses slightly adapted lexicon-grammar tables, in conjunction with **metagraphs** (Silberztein 1999) (Vietri 2004). INTEX meta-graphs are context-free grammars (CFGs) that contain links to cells in the table. For instance, the following is a sample of a lexicon-grammar table that represents French frozen expressions of the **N0 V N1** syntactic structure (e.g. *Jean abandonne le terrain* – the 4th line in Figure1):

| | A | B | C | D | E | F | G | H | I | J | K |
|----|-------------|-------|--------|----------|-----|-----|------|----------|-------------------------|------|--------|
| | V | NOnum | NO_num | NegOblig | PPV | NOV | Dest | NOVDEFNI | NI | Nlpe | Possif |
| 2 | abandonner | + | | | | + | la | | partie | | + |
| 3 | abandonner | + | | | | | la | + | pose | | + |
| 4 | abandonner | + | | | | | le | | domicile conjugal | | + |
| 5 | abandonner | + | | | | | le | | terrain | | + |
| 6 | abandonner | + | | | | | les | | lieux | | + |
| 7 | abjurer | + | | | | | la | | foi catholique | | + |
| 8 | abolir | | + | | | | le | | temps | | + |
| 9 | accélérer | + | | | | | le | | mouvement | | + |
| 10 | accélérer | + | + | | | | le | | mouvement | | + |
| 11 | accompagner | + | | | | | le | | pas | | + |
| 12 | accomplir | + | | | | | le | | rêve de «DET+PossO» vie | | + |
| 13 | accrocher | + | | | | | le | | linge | | + |
| 14 | accuser | + | | | | | le | | coup | | + |
| 15 | accuser | + | + | | | | les | | années | | |
| 16 | accuser | + | + | | | | les | | ans | | |
| 17 | achever | + | | | | | le | | tableau | | |
| 18 | activer | + | + | | | | le | | mouvement | | |
| 19 | adorer | + | | | | | le | | veau d'or | | + |
| 20 | adoucir | | + | | | | le | | temps | | + |
| 21 | affronter | + | | | | | la | + | mort | | + |
| 22 | afficher | + | | | | | la | | couleur | | + |
| 23 | affronter | + | | | | | le | + | danger | | + |
| 24 | agiter | + | | | | | la | | main | + | |
| 25 | amener | | + | | | | la | + | montagne | | + |

Figure 1: A lexicon-grammar table for INTEX

Comments on INTEX dictionaries

Problems with the INTEX approach have been discussed extensively in (Silberztein 2004). The most important are:

- Separation between simple and compound inflected forms dictionary, DELAF and DELACF. In addition, INTEX is not capable of processing compounds inflection automatically, etc.
- Use of the pair (lexicon-grammar table, meta graph) is awkward. Note that meta-graphs cannot be included or re-used within other INTEX syntactic grammars.
- Non adequacy of DELAS/DELAC dictionaries to either multilingual applications or semantic applications in which lexical entries are associated with synonymous terms or expressions, hyperonyms, etc (Koeva S., Mihov S. 2005). Not to mention that INTEX needs different types of dictionaries, each one with its own formalism...
- Limitation of use to Latin alphabet Romance languages.

So, the INTEX linguistic engine needed to be redesigned to process Asian and Semitic languages and much more ones known as languages with rich morphology (e.g. German, Korean).

2 NooJ's lexical engine

In NooJ, these five dictionaries are represented in one unique format. In order to unify the formalism, we needed to (1) merge simple and compound words (2) get rid of DELAFs and DELACFs and (3) add information to DELA-type dictionaries so that they can be processed in the same way as lexicon-grammar tables (Silberztein, 2005).

2.1 Merging simple and compound words

NooJ dictionaries contain indistinctly simple or compound words thanks to a new inflection system that can process both simple and compound words' inflectional morphology in a unified way.

