GenDR: A Generic Deep Realizer with Complex Lexicalization

François Lareau, Florie Lambrey, Ieva Dubinskaite, Daniel Galarreta-Piquette, Maryam Nejat
OLST, Département de linguistique et de traduction, Université de Montréal
C.P. 6128, succ. Centre-Ville, Montréal QC, H3C 3J7, Canada
{francois.lareau, florie.lambrey, ieva.dubinskaite, daniel.galarreta-piquette, maryam.sadat.nejat}@umontreal.ca

Abstract

We present a generic deep realizer called GenDR, which takes as input an abstract semantic representation of predicate-argument relations, and produces corresponding syntactic dependency structures in English, French, Lithuanian and Persian, with the possibility to fairly easily add more languages. It is generic in that it is designed to operate across a wide range of languages and applications, given the appropriate lexical resources. The focus is on the lexicalization of multiword expressions, with built-in rules to handle thousands of different cross-linguistic patterns of collocations (intensifiers, support verbs, causatives, etc.), and on rich paraphrasing, with the ability to produce many syntactically and lexically varied outputs from the same input. The system runs on a graph transducer, MATE (Bohnet et al., 2000; Bohnet and Wanner, 2010), and its grammar design is directly borrowed from MARQUIS (Lareau and Wanner, 2007; Wanner et al., 2009; Wanner et al., 2010), which we have trimmed down to its core and built upon. The grammar and demo dictionaries are distributed under a CC-BY-SA licence (http://bit.ly/2x8xGVO). This paper explains the design of the grammar, how multiword expressions (especially collocations) are dealt with, and how the syntactic structure is derived from the relative communicative salience of the meanings involved.

Keywords: multilingual natural language generation, deep realization, lexicalization, multiword expressions, collocations

1. Introduction

Natural language generation (NLG) is one of the rare tasks in natural language processing (NLP) that is not yet completely dominated by statistical or neuronal methods. We believe the main reason for this is that for real-life applications, the output of an NLG system must often be flawless, so that extremely high precision is required, and rule-based methods still outperform other approaches in that respect. A number of domain-independent text realizers have been developed to provide an easier way of producing text automatically. While some have impressive coverage, many expect as input a syntactic structure, thus offering very little flexibility in terms of lexical choice and structure. Others take a more abstract input, offering more syntactic flexibility, but their lexicalization model is rather rigid. We propose a multilingual generic deep realizer, GenDR, that is a platform for the modeling of the semantics-syntax interface in languages. In this paper, we will show how GenDR deals with two crucial tasks: **arborization** (building a syntactic structure that reflects the semantic structure) and **lexicalization** (picking the right words to express the desired meanings).

2. Previous work

Most realizers that we know of expect an input where both lexical choice and syntactic structure have already been computed, leaving the user with two particularly complex tasks. This is the case of FUF/SURGE (Elhadad, 1993; Elhadad and Robin, 1996), RealPro (Lavoie and Rambow, 1997; CoGenTex, 1998), SimpleNLG (Gatt and Reiter, 2009), its bilingual version, SimpleNLG-EnFr (Vaudry and Lapalme, 2013) and its Spanish version, SimpleNLG-ES (Ramos-Soto et al., 2017), JSReal (Daoust and Lapalme, 2015) and its bilingual version, JSRealB (Molins and Lapalme, 2015), as well as ATML3 (Weißgraeber and Madsack, 2017). KPML (Bateman, 1996) and OpenCCG (White, 2008) both start from a more abstract representation of the text's meaning, but they tend to focus on the grammar more than the lexicon, resulting in well-formed sentences that somehow lack lexical flexibility. More recently, statistical approaches have been applied to text generation from logical forms (Basile, 2015) or semantic structures (Mille, 2014), but again, lexical choice is rather rigid. All of these realizers have a hard time producing collocations.

MARQUIS (Lareau and Wanner, 2007; Wanner and Lareau, 2009; Wanner et al., 2010) was a multilingual data-to-text system for the air quality domain that addressed the issues that we are concerned with, and its linguistic realization component has been reused in a couple of projects from different domains, namely patents (Wanner et al., 2009; Wanner et al., 2011) and football (Bouayad-Agha et al., 2011; Bouayad-Agha et al., 2012). Its lexicalization model was designed to produce natural-sounding collocations and multiword expressions, and to be as generic as possible. However, the range of collocations it was able to produce was limited to the most common patterns. The system we present here extends this coverage by a very large margin.

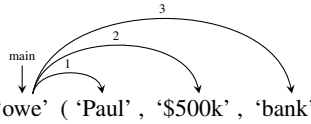
3. GenDR's architecture

GenDR runs on MATE, a graph transducer (Bohnet et al., 2000; Bohnet and Wanner, 2010). It consists of a graph grammar and some dictionaries, which we discuss below.

3.1. Input and output structures

The input to GenDR is a semantic structure *à la* Meaning-Text Theory (MTT) (Mel'čuk, 2012), which is a graph representation of first-order logical form where predicates are linked to their arguments by relations labelled with numbers indicating the arguments's position in the predicate. One of the meanings in the structure has to be flagged as the most salient meaning in the sentence. It is the dominant node of the sentence's rheme/focus (Mel'čuk, 2001) and it will be mapped to the syntactic root.

(1)



The semantic structure in (1) is a simplified visualization of the actual input structure, which is as follows:

```
structure Sem debt {
  S {
    owe {
      tense=PRES
      1-> Paul {class=proper_noun}
      2-> "$500K" {class=amount}
      3-> bank {number=SG definiteness=DEF}}
    main-> owe}}
```

The output of the system is a set of surface syntactic dependency structures (Mel'čuk, 1988). For example, with the input in (1), GenDR produces the six structures in (2).

