Verbalizing AMR Structures

Guy Lapalme
RALI-DIRO, Université de Montréal
lapalme@iro.umontreal.ca

August 25, 2019

Abstract

We describe a system for generating a literal reading of Abstract Meaning Representation (AMR) structures. The system uses a symbolic approach to transform the original rooted graph into a tree of constituents that is transformed into an English sentence by an existing realizer. The system is quite fast and has been applied on a wide variety of AMR structures ($\approx$ 40K of them). The generated sentences are usually longer than the reference ones because they explicit all relations contained in the AMR. 98% of the sentences have been judged good or satisfactory for the stated goal: helping human annotators to check the AMRs they have written. A comparative evaluation of statistical generators showed equivalent or better quality output while being much faster to execute. Based on this experiment, we give suggestions for further AMR annotation.

1 Introduction

The goal of this work is to create a system to *verbalize literally* Abstract Meaning Representation (AMR) [1] graph structures. An AMR represents the semantics of an English sentence by mapping different grammatical realizations into a single graph in order to focus on the meaning of the sentence. Important syntactic phenomena such as articles, number, tense and voice are not represented in the graph; variables being quantified existentially, universal quantification cannot be represented in all cases. Given the current state of the art in NLP, parsing an English sentence in order to get its AMR graph is not yet reliable, so currently AMR graphs must be created by human annotators who fortunately can rely on useful tools such as a computer-aided editor [11][1] that validates the syntactic form of the graph and provide other annotating guides. These graphs are then revised in order to obtain consensus AMRs. One stated goal of the AMR project is to develop a large *sembank* for shared tasks in natural language understanding and generation.

[1] https://amr.isi.edu/editor.html

Over the years, research groups have created AMRs for different types of texts: novels (e.g. Le Petit Prince), scientific texts in biology, news articles, English translations of Chinese texts, etc. The latest release by LDC (2.0) [13] provides ≈ 40K sentences with their AMR structures. This corpus has been used for developing automatic AMR parsers (sentence to AMR) (some of which will be presented in Section 6) and generators (AMR to sentence) by means of machine learning or machine translation techniques.

Their results were evaluated in the context of two *SemEval* tasks. *Task 8* of *SemEval-2016* [15] was devoted to the parsing of English sentences to get the corresponding AMR. *Task 9* of *SemEval-2017* [16] comprised two tasks: parsing biomedical sentences to get AMRs and generate English sentences from valid AMRs. Given the many mappings between a sentence and an AMR structure, automatically evaluating the output of such systems is quite difficult. For an AMR parser, approximate graph matching between the original graph and the one created by the parser is used [5]. For an AMR generator, BLEU scores [19] are used to compare the output sentence with the reference one. Section 6 will describe some generators that participated in the generation task of *SemEval-2017*.

All participants in the generation task used machine learning and statistical techniques. But as the input AMR is a formal language that can be easily parsed using a context-free grammar, we decided to try a symbolic approach in Prolog for generating the English sentence. Our system is called Γω-Φ[2]. In Section 5, we will see that Γω-Φ is quite competitive with previous generators both in terms of quality of the generated sentence and execution speed. But it must be emphasized that the goal of our AMR generator is different from the previous attempts: we do not try to reproduce the reference sentence verbatim, but instead we generate a literal reading of the graph that we hope would be helpful for annotators when they create their graph by providing them a quick feedback on the annotation they have created.

The wide gap between the reference and the corresponding graph is illustrated in the example of Table 1 in which the reference sentence is quite cryptic for people whose mother tongue is not English (like me) and who are not aware that *DH* refers to a cherished person and that *A&E* is a hospital department. Such colloquial expressions whose semantics is *expanded* in the AMR are one of the reasons why BLEU scores of previous generators are so low (less than 20%) and that these scores are not very useful for improving the generation systems. Mapping between the input AMR and the corresponding sentence being many to many and the relatively low number of AMR annotated sentences, it proved quite difficult to develop efficient automatic learning algorithms.

We do not consider the generated verbalization of Γω-Φ as perfect, but more explicit as it better describes all the elements of the original graph. In fact, during the development of our system, we managed to find a few typos and errors in some examples of the AMR Guidelines [2] which have been since corrected. This would probably not have happened using machine learning techniques which do not usually challenge input-output pairs.

[2]In homage to Donald Knuth, we use three Greek letters for the name of the system that can be read aloud as GOPHI (Generation Of Parenthesized Human Input) pronounced as GOFAI (Generating Output From AMR Input) an acronym that also has an older reading originally given by John Haugeland.

```
(n / need-01
     :ARG0 (p / person
          :ARG0-of (h / have-rel-role-91
               :ARG1 (i / i)
               :ARG2 (h2 / husband))
          :mod (d / dear))
     :ARG1 (t / treat-03
          :ARG1 p
          :location (d2 / department
               :topic (a / and
                    :op1 (a2 / accident)
                    :op2 (e / emergency))))
     :time (a3 / after
          :op1 (a4 / attack-01
               :ARG1 p)))
```

Reference	DH needed treatment at A&E after the attack
Γω-Φ	My husband dear needs the treatment in the department about the accident and the emergency after the attack.
JAMR	dear i have-rel-role husband person need to accident and emergency department treatment after attacks
ISI-MT	after the attack dear husband need treatment in an accident and emergency department
Generate	Dear person that have-rel-role i husband need person treat in department about accident , and emergency after person attack .
Baseline	person i have husband dear need treat department accident and emergency attack after.

Table 1: An AMR corresponding to the *Reference* sentence and their realization produced by different systems: Γω-Φ is the system described in this paper; JAMR and ISI-MT are existing generators based on machine learning techniques described in Section 6; *Generate* is the output `generate` button of the *AMR editor* [11] but as the author acknowledges, it is not very reliable because it has been developed in three days; *Baseline* is a generator written in ten lines of Prolog code described in Section 9.1.

Another motivation for our work was developing a *test bench* for jsRealB [18] (a bilingual French-English[3] realizer written in Javascript) that has been developed in recent years in our lab[4]. As this realizer takes care of the details of the English language generated from an abstract constituency structure, we hoped it would be easier to go from an AMR to a jsRealB structure than to a well-formed English sentence.

It would be interesting to develop a machine-learning approach for transforming between these two formalisms, but we leave this as an *exercise to the reader*. We will further discuss this in section 6.

In the following, we first briefly recall what an AMR is and the constituency structure that we target. We then describe the intermediary representations that we use for transforming an AMR to an English sentence. The implementation of the system and the tests are then presented. The evaluation results (both automatic and manual) are given and compared with previous work. We end with a discussion of the pros and cons of our approach. We also make suggestions for streamlining the AMR and for limiting the number of *primitives*. Three appendices give supplementary information: 9.1 compares the output of different AMR generators on a few selected AMRs; 9.2 gives implementation details for the core algorithm; 9.3 describes the data on which our system has been applied.