For instance, the two following lexical entries:

```
table ,N+FLX=Table+Conc
voiture ,N+FLX=Table+Conc
```

inflect the same way (they take an 's' in the plural). Therefore, they are both associated with the same inflectional class: Table. The class Table is defined by the following expression:

```
Table = <E>/singular + s/plural;
```

that states that if one adds nothing to the lexical entry (“<E>” is the empty string), one gets the singular form (“singular”); if one adds an “s” to the end of the lexical entry, one gets the plural form (“plural”).

NooJ's inflectional engine is equivalent to a stack automaton. It uses a dozen default commands that operate on the suffix of lexical entries, as follows:

```
<B>: equivalent to the keyboard key “Backspace”
<D>: Duplicate current character
<E>: Empty string
<L>: equivalent to the keyboard key “Left arrow”
<N>: go to end of Next word form
```

<P>: go to end of Previous word form
<R>: equivalent to the keyboard key “Right arrow”
<S>: equivalent to the keyboard “delete” key

In addition to predefined operators, NooJ allows linguists to override these commands or add their own operators for each language to the inflectional engine. For instance:

- For Romance languages such as French, Spanish, Portuguese, 5 language-specific commands are added to process accentuated letters:
 - ✓ <A>: removes Accent
 - ✓ <Á>: adds acute accent
 - ✓ <Â>: adds circumflexe
 - ✓ <Ä>: adds dieresis
 - ✓ <À>: adds grave accent
- For Arabic, 3 language-specific commands are added to deal with the complex inflectional system (Mesfar, 2006):
 - ✓ <T>: processes final "ة" (Teh marbuta) in noun inflections
 - ✓ <M>: processes consonant 'n' in verbal lemmas to have the past, 3rd person, feminine, plural inflected form
 - ✓ <Z>: processes consonant 't' in verbal lemmas to have the past, 1st person, plural inflected form

In contrast with INTEX, NooJ is capable of inflecting compounds. For instance, the class “PommeDeTerre” is defined by the following expression:

```
PommeDeTerre = <E>/f+s + <PW>s/f+p;
```

The operator <PW> stands for: “go to the end of the first component of the lexical entry”. Note that the following three entries are associated with this class, even though their length is different:

```
pomme de terre,N+Conc+FLX=PommeDeTerre  
mouvement de terrain,N+Conc+FLX=PommeDeTerre
```

In the same manner, even though the lexical entries *carte de crédit*, *mouvement de grève*, *station essence*, *station de sport d’hiver*, etc. have different lengths, they can be associated with a unique inflectional class because the inflection is carried by the same component (the second one). More complex transformation could be described such as one in:

```
Cheval de bois,N+Conc+FLX=ChevalDeBois  
Cheval de course,N+Conc+FLX=ChevalDeBois  
ChevalDeBois=  
<E>/m+s + (→ cheval de bois)  
<PW><B>ux/m+p; (→ chevaux de bois)
```

To get the plural form, go to the first component of the compound (<PW>), delete the last character (), and add the suffix “ux”.

We can also process agreements between components of a compound can be described as well. For instance, the following inflectional expression formalizes the agreement between the two components of compounds such as *cousin germain*:

```
CousinGermain =  
<E>/m+s + (→ cousin germain)  
e<P>e/f+s + (→ cousine germaine)  
s<P>s/m+p + (→ cousins germains)  
es<P>es/f+p;( → cousines germaines)
```

To get the masculine plural form, add an “s” to the end of the compound, then go back to the previous (<P>) component, add an “s”.

In conclusion, NooJ can process simple and compound words’ inflection completely automatically, whereas INTEX could only process simple words automatically. This has allowed us to unify the description of simple and compound words.

2.2 Getting rid of DELAFs

NooJ’s lexical parser requires no DELAF/DELACF-type dictionaries. NooJ dictionaries, which are in effect similar to DELAS (or DELAC) dictionaries, are compiled into a Finite-State Transducer, see § 3, that is more powerful than previous DELAF transducers, and faster in some cases.

