Joe Loves Lea: Transformational Analysis of Direct Transitive Sentences

Max Silberztein(✉)

ELLIADD, Université de Franche-Comté, Besançon, France
max.silberztein@univ-fcomte.fr

Abstract. NooJ is capable of both parsing and producing any sentence that matches a given syntactic grammar. We use this functionality to describe direct transitive sentences, and we show that this simple structure of sentence accounts for millions of potential sentences.

Keywords: Nooj · Syntactic analysis · Transformational analysis · Transformational grammar

1 Introduction

NooJ allows linguists to formalize various types of linguistic description: orthography and spelling, lexicons for simple words, multiword units and frozen expressions, inflectional and derivational morphology, local, structural and transformational syntax. One important characteristic of NooJ is that all the linguistic descriptions are reversible, i.e. they can be used both by a parser (to recognize sentences) as well as a generator (to produce sentences). (Silberztein 2011) and (Silberztein 2016) show how, by combining a parser and a generator and applying them to a syntactic grammar, we can build a system that takes one sentence as its input, and produce all the sentences that share the same lexical material with the original sentence.

Here are two simple transformations[1]:

– [Pron-0] Joe loves Lea = He loves Lea
– [Passive] Joe loves Lea = Lea is loved by Joe

The second one can be implemented in NooJ via the following grammar:

This graph uses three variables $NO, $V and $N1. When parsing the sentence *Joe loves Lea*, the variable $N0 stores the word *Joe*, $V stores the word *loves* and $N1 stores *Lea*. The grammar's output "$N1 is $V_V+PP by $N0" produces the string *Lea is loved by Joe*.

Note that morphological operations such as "$V_V+PP", operate on NooJ's Atomic Linguistic Units (ALUs) rather than plain strings; in other words, NooJ knows that the word form *loves* is an instance of the verb *to love* and it can produce all the conjugated

[1] I am using the term *transformation* as in (Harris 1968): an operator that links sentences that share common semantic material, as opposed to (Chomsky 1957) whose transformations link deep and surface structures.

T. Okrut et al. (Eds.): NooJ 2015, CCIS 607, pp. 55–65, 2016.
DOI: 10.1007/978-3-319-42471-2_5

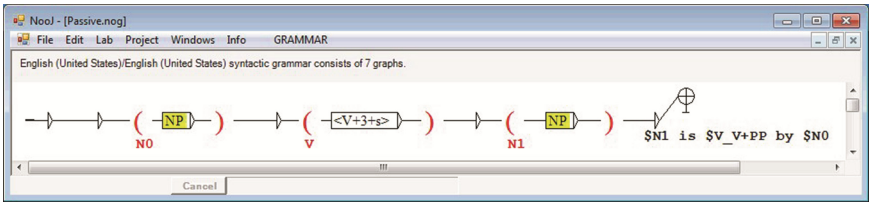

Fig. 1. The [Passive] transformation

and derived word forms from this ALU (e.g. "loving", "lovers"). Here, $V_V+PP takes the value of variable $V (*loves*), lemmatizes it (*love*), produces all its verb forms and selects the ones that have property +PP (i.e. Past Participle) to get the result *loved*.

One application of this rewriting system is Machine Translation, whereas one grammar recognizes sentences in one input language, and produces the corresponding "rewritten" sentences in another language, see for instance (Barreiro 2008) for Portuguese-English translation, (Fehri et al. 2010) for Arabic-French translation and (Ben et al. 2015) for Arabic-English translation.