2 AMR concepts

An AMR is a singly rooted, directed acyclic graph with labels on edges (relations) and on nodes (concepts). AMR structures can use PropBank semantic roles [3], within-sentence coreference and can take into account some types of polarity and modality. The AMR Specifications [2] are the authoritative source for details about the formalism and its use for annotation of sentences.

The Lisp-inspired syntax of an AMR is quite simple: an AMR is either a variable, a slash and the name of a concept followed by a list, possibly empty, of role names followed by an AMR all within parentheses; an AMR can also be a variable reference or a string constant. A *concept* that stands for an event, a property or a state often corresponds to a PropBank frame or is an English noun, adjective or pronoun. A *role* name, an identifier preceded by a colon, indicates a relation between concepts; more than 100 role names are predefined. A *variable* is a letter followed by 0 or more digits associated with a concept. The variable can be used to cross-reference a concept in an AMR. A *constant* is either a number, a quoted string or a plus or minus sign.

In the following, we will use the example of Table 2[5] to illustrate the steps of our system. It is a simple AMR structure corresponding to the sentence `The boy desires the girl who does not like him.` As AMR structures abstract many common syntactic concepts such as number, tense or modality, it could also represent `The desire of boys for girls that do not like them` that we give as reference sentence.

[3]In this work, we only use the English realization part of jsRealB

[4]jsRealB is freely available at `https://github.com/rali-udem/jsRealB`

[5]This AMR is adapted from an example of the AMR Guidelines [2]

```
(d / desire-01
    :ARG0 (b/boy)
    :ARG1 (g/girl
           :ARG0-of (l/like-01
                       :polarity -
                       :ARG1 b)))
```

Reference	The desire of boys for girls that do not like them
Γω-Φ	The boy desires the girl who doesn't like him.
JAMR	boy - like desire to girls
ISI-MT	the boy has a desire to be the girl who do not like
Generate	Boy desire girl that not like .
Baseline	boy desire girl like -.

Table 2: A simple AMR and their realization with the systems described in Table 1. The transformation steps followed by Γω-Φ to produce the English sentence are given Table 3.

The root of an AMR usually represent the focus of the sentence. This AMR contains two PROPBANK predicates (`desire-01` and `like-01`), each with two arguments (`:ARG0` and `:ARG1`) that stand respectively for the subject and the object of the verbs. The `boy` is the subject of `desire-01` and the object of `like-01`; in the former, it is referred to by the use of the variable `b` introduced above. Inverse roles, denoted with the suffix `-of`, can also be used to link two predicates to a single concept by keeping the focus: in our example, `girl` is the object of `desire-01` and object of `like-01`. Negation is indicated with the `:polarity` role whose value is `-` (minus). The table also shows the output produced by other generators for comparison.

The first row of Table 3 shows the translation into First-Order Logic (FOL) by means of a conjunction of existentially quantified predicates, called logical triples in the AMR community, as given by the algorithm described by Bos [4]. This form is useful for comparing the output of AMR parsers by means of the Smatch [5] algorithm which calculates the degree of overlap between two semantic feature structures. We first attempted to map this FOL to English, but we found it difficult to keep the right focus in the generated sentence, although in hindsight it might be worth trying this approach again. So we decided to start directly from the AMR and go through a series of intermediary structures that we label using a nomenclature loosely borrowed from Meaning-Text theory [17].

Semantic Representation (row 2 of Table 3) is a parsed representation of the original AMR (a Prolog term) in which variables that are referenced elsewhere in the AMR are prefixed by \. They correspond to the *projected* variables that Bos introduced for making sure that they are appropriately scoped in the FOL formula. This semantic representation is then modified by removing the inverse roles and adding a reference (projected if necessary) in the context of the modified predicate. Row 3 of Table 3 shows the result

applied to our example, in which the `:ARG0-of` role of the `like-01` predicate has been transformed.

Deep Syntactic Representation (row 4 of Table 3) is the result of transforming the Semantic Representation into a tree of constituents which are lambda expressions with possibly `null` arguments which will be *applied* to produce the following representation. The production of the Deep Syntactic Representation is the *hard* part of our work and will be detailed in the following section.

Surface Syntactic Representation (row 5 of Table 3) is the input format that jsRealB uses to produce a well-formed English sentence.

1	First-Order Logic	∃b(boy(b) ∧ ∃d(desire-01(s) ∧ :ARG0(d,b) ∧ ∃g(girl(g) ∧ ∃l(like-01(l) ∧ :polarity(l, -) ∧ :ARG1(l,b) ∧ :ARG0(l,g)) ∧ :ARG1(d,g))))
2	Semantic Representation	['desire-01',d, [':ARG0',[boy,\b]], [':ARG1',[girl,g, [':ARG0-of',['like-01',l, [':polarity',-], [':ARG1',b]]]]]]
3	Semantic Representation without inverse role	['desire-01',d, [':ARG0',[boy,\b]], [':ARG1',[girl,\g, [':*:ARG0',['like-01',l, [':ARG0',g], [':polarity',-], [':ARG1',b]]]]]]
4	Deep Syntactic Representation	s(np($(null)/d("the"),null,n("boy")), vp(v("desire"), np($(null)/d("the"),null, n("girl"), ls(pro("who"), s(null, vp(v("like"), pro("me")*pe("3")*g("m")))*typ({"neg":true})))))
5	Surface Syntactic Representation	S(NP(D("the"), N("boy")), VP(V("desire"), NP(D("the"), N("girl"), Pro("who"), S(VP(V("like"), Pro("me").pe("3").g("m"))).typ({"neg":true}))))
6	English	The boy desires the girl who doesn't like him.

Table 3: Example of representations used in the transformation of the AMR structure shown in Table 2 to the sentence shown in the last row. Rows 2 to 4 are Prolog terms. Row 5 is a Javascript expression evaluated by jsRealB to realize the English sentence shown in row 6.

```
like-01  :  λa.λb.s(a,vp(v("like"),b))
desire-01  :  λa.λb.s(a,vp(v("desire"),b))
boy  :  λa.λb.np($a/d("the"),b,n("boy")))
girl  :  λa.λb.np($a/d("the"),b,n("girl")))
```

```
like-01 (boy null null) (girl d("a") null)
```

```
s(np($(null)/d("the"),
     null,
     n("boy")),
  vp(v("like"),
     np(/($(d("a")),
          d("the")),
        null,
        n("girl"))))
```

```
S(NP(D("the"),
     N("boy")),
  VP(V("like"),
     NP(D("a"),
        N("girl"))))
```

Table 4: The top part shows an excerpt of the dictionary for the concepts used in the example of Table 3, the middle part shows a simple λ-expression expression and the bottom part shows the pretty-printed result of the evaluation and the corresponding constituency tree once null values are removed taking the default into account.

3 From Semantic Representation to Deep Syntactic Representation

As we said before, getting the Semantic Representation from an AMR is a relatively simple parsing job, and transforming a Deep Syntactic Representation to the Surface Syntactic Representation is merely a change of notation after removing `null` values. So the core of the transformation work is going from the Semantic Representation to a Deep Syntactic Representation (i.e. from the third to the fourth row of Table 3) which is detailed in this section.