This characteristic is more important than a mere increase in comfort for the users, or even an increase in efficiency: encoding the inflectional information inside NooJ dictionaries gives NooJ the ability to perform morphological operations *during parse time*. Whereas INTEX could link an inflected form to its lemma (by a lookup of the DELAF transducer), NooJ can link any inflected form to any other inflected form. Thus, NooJ can perform complex transformations within texts. For instance, it is now possible to replace a certain conjugated verb with its past participle form, and vice-versa, within a particular text:

Jean *mange* la pomme => la pomme *est mangée* par Jean

Using this new functionality will enhance several current NLP applications, such as automatic translation and information retrieval applications.

2.3 Unifying Lexicon-Grammar tables and DELA dictionaries

In INTEX, lexicon-grammar tables are associated with a predefined set of properties. Any modification or addition of a property requires modification of the whole table and associated meta-graph, as well as the recompilation of the resulting CFG grammar. However, DELA-type dictionaries had open sets of properties. This openness has some disadvantages such as the fact that properties cannot be checked, and it is difficult for lexicographers to maintain more than a few properties.

In NooJ, we try, at the same time, to overcome these inconveniences and to not lose the advantages of both lexicon-grammar tables and DELA-type dictionaries. We adapted both the notation of lexicon-grammar tables and DELA-type dictionaries so that both can be displayed either in list form or in table form.

Importing a lexicon-grammar table in NooJ is straightforward (4), but it requires rendering all the properties that are common for all the entries of the table explicit. Each table of the lexicon-grammar is indeed defined by a set of syntactic properties, see (Leclère 2002). For instance, the table “4” which is described in (Gross 1975), is associated with verbs of the transitive structure **N0 V N1**, in which **N0** (the subject) is a Human noun, and **N1** (the object) is an abstract noun or a sentence, e.g.:

Jean abandonne la pose
 Il abandonne la compétition
 Elles abandonnent le domicile conjugal

Therefore, in NooJ all the entries of table 4 are associated with the explicit properties: **NOVDET1N1** (to state that the verb enters in the transitive syntactic structure), **N0=Hum** and **N1=Abst**.

When a lexicon-grammar table has been adapted to NooJ, displaying it in a list view is straightforward: each row of a table corresponds to a line in the list; non-negative cells are prefixed with a character “+”, e.g. “+NOVDET1N1”; textual cells are transcribed into a pair Property=Value, e.g. “+DET1=la”; negative cells are simply ignored. Thus, the corresponding NooJ list dictionary related to the lexicon-grammar table given in Figure 1 seems like:

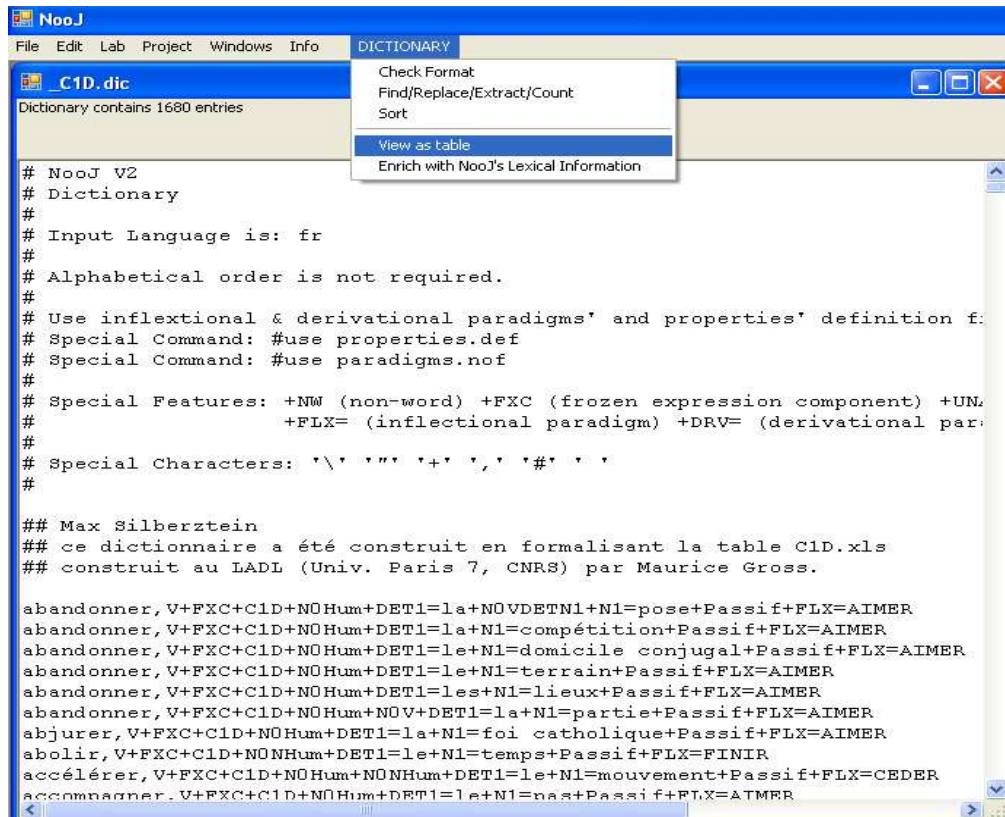


Figure 2: A list view for a lexicon-grammar table

The NooJ versions of DELA dictionaries will allow lexicographers to associate each word with corresponding derivational variants based on the integration of lexicon-

grammar tables. Morphological derivations will then be described precisely for each entry, For instance, lexical entries could seem to:

donner,V+Tr+FLX=Aimer+DRV=RE+DRV=ABLE:Artiste

Linking derived forms in the dictionary will allow for a much lighter dictionary: for instance, if the feature “+DRV=RE” is associated with the verb “donner”, then it becomes unnecessary to add a lexical entry for “redonner”. At the same time, the presence of “+DRV=ABLE” feature will avoid adding the adjectival form “donnable” in the dictionary. The generated adjective will be automatically inflected using the inflectional paradigm “:Artiste” that gives the masculine| feminine, singular| plural forms

2.4 Multi-fields dictionaries

The number of fields of NooJ dictionaries is no longer limited to one! NooJ dictionaries can contain entries associated with a “super-lemma”, that can be an orthographical variant, the translation in another language, a synonymous entry or an hyperonym.

For instance, consider the following lexical entries:

ONU,Organisation des Nations Unies,N+Org

czar,tsar,N+Hum+FLX=Crayon

The first entry (ONU) is associated with super-lemma “United Nations”; it does not inflect. This entry is similar to a INTEX/DELACF entry. The second entry (czar) is associated with super-lemma “tsar”; it inflects according to the paradigm “Crayon” (i.e. takes an ‘s’ in the plural).

Being able to associate words with super-lemmas, i.e. words that do not necessarily correspond to their inflectional lemma (“czar” is czars’s lemma, not “tsar”) opens up a new range of applications for NooJ.

3 Formal representation of electronic dictionaries

Finite-State automata (FSA) are used in a variety of Natural Language Processing (NLP) applications. In particular, they provide efficient storage and retrieval of finite sets of strings over a finite alphabet, and thus can be used to represent the vocabulary of a language. The use of automata could be extended to a use of Finite-State Transducers (FSTs) that allow users to associate each element of a vocabulary with some information, such as morpho-syntactic information (POS, Gender, Number, etc.), syntactic and semantic information (e.g. transitive, Human, etc.), or more complex, such as the term’s translation in other languages, a set of synonymous expressions, etc.