As (Silberztein 2016) has shown, any serious attempt at describing a significant part of a language will involve the creation of a large number of elementary transformations (Fig. 2):

[Pron-0]	*Joe loves Lea = He loves Lea*
[Pron-1]	*Joe loves Lea = Joe loves her*
[Pron-2]	*Joe gives an apple to Lea = Joe gives her an apple*
[Preterit]	*Joe loves Lea = Joe loved Lea*
[Impfct]	*Joe loves Lea = Joe has loved Lea*
[Futur]	*Joe loves Lea = Joe will love Lea*
[Cond]	*Joe loves Lea = Joe should love Lea*
[Passive]	*Joe loves Lea = Lea is loved by Joe*
[Negation]	*Joe loves Lea = Joe does not love Lea*
[Cleft-0]	*Joe loves Lea = It is Joe who loves Lea*
[Cleft-1]	*Joe loves Lea = It is Lea that Joe loves*
[Question-0]	*Joe loves Lea = Who loves Lea?*
[Question-1]	*Joe loves Lea = Who does Joe love?*
[Nom-0]	*Joe loves Lea = Joe is in love with Lea*
[Nom-V]	*Joe loves Lea = Joe feels love for Lea*
[Nom-1]	*Joe loves Lea = Lea is Joe's love*
…	

Fig. 2. Elementary transformations

Previously, each of these transformations would have to be described by two NooJ grammars, because each pair of sentences involve two opposite computations: we want to produce not only *Lea is loved by Joe* from the sentence *Joe loves Lea*, but also the sentence *Joe loves Lea* from the sentence *Lea is loved by Joe*, and the grammars that can perform the reverse operation is different, as we can seen in the following figure (Fig. 3):

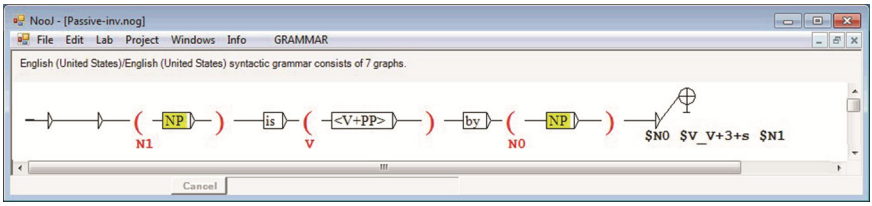

Fig. 3. The [Passive-inv] transformation

(Silberztein 2016) shows that it is not always possible to combine elementary transformations with each others in order to take account of all complex sentences. For instance, it would not be possible to use the elementary graphs [Cleft-1], [Passive] and [Neg] to produce the sentence *It is Lea who is not loved by Joe* from the original sentence *Joe loves Lea* without modifying them profondly, as the graph in Fig. 1 would not parse the intermediary sentence *Joe does not love Lea*.

The solution described by (Silberztein 2016) is based on the fact that all NooJ's linguistic resources are reversible, i.e. they can be used both to parse texts and also to produce them. For instance, NooJ's morphological grammars can be used to recognize and analyze word forms, but also to produce lists of forms in the form of a dictionary, see following figure (Fig. 4).

NooJ's command GRAMMAR > Generate Language automatically constructs the dictionary that contains all the word forms recognized by the morphological grammar,