We transform the Semantic Representation compositionally by means of lambda expression applications. Each concept in the dictionary is coded as a lambda expression which, when it is applied to another concept, creates a new Deep Syntactic Representation corresponding to their composition.

The top of Table 4 presents the four concepts of the example of Table 3. `like-01` is a lambda expression that corresponds to a constituency tree with `s` as root; the first child is the subject given by a, the second child is a `vp` tree with its first child being the verb `"like"` and b its second child, the object of the verb. When a lambda term is applied, a missing subject or object is indicated with the `null` value. `desire-01` is coded similarly for the first two arguments.

`boy` is a lambda expression that produces a constituency tree having `np` as root with

three children: the determiner, an adjective and the noun itself. We introduce a notation for a default value: when a variable is preceded by a dollar sign, it means that if a value is given then it will be used, but if `null` is given then the value after the slash will be used instead. For example, `$(null)/d("the")` means that if a determiner is specified, then it will be used, but if not, then the definite determiner `the` will be used instead.

In AMR, argument values are given by the values of the roles which are recursively evaluated until they are given simple values found in the dictionary. AMR concepts being the ones of PropBank, we created 7,792 verb entries of our dictionary automatically by parsing the XML structures of PropBank frames to determine the arguments and used the annotated examples for determining the proper preposition to use. We also parsed other PropBank XML entries for determining related 2,477 nouns and 1,264 adjectives. We also added about 33,300 nouns, adjectives, conjunctions and adverbs from an internal dictionary of our lab that did not appear in the entries of PropBank. Some pronouns and other parts of speech were added manually.

If AMR expressions were limited to PropBank concepts with frame arguments (i.e. `:ARG`$_i$) as roles then we could limit ourselves to this single application process (see Section 9.2 for implementation details). But there are many other AMR peculiarities to take into account. We leave it to the reader to decide if these singularities are fundamental or just an artifact to ease the annotation process and even then, we conjecture that a single and more uniform process would be preferable. We now describe how some of these special cases are handled.

3.1 Polarity

As described by Bos, one difficulty in processing AMR for getting the First-Order Logic form is the fact that negation is indicated by a special role (`:polarity`) associated with `-`; this does not fit well with the lambda application process that we explained above. Fortunately, jsRealB also uses this type of flag to indicate that the current sentence should be negated. This process was borrowed from a similar mechanism in SimpleNLG [9] in which it is possible to mark a sentence as negated; the realization process then modifies the dependency structure to produce the negation of the original sentence. In jsRealB, negation is indicated by specifying the `type` of sentence by adding `.typ("neg":true)` after the `S` constructor; this type of modification is called an `option` in jsRealB. An example of this is shown in the row 5 of Table 3. For a technical reason, we use `*` instead of `.` in the Deep Syntactic Representation[6]. So when the application process encounters a `:polarity` role, it must keep track of the fact that an option should be added to this effect in the Deep Syntactic Representation. Other roles also use this option *trick*. In some AMRs, the `:polarity` role is also added to concepts that correspond to nouns or adjectives. Unfortunately this is not implemented by jsRealB, so we had to check for special cases and even there, this does not yet take into account all cases.

[6]In the Prolog implementation that we use (SWI-Prolog 7.6), `.` has been given a special meaning that prevents it from being used as an *ordinary* operator.

3.2 Inverse Roles

Inverse roles (indicated by `-of` at the end of a role) do not fit easily in the application process. They are introduced in AMR mainly for keeping the focus on a single concept that is used in different frames such as the `girl` in our example of Table 3 that is an object of one frame and the subject of another. The inverse roles can be quite difficult to process in the general case, but for our verbalization context we decided to systematically introduce a relative clause. After trying some alternatives, we resorted to a *hack*: we transform the original AMR containing inverse roles into one that only uses active roles but flagged to indicate to the realizer that a relative clause should be introduced. Table 3 shows an example of this transformation between rows 3 and 4 where a `:ARG0-of` has been transformed into a `:*:ARG0` role with an explicit `:ARG0` role added in the inner concept.

3.3 Non-Core Roles

As described in the AMR Guidelines [2], there are about 20 other non-core roles such as: `:domain`, `:mod` etc., each of which must be dealt specifically. But the basic principle remains the same, each AMR associated with a role is evaluated recursively and its Deep Syntactic Representation is added to the deep structure of the current concept most often at the end of the values of the core roles. For example, for a `:destination`, we add a prepositional phrase starting by preposition `to`. In other cases, e.g. `:polite +`, then the word `Please` is added at the start of the deep structure. In an AMR the `:wiki` role is used to disambiguate named entities, we use its value as the target of an HTML link that the system generates.

3.4 Special concepts

Some *special* concepts appear quite frequently, but we did not manage to deal with them using the above framework.

`amr-unknown` is most often used for annotating an interrogative form; although intuitively this is convenient for the annotator and easy for a human to understand, automatically finding the appropriate form of interrogation is quite difficult because it depends on the enclosing role: e.g. if it appears as the value of an `:ARG0` or `:ARG1` then the question should start with `who`, if it appears as an argument of `:polarity`, then it should be a yes-or-no question, etc. Fortunately, jsRealB has options to transform affirmative into interrogative forms of different types, so in most cases, it is only a matter of adding the right option depending on the context. But there are still cases that are not dealt with correctly in the system. A closer examination of the use of `amr-unknown` revealed that this *vague* concept seemed to have been interpreted in diverse ways by different annotators.

`have-degree-91` indicates the comparison between two entities and often induces dependencies between structures of different roles. For example, a comparative is annotated

using different roles for the domain, the attribute, the degree, the object of the comparison and a possible reference to a superset. Finding appropriate verbalizations for all these cases proved quite tricky but essential as this concept is often used.

`have-polarity-91` is a reification (see below) of polarity with a specific use of `:ARG2`.

`have-quant-91` which serves to mark a relation between an owner and specific types of quantifiable goods for certain goals.

`have-rel-role-91` indicates the relation between two entities and the type of relation (e.g. `father`, `sister`,...) all using different roles. This also needed a specific processing.

`date-entity` has more than ten specific roles which have to be taken into account.

`ordinal-entity` to deal with some specific ways of verbalizing ordinal numbers: e.g. `last` for -1, `second to last` for -2, etc.

`*-quantity` there are about a dozen types of quantities, for which the roles `:quant` and `:unit` must be combined appropriately (using the value of the quantity as determiner for the unit when this value is a numeric value).

It would have been interesting to find a more systematic handling of all these cases. But we also conjecture that these special cases are a symptom of some design deficiencies in the original AMR.

3.5 Pronoun generation

Variables in AMR enable the coreference between entities and create a graph between elements that could otherwise be considered as a tree. Although quite intuitive for the annotator, it proved to be quite tricky to generate the appropriate pronoun given the fact that our algorithm does not have easy access to the global context. Although we can deal properly with usual cases, there are still many pending problems. Some of them have to do with the fact that, by design, AMR abstract important grammatical information such as gender and number; for example, *Steffi Graf* is referred to using `he` or *Yankees* is used as singular (see Figure 1). English pronouns also are different when used as nominative (`:ARG0`:`I`) or as accusative (`:ARG1`:`me`). But this *trick* does not always work because it depends on the argument structure of the concept and the fact that the subject or object is animate or not, an information that currently we do not have because it would imply parsing the comments in the PropBank file.