(2) a. Paul owes $500k to the bank

b. Paul owes a debt of $500k to the bank

c. Paul has a debt of $500k to the bank

d. Paul's debt to the bank amounts to $500k

e. Paul's debt to the bank stands at $500k

f. Paul's debt to the bank totals $500k

For better readability, the sentences shown here are fully inflected and linearized, but the actual outputs are unordered trees with lemmas and grammatical features. They are to be fed to a surface realizer that computes the inflected forms and word order. GenDR focuses on the deeper tasks of lexicalization and arborization, and as one can see from this example, it offers a high level of lexico-syntactic flexibility. Between the semantic and surface syntactic levels of representation, there is a third, intermediate level: the deep syntactic structure (Mel'čuk, 1988). At this level, there are only meaningful lexemes and support verbs linked by two major types of relations: complementation (labelled with Roman numerals) and modification (labelled ATTR). Again, these structures are ordered here only for readability.

(3) a. PAUL OWE $500K BANK

b. PAUL OWE DEBT $500K BANK

c. PAUL HAVE DEBT $500K BANK

d. PAUL DEBT BANK AMOUNT $500K

e. PAUL DEBT BANK STAND $500K

f. PAUL DEBT BANK TOTAL $500K

3.2. Multilingual grammar

At its core, GenDR is a multilingual grammar. We have directly borrowed the core rules from MARQUIS (Lareau and Wanner, 2007; Wanner et al., 2009; Wanner et al., 2010), which dealt with Catalan, English, French, Polish, Portuguese and Spanish. We kept only the most basic rules, that described very general phenomena like simple lexicalization, complementation, modification, etc. Almost all of these rules are shared across languages, while a few language-specific rules model grammatical phenomena like auxiliaries, determiners and so on.

The mapping between semantic graphs and surface syntactic structures takes place in two steps, each handled by a different module of the grammar: the semantic module maps the input semantic structures onto deep syntactic structures (Mel'čuk, 2013), while the syntactic module maps deep syntactic structures to surface syntactic ones (Mel'čuk, 1988) – a layered architecture directly borrowed from MTT (Mel'čuk, 1973; Kahane, 2003; Milićević, 2006).

The semantic module contains 21 core rules, most of which were adapted from MARQUIS, and 132 bound lexicalization rules (see §5.4.) implementing lexical functions; they are documented in (Lambrey and Lareau, 2015; Lambrey, 2016). The syntactic module is much lighter, with only 20 rules, 12 of which are language-independent. Each rule models a linguistic phenomenon, and the grammar relies heavily on rich dictionaries.

4. Arborization

As shown above, one node in the input structure has to be marked as the main node, i.e., the most salient. This is because a semantic graph has no inherent hierarchy. Consider for instance the structure in (4).

(4) 'small' 'cat' 'lie' 'sofa' 'red'

Any of the predicates 'small', 'lie' and 'red' is a root. In a tree, however, there can only be one root, so we have to pick one. Picking 'small' yields *The cat lying on the red sofa is small*, while picking 'lie' produces *The small cat is lying on the red sofa*, and 'red', *The sofa on which the small cat is lying is red*. This choice can be left to GenDR, but typically one wants to control the communicative organization of the sentence by specifying the main node. The deep syntactic tree is then built in a top-down fashion, starting from this main node and using the semantic graph as a blueprint. This algorithm is inspired by (Wanner and Bateman, 1990; Wanner, 1992; Polguère, 2000) and is similar to the one used in MARQUIS. We provide a step-by-step example in §6.

1. We build the root of the syntactic tree and make it correspond to the main node of the semantic structure. In this step, we only create the node without lexicalizing it, but we do add some constraints to it. The main constraint is that it must be a verb in the indicative mood (though we could add alternative rules to handle, say nominal headlines, or languages with adjectival roots).
2. After a node has been created and constrained, we look for a lexicalization rule that can satisfy these constraints while expressing the desired meaning, cf. §5.

3. After lexicalization has taken place, we look at the edges attached to the corresponding semantic node. Edges leaving it point to its arguments, which must be realized as syntactic complements. Edges entering it lead to predicates that apply to it, which must be realized as modifiers. If we have a modifier, we create a dependent attached with the relation ATTR. If it is a complement, then we look up the government pattern (GP) of the governor in a dictionary, as explained in §5.1. The GP models the mapping between semantic, deep, and surface syntactic actants of a word. It also specifies the part of speech of the complements, as well as prepositions. It can also constrain certain grammatical features (e.g., impose a certain mood on a verbal complement) (Mel'čuk, 1995; Mel'čuk, 2014). This step creates new nodes that have not been lexicalized but that have some constraints. Now, for each of these nodes, we go back to step 2.

5. Lexicalization

Lexicalization in GenDR involves three levels of representation: semantics, deep syntax and surface syntax. The first step is to pick a deep lexical unit to express a given semanteme; this is deep lexicalization (or δ-lexicalization). It introduces meaningful words and support verbs. Then, surface lexemes are chosen to express the deep lexical units; this is superficial lexicalization (or σ-lexicalization). It introduces function words.

GenDR performs six types of lexicalization: simple lexicalization for lexemes, template lexicalization for idioms, bound lexicalization for collocations, class-based lexicalization for proper nouns, numbers and such, fallback lexicalization for unknown words, and grammatical lexicalization for function words. Though idioms and collocations are often conflated in a vague "multiword expressions" category, they really are distinct linguistic phenomena and, accordingly, are treated differently.