The purpose of a lexical analysis of a text is to map tokens that occur in texts into a vocabulary, usually described in a dictionary. In most languages, tokens are inflected and/or derived word forms that need to be associated with the corresponding lexeme (dictionary entry) and some linguistic information. Thus, transducers are well adapted to perform automatic lexical analyses. FSA can be easily extended into FSTs to produce information associated with the accepted tokens: the output of the transducer is simply

affixed to the corresponding accepting states¹ of the FSA. But then, special encoding techniques are necessary to represent the sets of information produced by the FST. Since these transducers can be represented very efficiently and are associated with linear lookup methods, we are using them to represent dictionaries within the linguistic platform NooJ

The first NooJ engine (v 1.x) inherited the compilation method of INTEX dictionaries which is a two-steps process: the first step was usually performed by building a TRIE, the second one consists on linear-time minimization. Transducer minimization algorithms are quite efficient in terms of the size of their input (the TRIE): NooJ's algorithm used memory and time requirements that are linear in terms of the number of states in the input TRIE. Unfortunately, even such linear performance is not sufficient when the TRIE is much larger than the available physical memory. For instance, the Hungarian dictionary, which contains over 50,000 entries, produces a 130+ million word form TRIE that contained over 100 million states (each state requires 10 bytes in average).

NooJ's linguistic units are recognized by looking up dictionaries, using morphological grammars to parse word forms as well as using syntactic grammars to parse phrases. Each of the recognized/matching sequence of the text is then associated with a complex string of information codes: its lemma, its syntactic category (e.g. Noun), inflectional codes (e.g. masculine, plural, Past Participle, etc.), syntactic codes (e.g. "transitive verb"), distributional codes (e.g. "Human") and semantic domains (e.g. "medical term").

For instance, here are three typical analyses of three word forms:

```
rats , rat , N+Animal+p  
chats , chat , N+Animal+p  
chiens , chien , N+Animal+p
```

Note that these three word forms are associated with the same information string (Noun, Animal, Plural) but their lemmas are different: "rat" for "rats", "chat" for "chats" and "chien" for "chiens". Adding these word forms, associated with their information, in an automaton could generate the following FSA that looks like a TRIE since these different analysis would lead to different transitions in the resulting transducer. In order to unify these three analyses, we replace each lemma with a morphological command based on use of the deletion operator with the special code (for "Backspace") that computes it from the word form. In other words, in order to compute the lemma "cat" from the word form "cats", we delete the last letter of the entry which corresponds to the command "".

This encoding allows us to associate the three word forms "chats", "rats" and "chiens" with a unique analysis:

```
rats , <B> , N+Animal+p  
cats , <B> , N+Animal+p  
cows , <B> , N+Animal+p
```

Thanks to this encoding, a large number of word forms can be associated with the same exact analysis. As all analyses are stored in a hash table, the size of the resulting hash

¹ An accepting state (also called terminal state) is the final state of a recognized word form. It's usually represented by a double circle (see Figure3)

table is reduced significantly, and, more importantly, a large number of input strings such as “chats” and “chiens” lead to a small number of common terminal states, that triggers a very efficient minimization process:

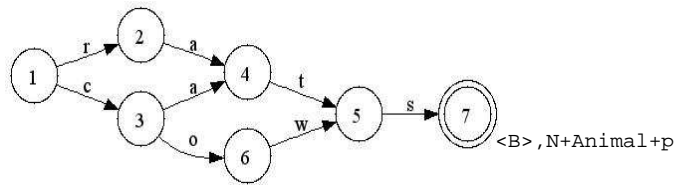


Figure 3: The result of adding the three words (rats, chats and chiens) using a compression function for related set of information. Note that the suffix “s” produces a single information string for added word forms.