3.6 Passive and other peculiarities

AMR does not indicate if the sentence should be realized as a passive. Instead annotators seem to rely on a convention that a verb with a subject, usually indicated by `:ARG0`, is used with an `:ARG1` instead. So this must be checked before any other processing because passive

is used extensively in English, especially in the news genre often encountered in the annotated corpora. Other special cases that must be checked are imperative and *yes or no question* that use the `amr-choice` special concept as `:ARG1`. Moreover the AMR editor provide some *shortcuts* that are expanded in the resulting AMR. For example, `cause-01` is expanded as an inverse role, that we have to check and *un-expand* to produce a more readable sentence. Many named entities[7] followed by a proper name are explicitly given in the AMR, but only the proper noun is written out in the text (see *Yankees* in Figure 1). Γω-Φ checks for these in order to simplify the output.

3.7 Verbalization and Reification

AMR is oblivious to verbalization, see our example of Table 2 for which `The boy desires` and `The desire of the boys` are annotated using the single concept `desire-01`. Using tables provided on the ISI website[8] for helping annotators, we added some verbalization information to the dictionary which is used to nominalize a verb when it does not have a subject.

Roles can sometimes be used as a concept, a process called *reification* e.g. `:location` is reified as `be-located-at-91`. This is used to indicate that the focus of the sentence is the locating process itself instead of the object being located or to the location itself. Mapping tables between roles and their reifications are given on the ISI website and we adapted them for our context. Our current implementation does not yet deal satisfactorily with all cases of reification.

3.8 Unknown Role or Concept

When an unknown concept is encountered, we use the fact that AMR is English centric and that English morphology is relatively simple, at least compared to French and German. So we merely use the name of the concept, after removing dashes or the frame number, as the word to add to the sentence. When an unknown role is encountered, the corresponding AMR value is added at the end of the current constituent, ignoring the name of the role.

3.9 Conclusion

Although the basic principle for creating an AMR verbalizator is a simple β-reduction of lambda forms, there are (too ?) many special cases that do not fit well within the declarative approach to the transformation from Deep Syntactic Representation to the Surface Syntactic Representation.

An important limitation of our current implementation is the fact that we do not take lexical ambiguity into account, so if a concept corresponds in the dictionary either to a verb or to a noun, we consider only the verb entry. Some AMR examples (e.g. given in [**?**]) include

[7] A list is given at `https://www.isi.edu/~ulf/amr/lib/ne-types.html`

[8] Resource list section at `https://amr.isi.edu/download.html`

the part of speech in the name of the concept e.g. `check.v.01` or `check.n.01` instead of `check-01`, but our corpora do not provide this information.

4 Implementation

Given the mostly declarative approach we have used and the fact that Prolog unification is equivalent to β-reduction, we implemented the system (AMR to Surface Syntactic Representation) in Prolog: around 3,000 lines of SWI-Prolog 8.1, most of which for dealing with *special cases* described in sections 3.1 to 3.4, 57,000 lines for the automatically created dictionaries. Python was used for the transformation of the PropBank XML files to generate the dictionary and for scripts for computing statistics and creating an Excel file to ease the evaluation process. The final realization of the English sentence is produced by jsRealB as a Node.js module taking as input the Surface Syntactic Representation.

To develop Γω-Φ, we extracted the 268 examples from the AMR Guidelines [2] and the 826 examples from the AMR Dictionary [12] used to help the annotators. These AMRs are usually short sentences that combine a few concepts for didactic purposes and are designed to cover most of the annotation cases; they are thus ideal for developing a system, even though most sentences in the *real* corpora are much longer and combine diverse roles in a single AMR.

We recall that our goal is not to reproduce verbatim the original sentence, but to give a literal reading of the AMR in which all verbs are generated at the present tense and in the active voice; all nouns are singular with a definite determiner. Given the fact that the original sentences usually have a great variation in number, tense (some informal sentences are even encountered, see Table 1), it is expected that the BLEU scores between the reference and our *standardized* output will not be high. Moreover, Γω-Φ also generates an HTML link when a `:wiki` role is used in an AMR (see the right part of Figure 1).

Γω-Φ produced a sentence for all AMRs of our development, test and training examples: corpora available at the ISI website [9] and the ones from the AMR 2.0 Distribution [13] (see Table 14 for more details). Γω-Φ run on a MacBook laptop (1.2 GHz without GPU!) is quite fast: a few milliseconds of CPU and about 7 ms of real time per sentence. Of course, longer sentences take more time to verbalize, but it is still quite fast compared to other statistical systems which take seconds for computing an English sentence even after a longer initial loading phase. These timings are for the Prolog engine to parse the AMR and producing the Surface Syntactic Representation, jsRealB is also almost instantaneous. The generated sentences are systematically longer than the reference sentence, between 15% up to 70% longer, for a mean of 38%.

We also developed a web server[10], also written in Prolog, that displays a web page (see Figure 1) in which a user can edit an AMR. The AMR is then transformed and its English realization is generated by an instance of jsRealB integrated in the response web page itself. This setup allows the web links generated by jsRealB to be clicked directly from

[9] `https://amr.isi.edu/download.html`

[10] `http://rali.iro.umontreal.ca/amr/current/build/amrVerbalizer.cgi`

this web page (see the words *Yankees* in the right part of the figure). This example (numbered isi_0001.12) taken from the AMR Dictionary gives as a reference sentence: *The boy doesn't think the Yankees will win.* in which the negation is not given the same scope in the reference and in the AMR. This is a case in which an AMR verbalizer could have helped catch this type of discrepancy because the annotator could have noticed that the generated sentence from the AMR has a slightly different meaning than the original sentence, so it should be double-checked.