To make this text hopefully easier to read, we will refer to the nodes involved in the lexicalization process as α for semantic nodes, β for deep syntactic ones, and γ for the ones in surface syntax. These nodes bear a labelling feature: `sem` for α nodes, `dlex` for β nodes, and `slex` for γ nodes.

Lexicalization is supported by three types of dictionaries: a semantic and a lexical dictionary for each language, and one language-independent dictionary of lexical functions (LFs). All are feature structures represented in a straighforward JSON-like format. We will first briefly introduce these lexical resources before explaining how lexicalization takes place.

5.1. Lexical resources

The **semantic dictionary (semanticon)** of a language maps semantemes onto simple lexemes or idioms in that language. A semanteme may be mapped onto several lexical units, which yields lexical paraphrases. This is not limited to synonymy, but also applies across parts of speech (POS). For example, the meaning 'cause(x, y)' may be realized as *x causes y*, *x is the cause of y*, *y (happens) because of/due to x*, *y is due to x*, etc., so the mapping is 'cause' → {BECAUSE, DUE_{ADJ}, DUE_{ADV}, CAUSE_{N}, CAUSE_{V}, ...}.

The **lexical dictionary (lexicon)** of a language should give detailed information about every lexical unit in that language. Obviously, in practice we have to make do with a subset of that. Both simple lexemes and idioms have entries in this dictionary. An entry provides information about the POS of the word, its diathesis (mapping of its semantic to syntactic actants), its subcategorization (including any constraints it imposes on its actants: POS, preposition, mood, definiteness, etc.), and the collocations it controls. Collocations (e.g., *make a decision*) are described in the entry of the base (in this example, *decision*), which is linked to its collocates via lexical functions (e.g., $\texttt{Oper}_1$(DECISION)=MAKE, see boxed text). Currently, GenDR has basic lexical dictionaries for the ~1500 most common words in English and French (most of which have not been disambiguated yet), and demo dictionaries for Lithuanian (~180 entries from the crime news domain) (Dubinskaite, 2017) and Persian (~60 entries).

The **lexical function dictionary (LF dictionary)** describes the semantics and syntax of ~37,000 simple and (mostly) complex LFs. For example, the LF $\texttt{Oper}_1$ (used for support verbs like *make a decision*) has the description {`dpos=V, gp={1=I, L=II}`} (where *dpos* stands for *deep part of speech* and *gp* for *government pattern*). This means that the collocate returned by $\texttt{Oper}_1$ is a verb that has no (or negligible) meaning in itself (it lacks a `sem` attribute) and that takes the first argument of the base as its first syntactic actant (`1=I`) and the base itself as its second actant (`L=II`). This is equivalent to the algebraic notation $(\frac{\#}{\text{V[1,\#]}})$ proposed by (Kahane and Polguère, 2001). The LF `Magn` has the entry {`sem=Magn, dpos=(Adj|Adv), gp={L=ATTR}`}, equivalent to $(\frac{\#\text{Magn}}{\text{(Adj|Adv)[\#\^{}]}})$. For more details, see (Lambrey and Lareau, 2015; Lambrey, 2016).

Lexical functions (LFs)

Collocations tend to be instances of recurrent patterns across languages. For example, *strong preference*, *gravely ill*, *intense flavour* and *win hands down* are all instances of the same pattern where a base is intensified by a syntactic modifier (the collocate). What defines a collocation is not really its apparent lack of compositionality, but the special relationship that exists between the base and the collocate it selects. This relationship is modeled as a function. Intensification collocations, for instance, are described with the `Magn` function: `Magn`(PREFERENCE)=STRONG, `Magn`(FLAVOUR)=INTENSE, etc. Over the years, ~60 basic LFs have been identified, and they combine to form a large number of complex LFs. For detailed discussions of LFs and their use in NLP, see (Žolkovskij and Mel'čuk, 1967; Mel'čuk, 1995; Mel'čuk, 1996; Mel'čuk, 1998; Wanner, 1996; Apresjan, 2000; Kahane and Polguère, 2001; Apresjan et al., 2002; Mel'čuk, 2007; Polguère, 2007; Jousse, 2010; Mel'čuk, 2014; Lambrey, 2016; Fonseca et al., 2016b).

5.2. Simple lexicalization: lexemes

This is the most basic and most common form of lexicalization. During δ-lexicalization, we look at the entry for α.sem in the semanticon and retrieve the set of lexical units that can express this meaning. For each item in this set, we look at its POS in the lexicon and keep only the ones that match the constraint present on β, if any. We create as many output structures as there are elements left, and set β.dlex to a different lexical unit in each. This is an important source of paraphrasing. During σ-lexicalization, we just recopy β.dlex as γ.slex if it is a simple lexeme; if it is an idiom, we branch to template lexicalization.

5.3. Template lexicalization: idioms

This type of lexicalization is only relevant to σ-lexicalization and may take place regardless of how β was lexicalized during δ-lexicalization. It is triggered when the entry for β.dlex in the lexicon has an attribute idiom. Its value is a feature structure that specifies the type of idiom, the lexemes that make it up, and the syntactic relations between them. Each type of idiom is echoed by a rule in the syntactic module that builds a template subtree with placeholders that will be filled with the lexemes and relation labels found in the lexicon entry's idiom attribute.