This encoding solution works very well when the difference between word forms and their lemma is located only in their suffix, which is the case for English and Romance Languages. NooJ can also be used to formalize prefixations. For instance, a NooJ dictionary can be used to produce the following analyses:

relèvent , lever , V+PR+3+p
 repèsent , peser , V+PR+3+p

If we use the original algorithm to compute the lemma (e.g. lever) from the word form (relèvent), we get the following encoding:

relèvent , <B8>lever , V+PR+3+p
 repèsent , <B8>peser , V+PR+3+p

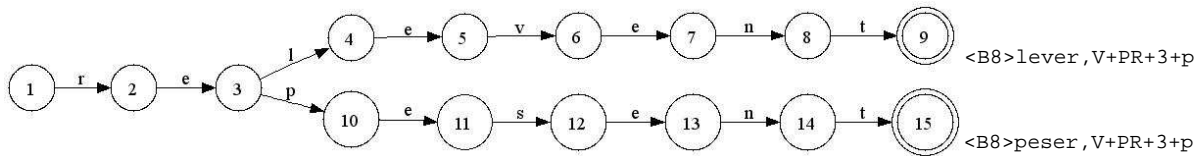


Figure 4: The result of adding the two words (distrusting, dismounting) with the associated set of information. The resulting FST is a TRIE.

In consequence, the two word forms “relèvent” and “repèsent” (as well as a large number of word forms, such as “recèpent”, “recèlent”, etc.) will have to be analyzed differently, even though their analyses are very similar.

This encoding uses only the operator (delete last letter) to compute the common prefix of the word form and the lemma. Then, it concatenates the resulting prefix with the remaining suffix of the lemma. For instance, to link the word form "had" to its lemma "have", it computes the command "ve": delete the last letter of the word form, then add the suffix "ve".

This algorithm is well adapted to languages for which inflectional and derivational morphology is performed by adding suffixes to lemmas, such as English, French and Romance languages. However, it produces poor results for other languages such as

Semitic ones (including Arabic and Hebrew), where basic inflectional and derivational rules modify prefixes and infixes massively.

As a consequence, NooJ's dictionaries for these languages were represented by FSTs that were largely TRIEs: the standard minimization process could not do much to prevent the FST to grow out of control.

In contrast with its first version inherited from INTEX, the actual NooJ engine (v 2.x) is equipped with a new incremental single step construction algorithm that adds new strings one by one and minimizing the resultant transducer incrementally in replacement to the traditional two-steps method – TRIE construction + minimization – used to build morphological lexicon in NooJ v1.x. The key of the new algorithm, which constructs the minimized FST sequentially, is a robust encoding technique that links prefixed, suffixed as well as infixed forms to their lemma in a unified way. The new lexical engine leads to very compact dictionaries, even for languages that have a “heavy” morphology, such as Hungarian and Arabic (Mesfar, Silberztein, 2008).

The newly proposed encoding routine computes a series of NooJ's morphological operators (and no longer only) to automatically encode each word form's analysis.

The word forms of Figure 4 are then analyzed in a unified way:

| Forms | Analysis | Morphological operations |
|-------------------------------------|----------------------|---|
| <i>relèvent,</i> <i>repèsent</i> | <B2>r<LW><S2><R>e<S> | <ul style="list-style-type: none"> - <B2>: delete the 2 last characters; - r: add “r” letter; - <LW>: move to beg. of word; - <S2>: delete 2 next letters; - <R>: skip one letter; - e: add “e” letter; - <S>: delete the next letter. |

Table 1: A unified morphological analysis

The resulting FST is represented as follows:

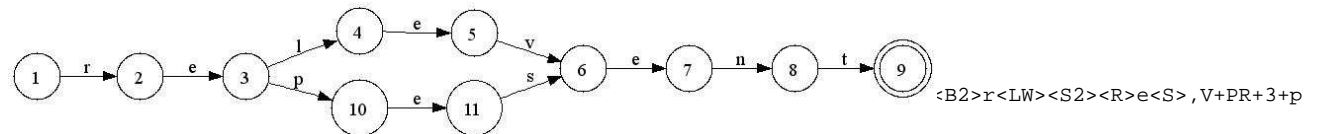


Figure 5: FSA after unification analysis

The new analysis can be shared by many word forms, hence these word forms lead to a common terminal state in the dictionary's FST, which in turn triggers a very efficient minimization process. Moreover, the hash table that stores all different analyses is reduced in size. While more complex to read, the new analysis unifies a large number of verb conjugation paradigms: (repèsent => peser), (relèvent => lever), (recèpent => ceper), (resèment => semer), etc.