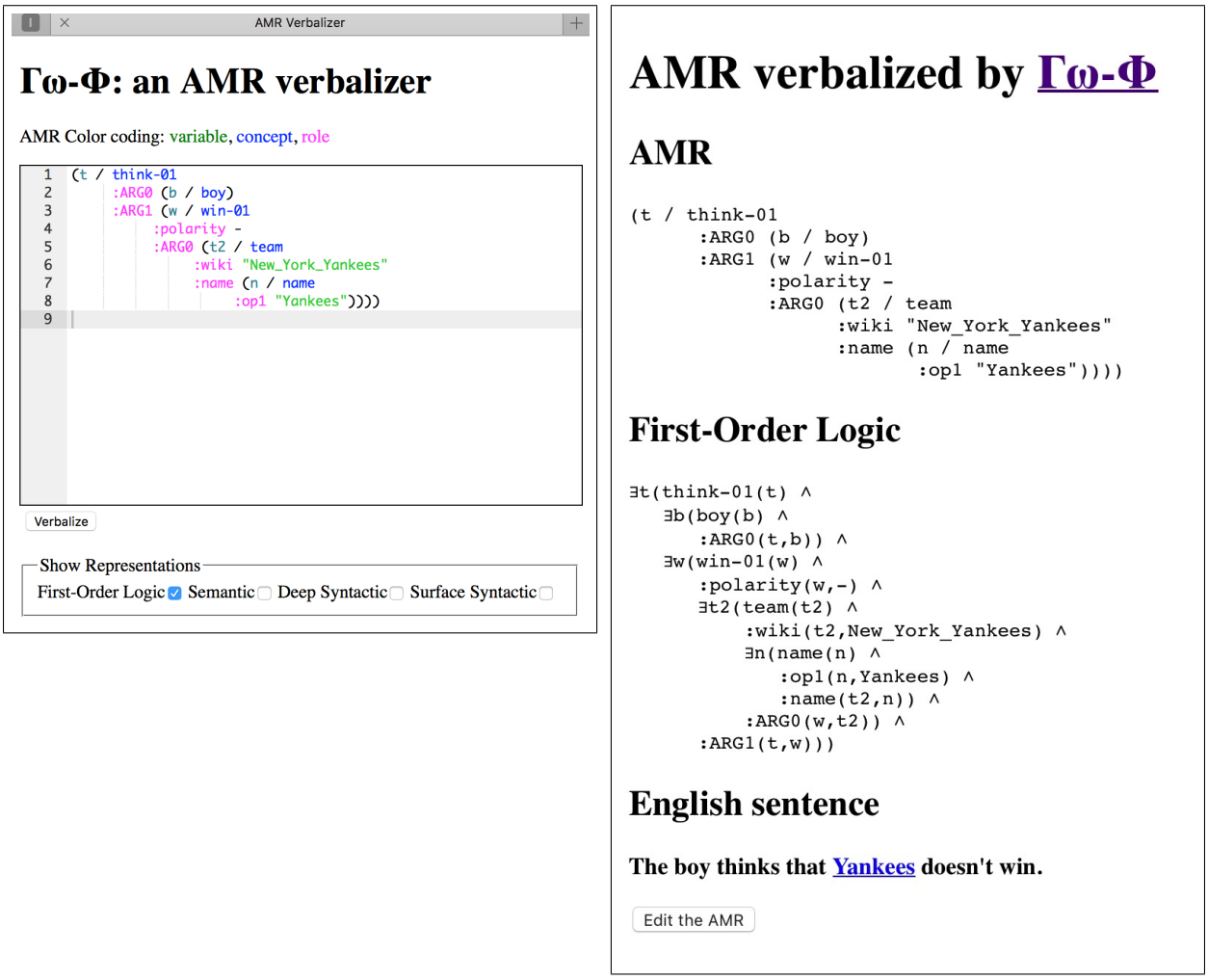

Figure 1: Web interface to the verbalizer. The input page on the left contains an editor with a special *mode* for editing AMRs; intermediate representations can also be requested with the checkboxes at the bottom (only the FOL representation is chosen here). The right part shows the corresponding FOL formula (in indented form) and the realized sentence created by the embedded jsRealB module in the webpage.

5 Evaluation

AMR generation outputs up to now have been evaluated either by means of BLEU scores [7, 20] and by means of pairwise comparisons [16]. These works are described in more details in the next section. But unfortunately these metrics are not very useful for helping the development or the improvement of a symbolic system.

5.1 Development Evaluation

So in order to keep track of the *progress* of our system, we devised a crude metric using the scale shown in Table 5. During the course of our development, we manually evaluated a small subset to measure the adequacy of the generated sentence for the intended purpose, that is, helping an annotator to check if the proper concepts and roles have been chosen in the annotation. This evaluation scale helped us focus on the most frequent drawbacks of the system.

We also evaluated our development examples and the first 25 AMRs of each test set of the AMR 2.0 Distribution [13] (see Section 9.3 for more details). Only 4% of examples were useless (score of 2) and none were missing some important information (score of 1). As the generation of web links is systematic, HTML tags were removed from the generated text before the evaluation. Comparison with the reference sentence is not taken into account in this evaluation as the generated verbs are always conjugated at the present tense, the nouns are always singular and the determiners always definite.

4	Perfect translation of all concepts of the AMR and acceptable English formulation
3	Translation correct, but bad English formulation
2	Translation correct, but English barely understandable
1	Gibberish in the English or missing important information from the AMR
0	Error in the parsing, translation or generation (not encountered anymore!)

Table 5: Scale used for the evaluation of the results

Although this evaluation scale helped us to develop our *symbolic* system, it does not say much about its comparative performance with other systems developed using *statistical approaches*.

Directory	AMR file	#ex	BLEU	|orig|	|gen|
AMR-ISI	examples.txt	56	0.27	6.14	7.34
	guidelines-1_2_5.txt	268	0.19	5.30	6.29
	dict-examples.txt	826	0.07	7.17	9.73
	v1.6 (Le petit prince)	1,562	0.08	13.69	11.87
	dev-bio.txt	500	0.03	27.06	39.81
	test-bio.txt	500	0.04	25.20	47.21
	training-bio.txt	5,452	0.03	25.22	38.71
Split/training	training-bolt.txt	1,061	0.06	24.16	30.46
	training-cctv.txt	214	0.10	16.85	17.67
	training-dfa.txt	6,455	0.07	17.74	22.63
	training-dfb.txt	19,558	0.06	13.41	17.22
	training-guidelines.txt	819	0.12	6.73	8.00
	training-mt09sdl.txt	204	0.08	26.13	28.67
	training-proxy.txt	6,603	0.08	17.05	22.64
	training-wb.txt	866	0.07	21.25	23.31
	training-xinhua.txt	741	0.09	30.14	34.34
Split/dev	dev-bolt.txt	133	0.05	23.32	28.80
	dev-consensus.txt	100	0.07	23.29	25.26
	dev-dfa.txt	210	0.06	15.22	18.62
	dev-proxy.txt	826	0.08	18.36	23.09
	dev-xinhua.txt	99	0.07	27.07	38.14
Split/test	test-bolt.txt	133	0.06	19.96	24.71
	test-consensus.txt	100	0.08	16.76	17.07
	test-dfa.txt	229	0.06	18.85	24.53
	test-proxy.txt	823	0.09	19.86	24.99
	test-xinhua.txt	86	0.09	24.43	23.83
Global		48,424	0.08	18.86	23.65

Table 6: AMR corpora used for testing $\Gamma\omega$-Φ. For each corpus, are given: the number of examples, the BLEU score for all examples in this corpus, the mean sentence length of the reference sentence and for the generated sentence. See Section 9.3 for more details on the content of the *Split* directories

Directory	AMR File	Mean	Scores 4	3	2	1	0	#
AMR-ISI	examples.txt	3.88	47	9	0	0	0	56
	guidelines-1_2_5.txt	3.81	220	46	2	0	0	268
	dict-examples.txt	3.76	635	177	12	0	0	826
	release-test-bio.txt	3.08	0	25	0	0	0	25
Split/test	test-bolt.txt	3.40	10	15	0	0	0	25
	test-consensus.txt	3.40	11	13	1	0	0	25
	test-dfa.txt	3.08	6	15	4	0	0	25
	test-proxy.txt	3.28	9	14	2	1	0	25
	test-xinhua.txt	3.52	13	12	0	0	0	25
Global		3.47	951	326	21	1	0	1,300
ratio			73%	25%	2%	0%	0%	
Only on Split/test		3.34	49	69	7	1	0	125
Ratio			39%	55%	6%	125%	0%	

Table 7: Statistics of the manual evaluation scores on a subset of the corpus of Table 6: the third column gives the mean score over all evaluations of this corpus; columns 4 to 8 give the number of sentences that were given the corresponding score between 4 and 0. The last column gives the total number of sentences evaluated in this corpus. The last four lines show the mean scores on all corpora, and also the scores limited to the first 25 distinct examples of the Split/test directory of the LDC distribution.