5.4. Bound lexicalization: collocations

The most complex type of δ-lexicalization is bound lexicalization, which produces collocations. A collocate is a lexical unit that has a certain meaning only when used in the context of a specific base (some collocates being more restrictive than others). Collocations are best described as LFs in the base's entry in the lexicon, because for a lexicographer, it is much easier to think of all the collocates controlled by a common base than the other way around.

Each LF used in the lexicon must have an entry in the LF dictionary that describes the semantics and syntax of the collocation pattern it corresponds to. The information in the LF dictionary is actually a set of parameters that are used by rules of the semantic module. Similarly to how the syntactic structure of idioms is described in the syntactic module, the LF rules of the semantic module are templates with placeholders for deep lexical units and names of syntactic relations. However, these templates are a lot more intricate and diverse. We have 132 rules for bound δ-lexicalization, each corresponding to a family of LFs, so that in total, our rules, combined with the LF dictionary, implement ~26,000 LFs (some of the ~37,000 LFs described in the LF dictionary have not been implemented in the rules yet). Each LF itself is a generic pattern that may be used to describe thousands of instances of actual collocations in a given language. We do not have the space in this abstract to explain how these rules are implemented; see (Lambrey and Lareau, 2015; Lambrey, 2016). The good news is that the user does not need to understand these rules to use them. All one needs to do is list LFs in the entries of the lexicon, like so:

```
fear_n { ...
  lf={name=Magn value=great}
  lf={name=Oper1 value=have}
  lf={name=CausFunc1 value=instill}
  lf={name=Propt value="out of"} ... }
```

When GenDR encounters a meaning in the input structure that corresponds to the meaning of a LF as per the LF dictionary (say, intensification or causation), it treats it as a potential collocation. It knows where to find the base in the semantic structure because that information is in the LF dictionary. Then it looks for the corresponding LFs in the base's entry in the lexicon. If a value is specified, it uses it to produce idiomatic forms like *dirt cheap*, *sharp increase*, *bleed profusely*, etc. If no value is specified, it falls back to a generic value, selected according to the base's POS (and possibly its semantic properties), to produce generic expressions like *very cheap*, *big increase*, *bleed a lot*.

5.5. Class-based lexicalization: numbers, etc.

There are meanings for which we don't want to have an entry in the semanticon and lexicon because their number is large and their behaviour is predictable: numbers, proper nouns, dates, etc. For such meanings, we use an attribute class on α. Its value points to a matching entry for the class in the lexicon, where information on POS and grammatical features may be provided. When α.class has a value, then α.sem is copied to β.dlex and any feature specified in the lexicon for the class is copied to β. During σ-lexicalization, we copy β to γ as is.

5.6. Fallback lexicalization: unknown words

When α.sem has no entry in the semanticon, and it is not the meaning of a known LF, and α.class is not defined, then we have to guess. If there is an entry in the lexicon for this label, we suppose it was just omitted in the semanticon and proceed with simple δ-lexicalization (taking into account POS constraints that may exist on β)—in other words, meanings with trivial one-to-one δ-lexicalization need not be listed in the semanticon. If the word is not in the lexicon either, then we take a shot in the dark. We recopy α.sem to β.dlex, but we have no way of verifying that it matches the POS constraints on β. If there are such constraints however, we treat the word as if it belonged to that POS. This triggers the default GP for that POS (given in the lexicon). If there are no constraints on β, we bet it is probably a noun and treat it as such. Guessed words are flagged with a special feature in the output structures, so that they can be filtered out or sent to an external program for further processing.

5.7. Grammatical lexicalization: function words

This type of lexicalization only happens in σ-lexicalization. It introduces two types of function words: the ones that express a grammatical meaning appearing as features on β, and the ones that are imposed by the GP of a word. Auxiliaries and determiners are of the first type. They do not appear as nodes in deep syntax, but as grammatical features on the verb or noun they apply to, and they have to be introduced as an extra node in surface syntax. These words belong to closed classes with a small number of items, so we have a specific rule for each of them in a language. Governed prepositions and complementers are of the second type. They also do not appear as nodes in deep syntax. They are introduced in surface syntax as an extra node γ_f between a governing word γ_g and its dependent γ_d. The label γ_f.slex is retrieved from the GP of γ_g.slex in the lexicon.

6. Step-by-step example

In this section, we will show how the semantic graph in (1) gets realized as the deep syntactic trees in (3). The formalism used for GenDR rules is explained in full detail in the accompanying manual, as well as in MATE's manual. Introducing it here again would take too much space and bring little to the discussion, so we will only provide below a textual explanation of what the most important rules do.

6.1. Lexical information

The semanticon provides two lexicalizations for 'owe':

```
owe {lex=owe lex=debt}
```

It does not contain information on 'Paul' (a proper noun) or '$500k' (an amount), both of which are candidates for class-based lexicalization need not be in the semanticon. These nodes are marked in the semantic graph with a `class` attribute. For illustration purposes, we also ommitted 'bank' from the semanticon to trigger fallback lexicalization.

The lexicon contains the following entries:

```
owe {
  dpos=V spos=verb
  gp = { // X owes Y to Z
    1=I 2=II 3=III  // trivial diathesis
    I={dpos=N rel=subj}
    II={dpos=Num rel=dobj}
    III={dpos=N rel=iobj prep=to} } }
debt {
  dpos=N spos=noun
  gp = { // X's debt of Y to Z
    1=II 2=I 3=III  // special diathesis
    I={dpos=Num rel=ncomp prep=of}
    II={dpos=N rel=det case=GEN}
    III={dpos=N rel=ncomp prep=to} }
  lf={name=Oper1 value=have}
  lf={name=Oper13 value=owe}
  lf={name=Func2 value=amount_v_1}
  lf={name=Func2 value=stand_v_2}
  lf={name=Func2 value=total_v} }
```

There are also entries for every lexical unit linked to these two: OF, TO, HAVE, AMOUNT_V**1**, STAND_V**2**, TOTAL_V. Their content, however, is not relevant to our discussion.