So, in practice, even dictionaries for Romance languages benefit enormously from the new algorithm. These benefits are much more spectacular with Hungarian: the dictionary that contains 130+ million entries is stored into a 5-million state FST and with Arabic

that makes a heavy use of prefixes: for instance, from the lexical entry "كَتَبَ" (kataba – to write), we get 122 word forms, including the following ones:

"أَكْتُبُ" (áakotubu – I write) => <LW><S2><R>a<S><R>a<S><R>a (rather than <B8>كَتَبَ)

"يَكْتُبَانِ" (yakotubaāni – They write[dual form]) => <B3><LW><S2><R>a<S><R>a<S> (rather than <B11>كَتَبَ)

"كَتَبَا" (katabaā - They wrote [dual form]) =>

Conclusion

NooJ's system of dictionaries is a significant enhancement of INTEX's. On the one hand, unifying DELAS, DELAF, DELAC and DELACF is a significant simplification of the former INTEX system, and results in a more powerful system thanks to its embedded morphological capabilities, as well as its possibility to associate entries with "superlemmas". On the other hand, the ability to manage both DELA-type dictionaries and lexicon-grammar tables in a unified way ends the artificial dichotomy that has existed for years, between two types of activities: users can now display their dictionaries as they wish, in table or list form, and can access both morphological and syntactic properties at the same time.

We also described the NooJ formal representation of morphological lexicon within its version v1.x inherited for INTEX and then its actual version v 2.x. While the inherited algorithm processes in two steps method – TRIE construction + minimization, the second one constructs the finite state transducer sequentially and uses an encoding technique that links prefixed infixed and suffixed forms to their lemma in a unified way.

With the help of several NooJ users, it already includes resources for a dozen languages: Acadian, Arabic, Armenian, Bulgarian, Catalan, Chinese, English, French, Hebrew, Hungarian, Italian, Latin, Polish, Spanish and Portuguese. For instance, we give in Appendix A some figures about the Arabic dictionaries.

See the WEB site: www.nooj4nlp.net to download the freeware software, its resources and documentations.

References:

Courtois B., Silberztein M. (1990), "Dictionnaires électroniques du français ". Langue française N° 87. Larousse :Paris

Gross, M. (1975). Méthodes en Syntaxe (414 pages). Hermann: Paris.

Koeva S., Mihov S. (2005). "Semantic Relations in INTEX". In *INTEX pour la Linguistique et le traitement automatique des langues (2)*. Les Cahiers de la MSH Ledoux. Presses Universitaires de Franche-Comté: Besançon.

Leclère C. (2002), "Organization of the Lexicon-Grammar of French verbs". In *Linguisticae Investigationes*. John Benjamins Publishing Company.

Mesfar S.(2006) "Standard Arabic formalization and linguistic platform for its analysis". In proceedings of Arabic NLP/MT conference, London, England (2006)

Mesfar S, Silberztein M. (2008) “Transducer minimization and information compression for NooJ dictionaries”. (forthcoming)

Silberztein M. (2005), "NooJ's dictionaries". In proceedings of LTC 2005, Poznan, Poland

Silberztein M. (2004). “NooJ: an oriented object approach”. In *INTEX pour la Linguistique et le traitement automatique des langues*. Les Cahiers de la MSH Ledoux. Presses Universitaires de Franche-Comté: Besançon.

Appendix A:

| Dictionary | Number of lemmas | Number of inflected forms |
|-------------|------------------|---|
| Verbs | 10 500 | 2 734 122 forms = 1 290 795 verbs + 1 443 327 deverbals |
| Nouns | 15 000 | 280 267 forms |
| Adjectives | 4 600 | 163 866 forms |
| First names | 12 400 | |
| Toponyms | 5 300 | |