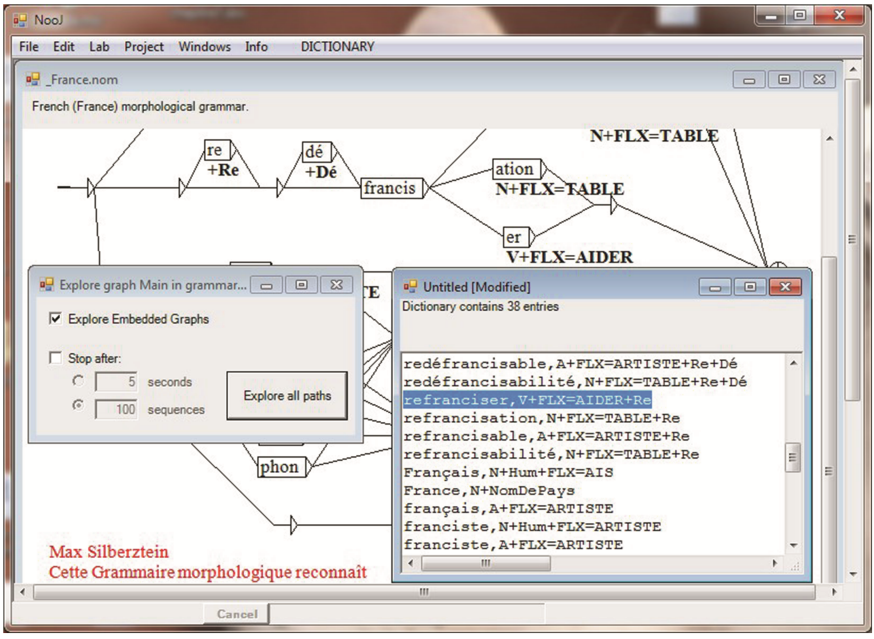

Fig. 4. Generate all the word forms derived from *France*

and associates each of the word forms with the corresponding grammar output. In the resulting dictionary, the linguistic information associated with each lexical entry corresponds in a sense to its linguistic analysis: analyzing the word form *refranciser* as a verb with the **+Re** (repetition) feature is similar to analyzing the sentence *Joe does not love Lea* with the **+Neg** (negation) operator. Our goal is then to construct a syntactic grammar that represents all the sentences transformed from the elementary sentence *Joe loves Lea*.

2 A Simple Grammar

I have constructed a simple grammar that contains three sub-grammars:

– a grammar that recognizes declarative sentences, e.g. *Joe cannot stop loving Lea*
– a grammar that recognizes interrogative sentences, e.g. *Does Joe love Lea?*
– a grammar that recognizes noun phrases, e.g. *Joe's love for Lea*.

2.1 Declarative Sentences

The graph **Declarative** shown in Fig. 5 recognizes the simple sentence *Joe loves Lea* (in the path at the very top), as well as over 1 million variants.

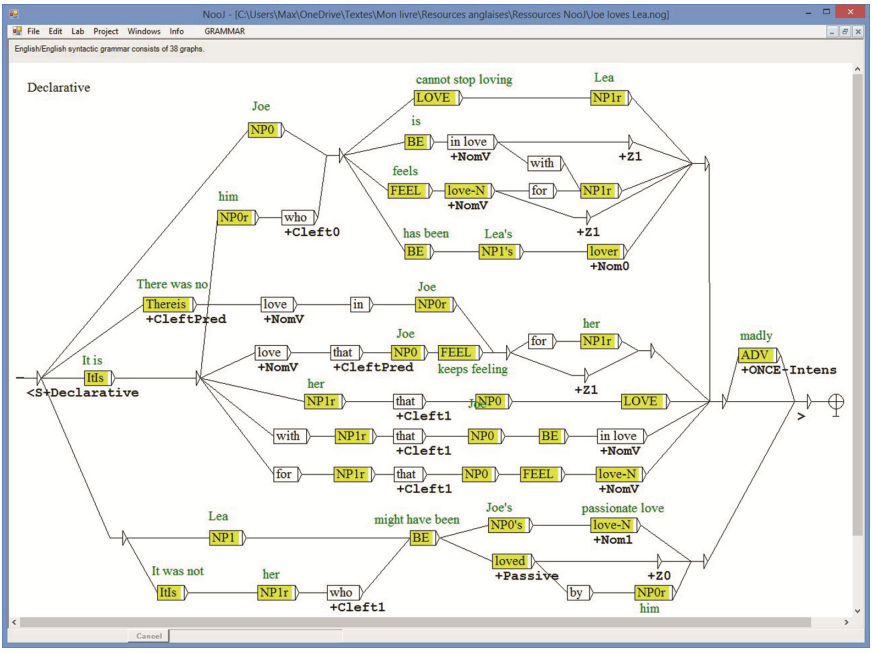

Fig. 5. Declarative sentences transformed from *Joe loves Lea*

– Three nominalizations:

Joe was in love with Lea (+NomV), *Joe did no longer feel any love for Lea* (+NomV), *Joe is Lea's lover* (+Nom0), *Lea has been Joe's love* (+Nom1)

– Three extractions:

It is not Joe who loves Lea (+Cleft0), *It is Lea that Joe loved* (+Cleft1), *It was love that Joe felt for Lea* (+CleftPred)

– Three pronouns:

He loves Lea (+Pron0), *Lea is loved by him* (+Pron0), *Joe is in love with her* (+Pron1), *She is loved by Joe* (+Pron1), *Joe feels something for Lea* (+PronPred).