The LF dictionary contains entries for each of the LFs. The ones relevant to us are the ones used in DEBT above:

```
Oper1 {  // e.g., X has a debt
  dpos = V
  gp = { 1=I L=II } }
Oper13 { // e.g., X owes a debt to Z
  dpos = V
  gp = { 1=I L=II 3=III } }
Func2 {  // e.g., the debt amounts to Y
  dpos = V
  gp = { L=I 2=II } }
```

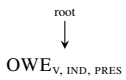
The meaning of these instructions is explained in §5.1. Note that these entries are not specific to DEBT: the description provided here for, say, $\texttt{Oper}_1$ is valid for any instance of that LF in any language (*take a walk*, *make a choice*, etc.).

6.2. Semantics ⇒ deep syntax

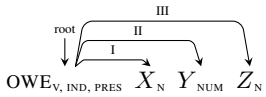
Now, let us see how (1) gets realized in deep syntax.

1. A new deep syntactic tree is created with only an empty root in it, marked as corresponding to the main node of (1), 'owe'. It is not lexicalized but it is constrained to be a verb in the indicative mood (at least in English). At this point, the output structure is:

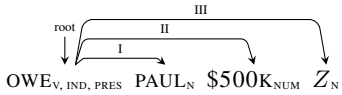
2. We try to lexicalize 'owe' in this syntactic position. The semanticon provides two competing simple lexicalizations, OWE and DEBT, only one of which is a verb and thus compatible with the constraints on the syntactic node. This yields only one simple δ-lexicalization. Now, the output structure gets updated with this label, and the tense feature is carried over:



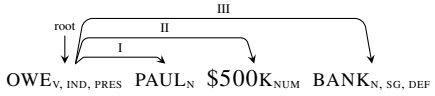
3. We now look for edges leaving 'owe' in the input structure. There are three semantic arguments, that we will realize syntactically according to the diathesis of OWE as encoded in the lexicon (`gp = {1=I 2=II 3=III}`, meaning that the first semantic argument becomes the first syntactic actant, and so on). The GP also constrains actants I and III to be nouns (`I={dpos=N}`, `III={dpos=N}`), and actant II to be a number/amount (`II={dpos=Num}`). Three more nodes are created in the output structure:



4. These nodes need to be lexicalized but none of them has an entry in the semanticon (remember we removed 'bank'), so simple lexicalization is not possible. Since 'Paul' and '$500k' bear a `class` feature, class-based δ-lexicalization applies, as explained in §4. The label of the semantic node is just copied and we check that the constrains on the node match the POS specified in the lexical entries for the classes (not shown here, but they are trivial: proper nouns are subtypes of nouns, and amounts are subtypes of numbers).

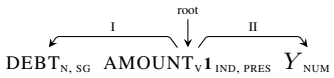


5. Finally, since we have nothing on 'bank' in the semanticon, we use fallback lexicalization. If BANK exists in the lexicon, we can match the POS. If not, then we assume it must be a noun and just copy the input label and carry its grammatical features:

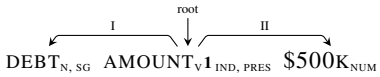


2′. At this point, we have a full deep syntactic structure. But there are alternative lexicalizations that we have not considered. Let us backtrack to step 2. We need a verbal lexicalization and the semanticon only provides

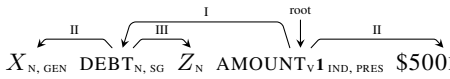
OWE, the other one being DEBT, a noun. However, the lexicon entry for DEBT contains several support verbs, with values for LFs $\texttt{Oper}_1$, $\texttt{Oper}_{13}$ and $\texttt{Func}_2$. This allows the bound lexicalization rules to apply, where a single semantic node, in this case 'owe', is realized in deep syntax by two nodes linked with a specific construction (say, DEBT + AMOUNT_V**1**). Because there are five support verbs and that all of them are compatible with the current state of the output structure, we create five copies of the structure and use a different support verb in each, so that at this point we have six output structures (these five and the simple one from step 2 above. Below, we show only the one for $\texttt{Func}_2$(DEBT)=AMOUNT_V**1**. (The `sg` feature on DEBT is a default value, and the `Num` constraint on Y comes from the GP of AMOUNT_V**1** in the lexicon.)



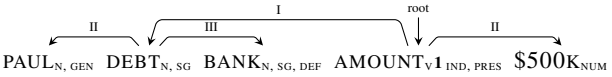
3′. We apply class-based lexicalization, as above, yielding:



4′. There are two semantic arguments of 'owe' that have not been lexicalized, so we open positions for them in the syntactic tree, as in step 3. This time, however, the lexeme controlling the diathesis is DEBT, not OWE. Its diathesis is `gp = {1=II 2=I 3=III}`, meaning that the first semantic argument becomes the second syntactic actant, and vice-versa. The GP also constrains the POS of the actants and imposes the genitive case on its second actant. So, the semantic arguments 1 and 3 of 'owe' become, respectively, the syntactic actants II and III of DEBT. It has no actant I because what could have filled this slot has already been realized as an actant of the support verb. Now, we have:



5′. Finally, class-based and fallback lexicalizations are applied, as in steps 4 and 5 above:



The treatment of $\texttt{Oper}_1$ and $\texttt{Oper}_{13}$ differs only in how some of the semantic arguments are attached to the support verb, as per the patterns described in the LF dictionary.