158	**The US agrees with the Brazilian assessment that the threat posed by drug traffickers is both very serious and increasing.**	brazil brazil 's assessment of threat of drug trafficking in person thing was very serious and increasing us united_states agreement	
159	(a / agree-01 :ARG0 (c / country :wiki "United_States" :name (n / name :op1 "US")) :ARG1 (a2 / assess-01 :ARG0 (c2 / country :wiki "Brazil" :name (n2 / name :op1 "Brazil")) :ARG2 (a3 / and :op1 (s / serious-02 :ARG1 (t / thing :ARG1-of (t2 / threaten-01 :ARG0 (p / person :ARG0-of (t3 / traffic-01 :ARG1 (d / drug))))) :degree (v / very)) :op2 (i / increase-01 :ARG1 t))))	The country "US" agrees that the country "Brazil" assesses at very the thing that the person who traffics of the drug threatens is serious and increases it.	x
		countries us agrees with the countries with the brazilian assessment of the seriousness of the thing that drug trafficking is a threat to the very and increase	
160			

Figure 2: An excerpt from a spreadsheet for the comparative evaluation of AMR generators: the first column shows the reference sentence above the AMR; the second column shows the output produced by the three generators (in this case, JAMR, Γω-Φ and ISI-MT, but the evaluators did not know this). In the third column, the evaluator chooses the *best* sentence(s).

5.2 Comparative evaluation

In order to compare the output of Γω-Φ with the one produced by another technology, we asked external evaluators to compare the output of two AMR generators, briefly described in Section 9.1. We sampled 100 of the 1,371 AMRs found in *Split/test* directory and 50 of 826 in the *amr-dict-examples.txt* (see their description in Table 7). The evaluators were given an Excel spreadsheet (an entry is shown Figure 2) giving for each AMR, the reference sentence and the output of the three systems (Γω-Φ, ISI-MT or JAMR) without knowing which one had generated them. The evaluator were given the task: *mark the output you consider to be the best*. For each AMR, the annotators had to mark at least one system output as the *best*, even though it might sometimes mean to choose one that was the *least bad*.

In Table 8, we can see that the Γω-Φ is selected as the best one (Nb *best*) more often than

Data set	nb AMR	Total *best*	System	BLEU	Nb *best*	%
amr-dict	50	73.5	Γω-Φ	0.05	34.0	51%
			ISI-MT	0.21	26.0	31%
			JAMR	0.05	13.5	18%
Split/test	100	127.5	Γω-Φ	0.04	52.0	41%
			ISI-MT	0.35	52.5	41%
			JAMR	0.14	23.0	18%

Table 8: Comparative evaluations on two datasets. Mean result of 2 evaluators. The fifth column indicates the mean BLEU score for each corpus of examples. The sixth column gives the number of times that they have been chosen as *best* by one of the two evaluators.

the other two on the *amr-dict* examples. This was expected because many of the examples of this dataset were used in the development of Γω-Φ. We also conjecture that this was also the case for the other two systems. This dataset features short examples (a mean length of 8 words). Γω-Φ has a desperately low BLEU score, but we see that it is not indicative of the quality as we argued a few times before.

For *Split/test* dataset which features much longer sentences (a mean of 18 words), Γω-Φ is on par with ISI-MT even with its very low BLEU score. So we see that the symbolic approach is quite competitive for this aspect.

Of course, this evaluation is preliminary and must be taken with a grain of salt. We did not tune ISI-MT and JAMR, they were built *out of their GitHub repository* without new training and with a small (2 GBs still) language model for the case of JAMR. It should be noted that these systems were designed to deal with AMR 1.0 while we tested on the AMR 2.0 dataset. But we consider that we have reached our goal of showing the competitiveness of Γω-Φ which is much lighter and faster to run than the other two. The symbolic approach also allows a better understanding of the underlying process and allows learning lessons for improving the annotation process as we discuss in Section 7.1.

6 Related Work

Flanigan et al. [7] present JAMR which was the first sentence generator from AMR. Given the fact that the system is open-source, it is the de facto reference implementation for the AMR-to-text task. Examples of its output are shown in tables 1 and 2 and other examples are given in tables 9 to 13 in the appendix 9.1. JAMR transforms the AMR graph into a tree which is decoded into a string using a weighted tree-to-string transducer and a language model. The decoder uses similar techniques as those of machine translation systems but with a rule extraction approach tailored for text generation. The rules are extracted from the training AMR data using an algorithm similar to extracting tree transducers from tree-string aligned parallel corpora [8]. Some rules are then *synthesized* over arguments and concepts and abstracted over part of speech tags. Finally, handwritten rules are added for dates, conjunctions, multiple sentences, and `*-91` type concepts. Evaluation shows acceptable BLEU scores (about 0.22), synthetic rules bringing the most improvement and handwritten ones being also important.

Pourdamghani et al. [20] consider AMR to English generation as a Phrase-Based Machine Translation problem by first linearizing the AMR, removing variables, role names and special concepts and then aligning reference sentences with concepts found in the graph (see Tables 9 to 13 for examples of generated sentences, marked ISI-MT). Even though this process ignores much of the original graph or tree structure of the input AMR, they show interesting improvement in BLEU scores.

The second subtask of *SemEval-2017 - Task 9* organized by May and Priyardashi [16] was devoted to text generation from AMR. The output of 5 systems (including JAMR and ISI-MT) were compared. They used human judgment to rank systems' output to yield pairwise preferences that are used to compute an overall system ranking. They also

computed BLEU scores, but stated that "*BLEU, which is often used as a generation metric, is woefully inadequate compared to human evaluation*". The best system in this evaluation is RIGOTRIO [10] which used manually written rules to transform AMR into Grammatical Framework [21] (GF) abstract syntax from which a correct English Sentence can be rendered automatically. In 12% of the test cases, these rules managed to fully convert the AMR to GF. When the rules could not be applied, they used the output of JAMR. They did not compare directly the output of this symbolic process to the output of JAMR, but as JAMR was also a participant in this task, one can conclude that the symbolic approach did bring some improvement. This experiment shows that a symbolic approach similar to the one that we have explored in this paper can be competitive in this type of task.