– Three elisions:

Lea is loved (+Z0), *Joe is in love* (+Z1), *Joe feels for Lea* (+ZPred).

– Five intensive adjectives and adverbs:

Joe feels little love for Lea (+Intens0), *Joe loves Lea a lot* (+Intens1), *Joe loved Lea very much* (+Intens2), *Lea was Joe's passionate love* (+Intens3), *Joe is madly in love with Lea* (+Intens4)

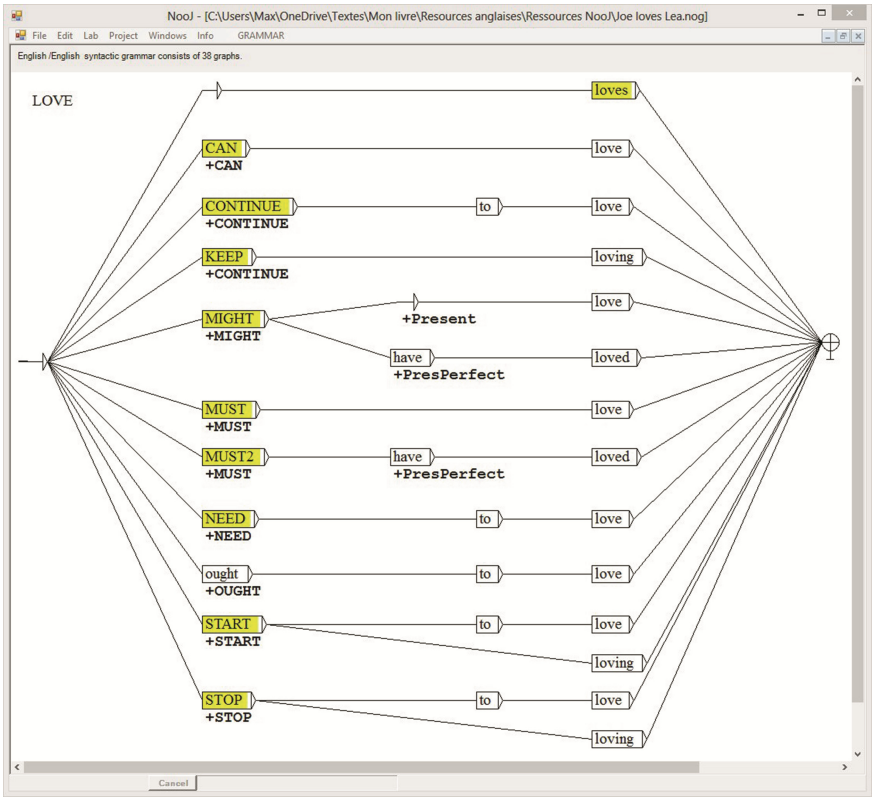

Fig. 6. The verb group

– Two aspectual adverbs:

Joe no longer loves Lea (+STOP), *Joe still loves Lea* (+CONT),

– In the graph **LOVE** shown in Fig. 6, the embedded graph **loves** recognizes all conjugated forms of *to love* (*loves, loved, has loved*, *was loving*) as well as the corresponding negative (*doesn't love*) aspectual (*is still loving*) and emphasis (*does love*) sequences.
– The graphs **CAN**, **MIGHT**, **MUST**, **MUST2**, **NEED** recognize modal sequences such as *could no longer love, must have loved, needs to love*, etc. The embedded graphs **CONTINUE**, **KEEP**, **START** and **STOP** are used to recognize aspectual sequences such as *kept on loving*, *started to love*, did not *stop loving*, etc.

By exploring all the paths in the Declarative graph, NooJ produces over 1 million declarative sentences (Fig. 7).

Similarly, the graph **NounPhrase** represents over 500,000 noun phrases such as *Lea's lover* (+Nom0), *Joe's mad love for Lea* (+NomV) and *Joe's love* (+Nom1), etc.