As one can see, arborization and lexicalization are intertwined: choosing a lexeme imposes a GP, which in turn imposes restrictions on the part-of-speech and potentially other features on its actants, hence restricting what lexemes can be chosen to fill the slot. Support verbs provide a way to fill a verbal slot with a nominal lexeme, thus giving more flexibility in lexico-syntactic choices.

Our top-down approach has a slight bias towards syntax. Every time a node is created, it is immediately constrained (either by the GP of its governor or by the rule that builds the root), limiting the range of lexemes that can be used in a given position. This is desirable of course, to avoid putting, say, a verb where a noun is needed. However, the downside is that there is no guarantee that such a lexicalization exists for a given node that satisfies the constraints, so we might end up with a dead-end structure. It is easy however to filter these out, and since the system usually produces several outputs, there is usually at least one valid output structure.

7. Conclusion and future work

GenDR is a multilingual deep realizer that provides a platform for modeling the semantics-syntax interface in languages. Its salience-driven arborization algorithm ensures high syntactic flexibility. The main strength of this system is its approach to lexicalization, especially its handling of a very wide range of collocation patterns. We have formally described the semantics and syntax of ~37,000 LFs in a dictionary, ~26,000 of which have also been implemented in grammar rules that are ready to be used. This makes it relatively easy to develop resources for new languages that produce texts with rich idiomatic expressions, as the process boils down to lexicography and requires little knowledge of formal grammars.

In lieu of an evaluation, we put this system to the test in a seminar where graduate linguistics students without prior experience in NLG had to generate texts in a domain and language of their choosing. In less than two months, they successfully adapted the system to generate syntactic trees for wedding ring descriptions, speech pathologist reports, weather forecasts and flight details in French, soccer game descriptions in English, and lexicographic illustrative sentences for the lexical field of emotions in Mandarin, all with very satisfying results. To be clear, they only developed the lexical and grammatical resources for the mapping between manually written semantic representations and automatically generated surface syntactic structures, not the whole generation pipeline. But this "experiment" showed that the system is flexible enough to handle very different kinds of texts and can be easily and quickly adapted to a new domain or language. GenDR also proved to be a potent pedagogical tool for teaching formal linguistics and lexicography.

Our coverage of LFs is by far the most comprehensive that we know of, the closest being Lexfom (Fonseca et al., 2016a; Fonseca et al., 2016b; Fonseca, 2018) with ~600, Ayeye (Lareau et al., 2011; Dras et al., 2012; Lareau et al., 2012) with ~220, and MARQUIS with ~30. Our lexical coverage, however, is limited to the ~1500 most common words for English and French, and demo dictionaries for a pair of other languages. Obviously, we will work on increasing these figures. In terms of grammatical coverage, the core phenomena relevant to the semantics-syntax interface (lexicalization, complementation, modification) are implemented, but relative clauses and coordination were hastily implemented and require more serious work. Also, GenDR only properly handles active voice at the moment.

In order to address the problem of grammatical voice, more fine-grained control over the communicative/information structure is needed. In particular, one must be able to control what the theme/topic of the sentence is. A shift of thematic structure has a big impact on the syntactic structure (the theme is often expressed as the grammatical subject in many languages), which impacts lexical choice. We are starting to investigate this aspect.

We have nearly finished weaving VerbNet's (Kipper Schuler, 2006) syntactic frames into our English grammar and we plan to do the same with its French equivalent, Verbəlet (Pradet et al., 2014). This will drastically reduce the amount of work necessary to add new verbs to these languages, and since verbs control most of the syntax of a sentence, it will greatly increase the coverage of our system.

We also plan to map our current output structures to Universal Dependency structures (Nivre et al., 2016) as well as SimpleNLG (English, French and Spanish versions) (Gatt and Reiter, 2009; Vaudry and Lapalme, 2013; Ramos-Soto et al., 2017) for surface realization.

Finally, we want to merge GenDR with FORGe (Mille et al., 2017; Mille and Wanner, 2017), which has fared pretty well on a recent WebNLG challenge (Gardent et al., 2017). Both systems share a common ancestor, so they should be reasonably compatible.

8. Acknowledgements

This project was financed by the Fonds de recherche du Québec—Société et culture (FRQSC) (#2016-NP-191042).

9. Bibliographical References

Apresjan, J. D., Boguslavsky, I. M., Iomdin, L. L., and Tsinman, L. L. (2002). Lexical functions in actual NLP applications. In *Computational Linguistics for the New Millennium: Divergence or Synergy? Festschrift in Honour of Peter Hellwig on the occasion of his 60th Birthday*, pages 55–72. Peter Lang, Frankfurt.

Apresjan, J. (2000). *Systematic lexicography*. Oxford University Press, Oxford.

Basile, V. (2015). *From Logic to Language: Natural Language Generation from Logical Forms*. Ph.D. thesis, University of Groningen.

Bateman, J. A. (1996). *KPML Development Environment*. GMD/Institut für Integrierte Publikations- und Informationssysteme, Darmstadt.

Bohnet, B. and Wanner, L. (2010). Open source graph transducer interpreter and grammar development environment. In *Proceedings of LREC'10*, pages 211–218, Malta.