Konstas et al. [14] present sequence-to-sequence (seq2seq) models that achieve good results for both AMR parsing and generation. Their main contribution is the method that they use for training seq2seq models using any graph-isomorphic linearization and they show that unlabeled text can be used to significantly reduce sparsity. They preprocess AMRs by removing variables, by anonymizing named entities and dates. They have also manually predefined a mapping between all types of possible type of named entities that can occur in an AMR and one of the four coarse entity types used in the Stanford NER system. Those steps are done to circumvent the scarcity of training material compared to the usual number of examples needed to train successfully these type of neural seq2seq models.

Kao and Clark [6] present a two-step process for generating syntactically varied realizations from an AMR structure. They first generate a *delexicalized constituency structure* from the AMR; then they generate the surface form from both the AMR (which provides the lexical choices) and the constituency parse tree (which provides the sentence structure). Breaking the process in this way enables them to realize the AMR using different syntactic structures and then using different words. Neural sequence-to-sequence with attention models are used in those two steps. The results show the interest of using intermediary representations in this type of transformation. But given the fact that AMRs do not have gold-standard parse annotations, they use Stanford Parser to automatically label the text in the training corpus. The resulting constituency structure is then linearized using a depth-first traversal.

We thus see that AMR generation is a research area in which many statistical approaches (either machine translation or neural inspired) have been tried with relative success but, in the words of May and Priyardashi [16] it remains *challenging* in which *there is still a long way to go to reach fluency*. Even though more than 40,000 sentences have been annotated with their AMR, this does not seem sufficient to get an acceptable performance for any kind of realistic NLP application (summarization, question-answering) even though evaluation scores show slight improvements. This is why we think that alternative approaches should be explored and that our work is interesting in this respect. It shows that an equivalent performance can be achieved with a fully symbolic approach which runs much faster than the statistical ones. But much more work is needed in order to improve coverage.

7 Future Work

This paper has described a first version of a symbolic AMR verbalizer that shows that the approach is viable and fast. There is still work to do on many aspects.

There are still many missing concepts and roles, most of which would imply adding or correcting dictionary entries. Currently the system outputs warnings when unknown concepts and roles are encountered, so this process would imply collecting those warnings. It would also be helpful for annotators to realize that they have *invented* new roles or concepts.

Γω-Φ should take lexical ambiguity into account. Currently if the same character string can refer to a verb, a noun or an adjective such as `war` or `good`, the system chooses the first fit in the above order. When this is not appropriate, we manually removed the *bad* entry (e.g. the verb entry for `war` and the noun entry for `good`). This process gives acceptable results most of the time, but it should be revised using a better linguistically justified process.

Currently an AMR is verbalized as a single sentence, except in the case of a `multisentence`, so this means that sentences are sometimes long-winded and even repetitious. This is due to the depth-first traversal of the graph and also because relative sentences are used most of the time for inverse roles. The nominalization should be better integrated and could sometimes help in reducing the verbosity of the generated text. A better use of pronouns could also help shorten or split long sentences.

Although the system has been tested on all available AMR corpora, the implementation is still a bit shaky: it involves feeding the output of a Prolog system (Γω-Φ) into jsRealB, a Javascript module. An error in the input of jsRealB must sometimes be linked back to the Prolog system (often an error in the generated dictionary). This process should be streamlined even though we could run many ten of thousands of examples without any problem.

It might also be interesting to revisit the starting point of the transformation: there has been extensive work for generating text from First-Order Logic which can be systematically generated from AMR. It would probably be worth a try to revisit this aspect that we gave up perhaps a bit too soon.

It would also be interesting to try to use machine learning techniques to develop the transformation rules between the Semantic Representation and Deep Syntactic Representation, hoping that the process will learn how to cope with many of the special cases. Of course, we would need learning techniques that deal correctly with tree structures and **not** their linearization which often hides important distinctions in the relation between the nodes.

7.1 Suggestions for AMR Developers

We also think that AMR developers should benefit from taking a *generative* view: we have observed a recent tendency to add new concepts and roles as a way to ease the annotation, but this makes the generation process more complex. This proliferation of new *primitives* also makes the use of machine learning techniques more difficult as there are only very few instances (often only one) of each in the training material. This why statistical techniques users must resort to all kinds of tricks for abstracting away from those specificities. We

suggest that the AMR developers try to limit the number of roles to the bare minimum as they have successfully done with the syntactic peculiarities. The annotators should also limit the use of inverse roles, not only because they are difficult to verbalize by our system ☺, but because they do not seem to add to the semantics, especially when inverse roles are composed (i.e. the inverse role of a concept that is itself used in an inverse role).

Although Bos has shown that the `meaning` of AMRs can be expressed in first-order logic, it would be interesting to develop a theory of *non-core roles* (e.g. why are some roles important or necessary ?) and their relations with language or other NLP applications. Such a theory would have been helpful for us when developing our system; for example, `:ARG`$_i$ roles have been studied for a long time and their meaning is relatively well defined, so their implementation for generation is relatively *clean*. Many other roles do not seem to have well-defined semantics (e.g. `:mode`, `:manner`, `:time` which depends on the context of use, etc.), some relations (e.g. comparisons) are expressed in many different ways in the corpus. A much more systematic coding of AMRs would surely simplify the analysis of their meaning and help in their use in future NLP applications.

The current AMR corpus is interesting in its diversity: news articles, biology texts, novels, tutorial examples, informal (even vulgar!) examples and contrived texts that *linguists love*[11]. This shows that AMR can convey almost any kind of text, but then it makes it difficult for system developers to focus on specific aspects. So it might be interesting to annotate texts that will target specific application areas.

8 Conclusion

We have described Γω-Φ a text generator from AMR input. It is a proof of concept of a feasible symbolic approach to AMR generation that takes advantage of the fact that AMR is a structured input that can serve as a plan for the output text. We have shown the interest of generating constituency structures instead of full sentences that can be obtained systematically from them. We have also shown that our previous system JSREALB can be a useful intermediary especially for producing variations of structure.

Acknowledgements

We thank Fabrizo Gotti for many fruitful discussions and suggestions, for helping with the evaluation and for installing the ISI AMR to English generator on our servers. We thank Philippe Langlais who made detailed suggestions for improving the organization of the paper. We also thank Frédéric Bastien of the MILA lab for giving us access to the AMR 2.0 corpus.

[11] See example isi_0002.315: `Buffalo buffalo Buffalo buffalo buffalo buffalo Buffalo buffalo.`

References

[1] Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. Abstract Meaning Representation for sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 178–186. Association for Computational Linguistics, 2013.

[2] Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. Abstract Meaning Representation (AMR) 1.2.5 Specification. `https://github.com/amrisi/amr-guidelines/blob/master/amr.md`, Feb 2018.

[3] Claire Bonial, Julia Bonn, Kathryn Conger, Jena D. Hwang, and Martha Palmer. PropBank: Semantics of new predicate types. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC-2014)*. European Language Resources Association (ELRA), 2014.