The graph **Interrogative** represents over 3 million questions such as: *Who used to love Lea? When did Joe's love for Lea end? Why could Joe no longer love Lea? How come Joe still loves Lea?*, etc.

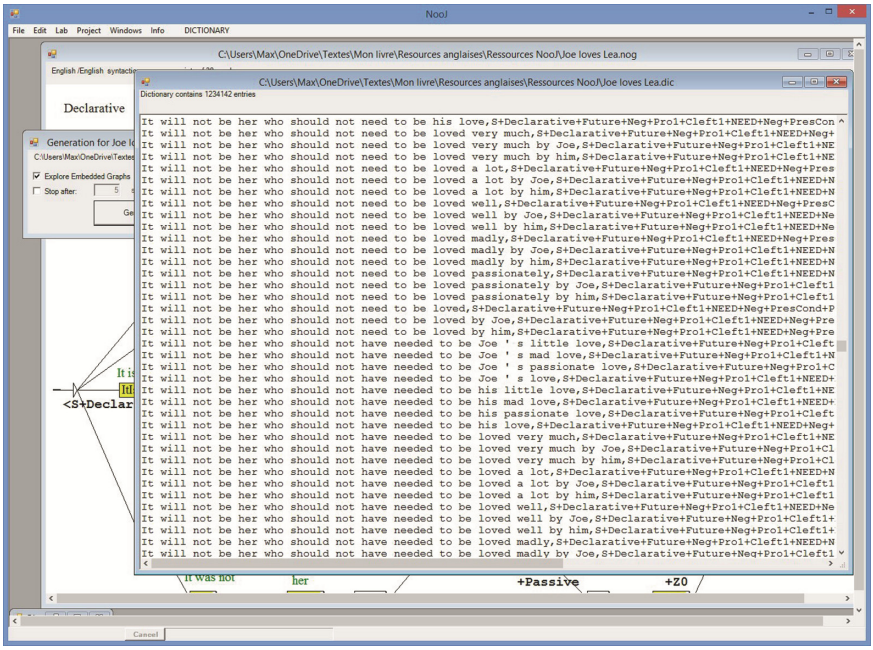

Fig. 7. The **Declarative** grammar represents over 1 million sentences

3 Generalizing the Grammar

If we want to construct a syntactic grammar that recognizes all direct transitive sentences such as *My cousin ate an apple*, we can start by replacing *Joe* and *Lea* with a description of Noun Phrases, and *love* with any direct transitive verb.

Such a grammar would indeed recognize any sentence of structure NP V NP successfully; however it would be useless as a transformational grammar because it would not restrict the elements of the sentence. For instance, it would recognize the three sentences:

Lea is seen by Joe. The busy teacher dropped the broken pen. Joe saw Lea.

but it would not be able to tell that the first and the third sentences are linked by a transformation, whereas the second one is unrelated. Moreover, we want the grammar to process the two following sentences as two unrelated sentences:

Joe loves Lea. Joe is loved by Lea.

In order to produce the sentence *Lea is loved by Joe* (and reject sentence *Joe is loved by Lea*) from the first one, we need to index the elements of the sentence:

$NP_0\ V\ NP_1 = NP_1$ *is V-pp by* NP_0

and then set NP_0, V and NP_1 respectively to *Joe*, *love* and *Lea*. To do that, we use NooJ's global variables.

The new grammar is very similar to original grammar *Joe loves Lea*. As a matter of fact, the main Declarative graph in the new grammar is almost identical to the one shown