Bohnet, B., Langjahr, A., and Wanner, L. (2000). A development environment for an MTT-based sentence generator. In *Proceedings of INLG'00*, volume 14, pages 260–263, Mitzpe Ramon.

Bouayad-Agha, N., Casamayor, G., Wanner, L., Díez, F., and López Hernández, S. (2011). FootbOWL: Using a generic ontology of football competition for planning match summaries. In *Proceedings of ESWC'11*, Heraklion.

Bouayad-Agha, N., Casamayor, G., Mille, S., and Wanner, L. (2012). Perspective-oriented generation of football match summaries: Old tasks, new challenges. *ACM Transactions on Speech and Language Processing*, 9(2):art. 3, 1–31.

Miriam Butt et al., editors. (2012). *Proceedings of LFG'12*, Denpasar, Indonesia. CSLI.

CoGenTex, (1998). *RealPro: General English Grammar*. User manual.

Daoust, N. and Lapalme, G. (2015). JSREAL: A text realizer for web programming. In Núria Gala, et al., editors, *Language Production, Cognition, and the Lexicon*, pages 361–376. Springer, Zürich.

Dras, M., Lareau, F., Börschinger, B., Dale, R., Motazedi, Y., Rambow, O., Turpin, M., and Ulinski, M. (2012). Complex predicates in Arrernte. In Butt and King (Butt and King, 2012), pages 177–197.

Dubinskaite, I. (2017). Développement de ressources lituaniennes pour un générateur automatique de texte multilingue. Master's thesis, Université Grenoble Alpes.

Elhadad, M. and Robin, J. (1996). An overview of SURGE: A reusable comprehensive syntactic realization component. In *Proceedings of INLG'96*, number 96-03, pages 1–4, Brighton.

Elhadad, M. (1993). FUF: the universal unifier. User manual version 5.2. Technical report, Computer Science, Ben Gurion University of the Negev, Beer Sheva, Israel.

Fonseca, A., Sadat, F., and Lareau, F. (2016a). Lexfom: a lexical functions ontology model. In *Proceedings of CogALex'16 at COLING*, Osaka.

Fonseca, A., Sadat, F., and Lareau, F. (2016b). A lexical ontology to represent lexical functions. In *Proceedings of LangOnto'16 at LREC*, Portorož.

Fonseca, A. (2018). *Représentation des collocations dans un réseau lexical à l'aide des fonctions lexicales et des formalismes du web sémantique*. Ph.D. thesis, Université du Québec à Montréal.

Gardent, C., Shimorina, A., Narayan, S., and Perez-Beltrachini, L. (2017). The WebNLG challenge: Generating text from RDF data. In *Proceedings of INLG'17*, pages 124–133, Santiago de Compostela.

Gatt, A. and Reiter, E. (2009). SimpleNLG: A realisation engine for practical applications. In *Proceedings of ENLG'09*, pages 90–93, Athens.

Jousse, A.-L. (2010). *Modèle de structuration des relations lexicales basé sur le formalisme des fonctions lexicales*. Ph.D. thesis, Université de Montréal/Université Paris 7.

Kahane, S. and Polguère, A. (2001). Formal foundation of lexical functions. In *Proceedings of ACL 2001*, Toulouse.

Kahane, S. (2003). The Meaning-Text Theory. In Vilmos Ágel, et al., editors, *Dependenz und Valenz: Ein internationales Handbuch der zeitgenössischen Forshung/ Dependency and Valency: An International Handbook of Contemporary Research*, volume 1, pages 546–570. Walter de Gruyter, Berlin/New York.

Kipper Schuler, K. (2006). *VerbNet: A Broad-Coverage, Comprehensive Verb Lexicon*. Ph.D. thesis, University of Pennsylvania, Philadelphia.

Lambrey, F. and Lareau, F. (2015). Le traitement des collocations en génération de texte multilingue. In *Actes de TALN'15*, pages 579–585, Caen.

Lambrey, F. (2016). Implémentation des collocations pour la réalisation de texte multilingue. Master's thesis, Université de Montréal.

Lareau, F. and Wanner, L. (2007). Towards a generic multilingual dependency grammar for text generation. In Tracy Holloway King et al., editors, *Proceedings of the GEAF07 Workshop*, pages 203–223, Stanford. CSLI.

Lareau, F., Dras, M., Börschinger, B., and Dale, R. (2011). Collocations in multilingual natural language generation: Lexical functions meet lexical functional grammar. In Diego Mollá et al., editors, *Proceedings of ALTA'11*, pages 95–104, Canberra.

Lareau, F., Dras, M., Börschinger, B., and Turpin, M. (2012). Implementing lexical functions in XLE. In Butt and King (Butt and King, 2012), pages 362–382.

Lavoie, B. and Rambow, O. (1997). A fast and portable realizer for text generation systems. In *Proceedings of ANLP 1997*, pages 265–268, Washington.

Mel'čuk, I. A. (1973). Towards a linguistic "Meaning-Text" model. In Ferenc Kiefer, editor, *Trends in Soviet Theoretical Linguistics*, pages 33–57. Reidel, Dordrecht.

Mel'čuk, I. A. (1988). *Dependency syntax: theory and practice*. SUNY series in linguistics. State University of New York Press, Albany.

Mel'čuk, I. (1995). The future of the lexicon in linguistic description and the explanatory combinatorial dictionary. In I.-H. Lee, editor, *Linguistics in the morning calm*, volume 3. Hanshin, Seoul.