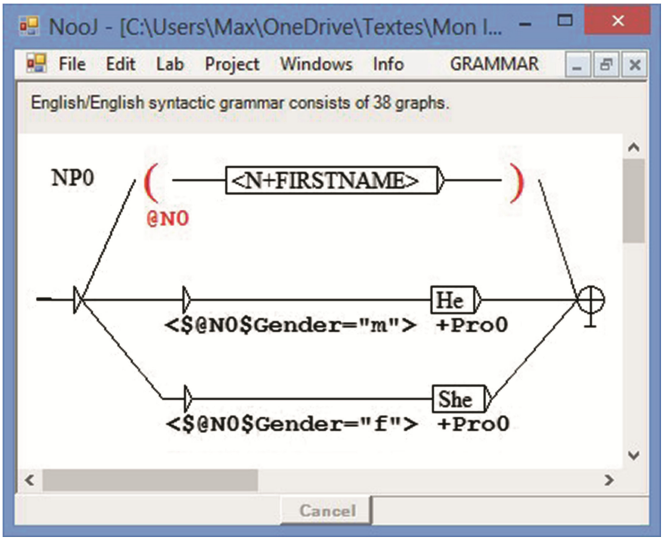

Fig. 8. A generic grammar to represent the main verb of a direct transitive sentence.

in Fig. 5[2]. However, we modified the graphs that contain the actual lexical material, as seen in Figs. 8 and 9.

The new version of the **NP0** graph sets the global variable @N0 and uses a lexical contraint on the gender of @N0 in order to produce the correct pronoun (i.e. *He* for *Joe* and *She* for *Lea*). The corresponding **NP0r** graph produces the pronouns *him* (instead of *He*) and *her* (instead of *She*) to take account of the object complements.

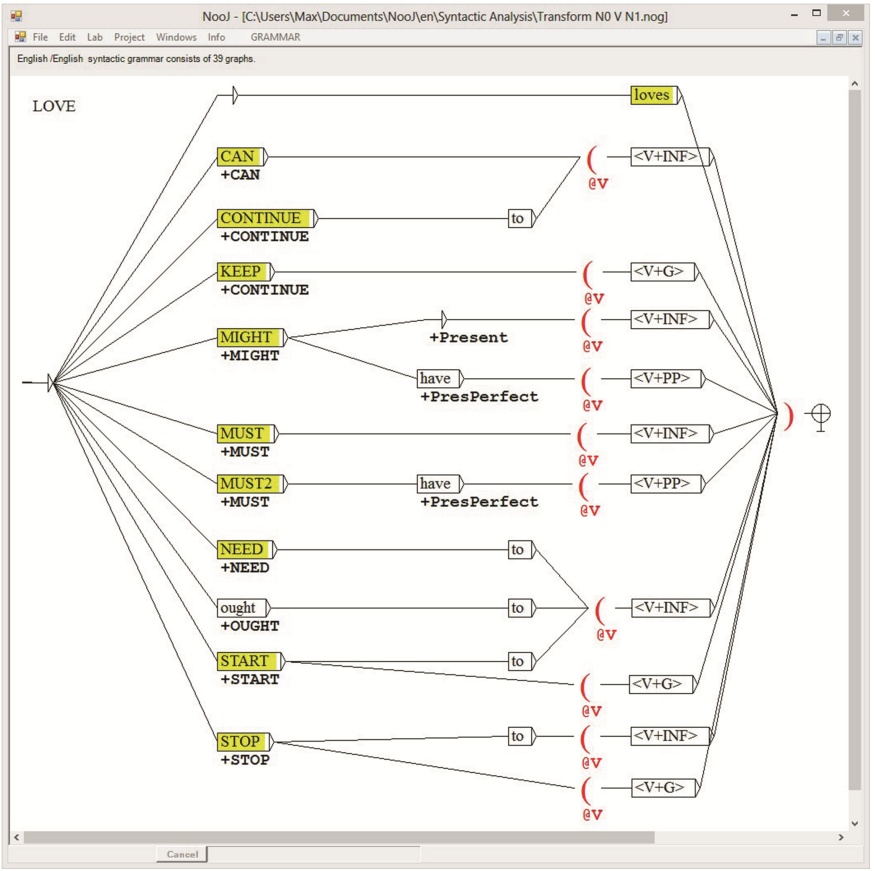

Fig. 9. A generic grammar to represent the main verb of a direct transitive sentence.

Similarly, the new version of the **LOVE** graph sets the global variable @V and replaces all conjugated forms of the verb to love with a syntactic symbol such as <V>, <V+PP>, <V-G>, etc.

[2] We have just replaced the regular nodes labeled with the word form "love" with auxiliary nodes that call embedded graphs in which the conjugated forms of *to love* are replaced with syntactic symbols such as <V>.

Nominalizations. Remember that the original grammar takes account of three nominalizations:

Joe was in love with Lea (+NomV), *Joe did no longer feel any love for Lea* (+NomV), *Joe is Lea's lover* (+Nom0), *Lea has been Joe's love* (+Nom1)

In the original grammar, these three nominalized forms were described in graphs **love-N**, **lover** and **love-0**. The same graphs have been modified as follow (Fig. 10):

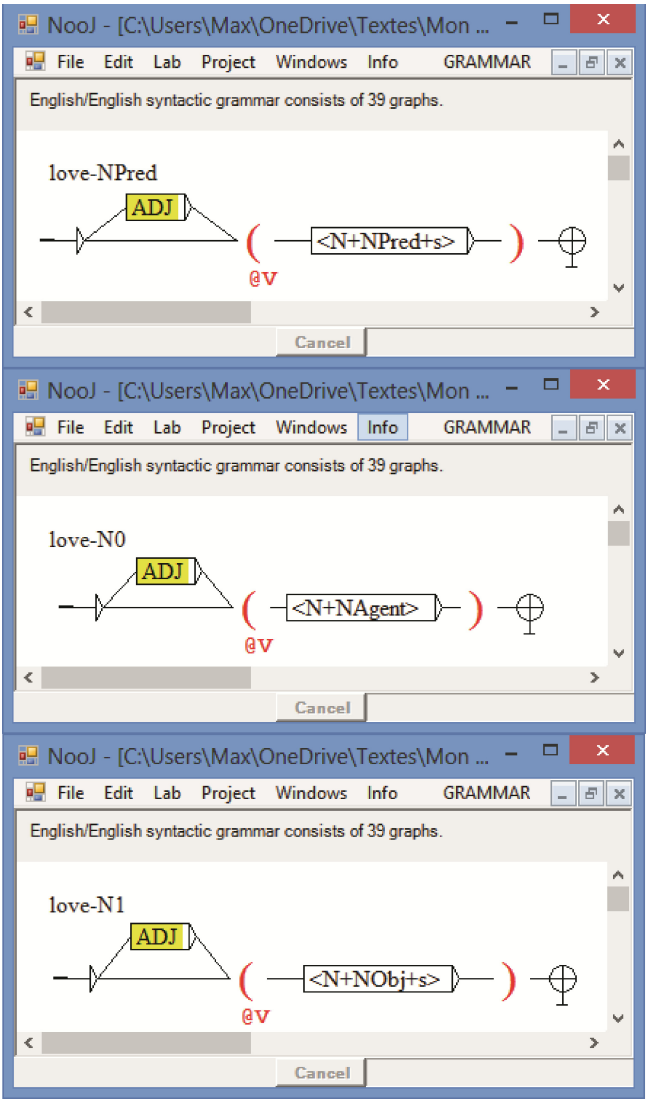

Fig. 10. Three nominalizations

Note that the variable @V is used to store the predicate of the sentence, whether it is a verb or if it has been derived as a noun. In order for each of these graphs to be activated, the initial verb needs to have the corresponding derivational property: +NPred (Predicate is nominalized), +NAgent (Subject is nominalized) and +NObj (Object is derived); these three properties are produced by the corresponding derivational DRV paradigms applied to lexical entry, e.g.:

```
love+FLX=LOVE+DRV=PRED_ID+DRV=AGENT_R:TABLE
+DRV=OBJECT_ID:TABLE
PRED_ID = <E>/N+NPred;
AGENT_R = r/N+NAgent;
OBJECT_ID = <E>/N+NObj;
```

Reciproquely, if a given verb does not accept one of these derivations, then the sentences that contain this derivation will just not be produced. For instance, if we start from sentence: *Mary sees Helen*, only 600,000 declarative sentences will be produced, because the verb to see does not accept any of the three derivations:

*Mary sees Helen = *Mary is a seer*
*Mary sees Helen = *Mary (does | feels) some seing for Helen*
*Mary sees Helen = *Helen is Mary's see*

The list of features produced with each transformed sentence represents its transformational analysis. For instance, the resulting sentence "Mary couldn't see Helen" is associated with the features S+Declarative+CAN+Preterit+Neg (Fig. 11).