Mel'čuk, I. A. (1996). Lexical functions: A tool for the description of lexical relations in a lexicon. In Wanner (Wanner, 1996), pages 37–102.

Mel'čuk, I. (1998). Collocations and lexical functions. In A. P. Cowie, editor, *Phraseology. Theory, analysis, and applications*, pages 23–53. Clarendon, Oxford.

Mel'čuk, I. A. (2001). *Communicative organization in natural language: the semantic-communicative structure of sentences*. Studies in language companion series. John Benjamins, Amsterdam/Philadelphia.

Mel'čuk, I. A., (2007). *Phraseology: An international handbook of contemporary research*, volume 1, chapter 11: Lexical functions. Walter de Gruyter, Berlin/New York.

Mel'čuk, I. A. (2012). *Semantics: From Meaning to Text*, volume 1. John Benjamins, Amsterdam/Philadelphia.

Mel'čuk, I. A. (2013). *Semantics: From Meaning to Text*, volume 2. John Benjamins, Amsterdam/Philadelphia.

Mel'čuk, I. A. (2014). *Semantics: From Meaning to Text*, volume 3. John Benjamins, Amsterdam/Philadelphia.

Milićević, J. (2006). A short guide to the Meaning-Text theory. *Journal of Koralex*, 8:187–233.

Mille, S. and Wanner, L. (2017). A demo of FORGe: the Pompeu Fabra open rule-based generator. In *Proceedings of INLG'17*, pages 245–246, Santiago de Compostela.

Mille, S., Carlini, R., Burga, A., and Wanner, L. (2017). FORGe at SemEval-2017 task 9: Deep sentence generation based on a sequence of graph transducers. In *Proceedings of SemEval'17*, pages 920–923, Vancouver.

Mille, S. (2014). *Deep stochastic sentence generation: Resources and strategies*. Ph.D. thesis, Universitat Pompeu Fabra, Barcelona.

Molins, P. and Lapalme, G. (2015). JSrealB: A bilingual text realizer for web programming. In *Proceedings of ENGL'15*, pages 109–111, Brighton.

Nivre, J., de Marneffe, M.-C., Ginter, F., Goldberg, Y., Hajič, J., Manning, C. D., McDonald, R., Petrov, S., Pyysalo, S., Silveira, N., Tsarfaty, R., and Zeman, D. (2016). Universal Dependencies v1: A multilingual treebank collection. In *Proceedings of LREC'16*, pages 1659–1666, Portorož.

Polguère, A. (2000). A "natural" lexicalization model for language generation. In *Proceedings of SNLP 2000*, pages 37–50, Chiangmai.

Polguère, A. (2007). Lexical function standardness. In Leo Wanner, editor, *Selected lexical and grammatical issues in the Meaning-Text theory: in honour of Igor Mel'čuk*, volume 84 of *Language Companion Series*. John Benjamins, Amsterdam/Philadelphia.

Pradet, Q., Danlos, L., and de Chalendar, G. (2014). Adapting VerbNet to French using existing resources. In *Proceedings of LREC'14*, Reykjavik.

Ramos-Soto, A., Janeiro-Gallardo, J., and Bugarín, A. (2017). Adapting SimpleNLG to Spanish. In *Proceedings of INLG'17*, pages 144–148, Santiago de Compostela.

Vaudry, P.-L. and Lapalme, G. (2013). Adapting SimpleNLG for bilingual English-French realisation. In *Proceedings of ENLG'13*, pages 183–187, Sofia.

Wanner, L. and Bateman, J. A. (1990). A Collocational Based Approach to Salience-Sensitive Lexical Selection. In *Proceedings of INLG'90*, Dawson, PA.

Wanner, L. and Lareau, F. (2009). Applying the Meaning-Text Theory model to text synthesis with low- and middle-density languages in mind. In Sergei Nirenburg, editor, *Language Engineering for Lesser-Studied Languages*, volume 21 of *NATO Science for Peace and Security Series - D: Information and Communication Security*. IOS Press, Amsterdam.

Wanner, L., Brügmann, S., Diallo, B., Giereth, M., Kompatsiaris, Y., Pianta, E., Rao, G., Schoester, P., and Zervaki, V. (2009). PATExpert: Semantic processing of patent documentation.

Wanner, L., Bohnet, B., Bouayad-Agha, N., Lareau, F., and Nicklaß, D. (2010). MARQUIS: Generation of user-tailored multilingual air quality bulletins. *Applied Artificial Intelligence*, 24(10):914–952.

Wanner, L., Bott, S., Bouayad-Agha, N., Casamayor, G., Ferraro, G., Graën, J., Joan Casademont, A., Lareau, F., Mille, S., and Vidal, V. (2011). Paraphrasing and multilingual summarization of patent claims. In Leo Wanner, et al., editors, *PATExpert: A Next Generation Patent Processing Service.* Technical documentation.

Wanner, L. (1992). Lexical Choice and the Organization of Lexical Resources in Text Generation. In *Proceedings of the European Conference on Artificial Intelligence*, Vienna, Austria.

Leo Wanner, editor. (1996). *Lexical functions in lexicography and natural language processing*. John Benjamins, Amsterdam/Philadelphia.

Weißgraeber, R. and Madsack, A. (2017). A working, non-trivial, topically indifferent NLG system for 17 languages. In *Proceedings of INLG'17*, pages 156–157, Santiago de Compostela.

White, M., (2008). *OpenCCG Realizer Manual*.

Žolkovskij, A. and Mel'čuk, I. (1967). O semantičeskom sinteze. *Problemy kibernetiki*, 19:177–238.