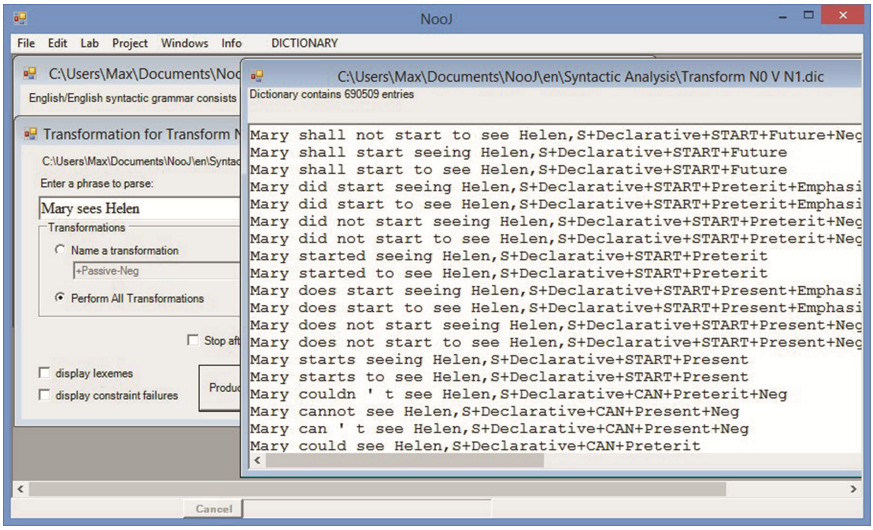

Fig. 11. Produce all the sentences transformed from *Mary sees Helen*

4 Conclusion

In this paper, I have presented a simple grammar capable of recognizing and producing a large number of sentences that are derived by transformations from the initial elementary sentence *Joe loves Lea*. This accounts for approximately 4 million sentences, including over 1 million declarative sentences, 500,000 noun phrases and over 3 million questions.

Modifying this grammar so that it can recognize any direct transitive sentence and produce the corresponding transformed sentences is suprisingly easy: it involves the replacement of word forms with more general syntactic symbols (e.g. replace *loved* with <V+PP>) and global variables that store the three elements of the sentence @N0, @V and @N1 and then provide the reference for the arguments, wherever they are located in the sentence.

As opposed to the traditional point of view on transformational grammars, the system presented here does not require linguists to implement a new level of linguistic description that would be different from the syntactic structural level: no need to implement specific transformational operators nor design a complex set of mechanisms to process chains of transformations.

References

Barreiro, A.: ParaMT: a paraphraser for machine translation. In: Teixeira, A., de Lima, V.L.S., de Oliveira, L., Quaresma, P. (eds.) PROPOR 2008. LNCS (LNAI), vol. 5190, pp. 202–211. Springer, Heidelberg (2008)

Ben, A., Fehri, H., Ben, H.: Translating Arabic relative clauses into English using NooJ platform. In: Monti, J., Silberztein, M., Monteleone, M., di Buono, M.P. (eds.) Formalising Natural Languages with NooJ 2014, pp. 166–174. Cambridge Scholars Publishing, Newcastle (2015)

Fehri, H., Haddar, K., Ben Hamadou, A.: Integration of a transliteration process into an automatic translation system for named entities from Arabic to French. In: Proceedings of the NooJ 2009 International Conference and Workshop, pp. 285–300. Centre de Publication Universitaire, Sfax (2010)

Harris, Z.: Mathematical Structures of Language. Interscience, New York (1968)

Silberztein, M.: Automatic transformational analysis and generation. In: Gavriilidou, Z., Chatzipapa, E., Papadopoulou, L., Silberztein, M. (eds.) Proceedings of the NooJ 2010 International Conference and Workshop, pp. 221–231. Univ. of Thrace, Komotini (2011)

Silberztein, M.: Language formalization: the NooJ's Approach. Wiley Eds. (2016)