[4] Johan Bos. Expressive power of Abstract Meaning Representations. *Comput. Linguist.*, 42(3):527–535, September 2016.

[5] Shu Cai and Kevin Knight. Smatch: an evaluation metric for semantic feature structures. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 748–752. Association for Computational Linguistics, 2013.

[6] Kris Cao and Stephen Clark. Generating syntactically varied realisations from AMR graphs. arXiv:1804.07707, April 2018.

[7] Jeffrey Flanigan, Chris Dyer, Noah A. Smith, and Jaime G. Carbonell. Generation from abstract meaning representation using tree transducers. In Kevin Knight, Ani Nenkova, and Owen Rambow, editors, *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*, pages 731–739. The Association for Computational Linguistics, 2016.

[8] Michel Galley, Mark Hopkins, Kevin Knight, and Daniel Marcu. What's in a translation rule? In Daniel Marcu Susan Dumais and Salim Roukos, editors, *HLT-NAACL 2004: Main Proceedings*, pages 273–280, Boston, Massachusetts, USA, May 2 - May 7 2004. Association for Computational Linguistics.

[9] Albert Gatt and Ehud Reiter. SimpleNLG: A realisation engine for practical applications. In *Proceedings of the 12th European Workshop on Natural Language Generation (ENLG 2009)*, pages 90–93. Association for Computational Linguistics, 2009.

[10] Normunds Gruzitis, Didzis Gosko, and Guntis Barzdins. RIGOTRIO at SemEval-2017 Task 9: Combining machine learning and grammar engineering for AMR parsing and generation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 924–928. Association for Computational Linguistics, 2017.

[11] Ulf Hermjakob. AMR Editor. `https://amr.isi.edu/editor.html`.

[12] Ulf Hermjakob. AMR Annotation Dictionary. `https://www.isi.edu/~ulf/amr/lib/amr-dict.html`, May 2018.

[13] Kevin Knight, Bianca Badarau, Laura Baranescu, Claire Bonial, Madalina Bardocz, Kira Griffitt, Ulf Hermjakob, Daniel Marcu, Martha Palmer, Tim O'Gorman, and Nathan Schneider. Abstract Meaning Representation (AMR) Annotation Release 2.0. `https://catalog.ldc.upenn.edu/LDC2017T10`, 2018.

[14] Ioannis Konstas, Srinivasan Iyer, Mark Yatskar, Yejin Choi, and Luke Zettlemoyer. Neural AMR: sequence-to-sequence models for parsing and generation. In Regina Barzilay and Min-Yen Kan, editors, *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 146–157. Association for Computational Linguistics, 2017.

[15] Jon May. SemEval-2017 Task 8: Meaning Representation Parsing. `http://alt.qcri.org/semeval2016/task8/`, 2016.

[16] Jon May. SemEval-2017 Task 9: Abstract Meaning Representation parsing and generation. `http://alt.qcri.org/semeval2017/task9/`, 2017.

[17] Igor Mel'čuk. Vers une linguistique Sens-Texte, Leçon inaugurale (faite le vendredi 10 janvier 1997), Collège de France, Chaire internationale. `http://olst.ling.umontreal.ca/pdf/melcukColldeFr.pdf`, jan 1997.

[18] Paul Molins and Guy Lapalme. JSrealB: A bilingual text realizer for web programming. In *European Conference on Natural Language Generation (Demo)*, pages 109–111, Brighton, UK, sep 2015.

[19] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. BLEU: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 311–318, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics.

[20] Nima Pourdamghani, Kevin Knight, and Ulf Hermjakob. Generating English from Abstract Meaning Representations. In Amy Isard, Verena Rieser, and Dimitra Gkatzia, editors, *INLG 2016 - Proceedings of the Ninth International Natural Language Generation Conference, September 5-8, 2016, Edinburgh, UK*, pages 21–25. The Association for Computer Linguistics, 2016.

[21] Aarne Ranta. *Grammatical Framework: Programming with Multilingual Grammars*. Center for the Study of Language and Information/SRI, 2011.

9 Appendix

9.1 Some AMRs with output of different generators

The following tables give a few AMR structures, the corresponding reference English sentences and the sentences produced by 5 different generators to give a perspective on the current state of our system and that of other generators:

Γω-Φ System described in this paper, we indicate in parentheses the development evaluation score given to this sentence;

JAMR Pretrained generation model of Flanigan [7] used *out of the github* with the provided Gigaword corpus 4-grams;

ISI-MT System developed by [20] and made available at the ISI website as the `AMR-to-English generator`[12];

Generate Output of the `generate` button in the AMR Editor [11];

Baseline Our own baseline generator (10 lines of Prolog) that merely does a top-down recursive descent in the AMR structure and outputs the concepts encountered. If an `:ARG0` or `:ARG1` role is present, it is output before the root concept in order to try to keep the subject in front of the verb.

[12] `https://www.isi.edu/projects/nlg/software_1`

```
(f / fire-01
      :ARG0 (a / aircraft-type
            :wiki "Mikoyan-Gurevich_MiG-25"
            :name (n / name
                  :op1 "MiG-25"))
      :ARG1 (m / missile
            :source (a2 / air)
            :direction (a3 / air))
      :destination (a4 / aircraft-type
            :wiki "General_Atomics_MQ-1_Predator"
            :name (n2 / name
                  :op1 "Predator")))
```

Reference	The MiG-25 fired an AAM at the Predator.
Γω-Φ (4)	MiG-25 fires the missile from the air to the air to Predator
JAMR	mig-25 mikoyan-gurevich_mig-25 general_atomics_mq-1_predator predator aircraft fired the missiles in the air in the air
ISI-MT	fire aircraft-type :wiki mikoyan-gurevich_mig mig-21 missiles from the air to air to aircraft-type :wiki general_atomics_mq-1_predator predator
Generate	fire missile air from air to
Baseline	aircraft Mikoyan-Gurevich_MiG-25 MiG-25 name fire missile air air aircraft General_Atomics_MQ-1_Predator Predator name

Table 9: AMR and their realization with different systems (ex: `isi_0002.701`)

```
(k / know-01
     :polarity -
     :ARG0 (i / i)
     :ARG1 (t / truth-value
          :polarity-of (s / straight-05
               :ARG1 (h / he))))
```

Reference	IDK if he's str8.
Γω-Φ (4)	I do not know whether he is straight.
JAMR	i know that he is straight -
ISI-MT	I not know truth-value that was noted straight he.
Generate	i don't know truth-value :polarity-of straight from him.
Baseline	i know - truth he straight

Table 10: AMR and their realization with different systems (ex:`isi_0002.691`)

```
(b / bind-01
   :ARG1 (s / small-molecule
     :wiki -
     :name (n / name
          :op1 "TKI258"))
   :ARG2 (e / enzyme
     :wiki "Fibroblast_growth_factor_receptor_1"
     :name (n2 / name
          :op1 "FGFR1")
     :ARG1-of (k / knock-down-02)
     :ARG2-of (m / mutate-01
          :value "V561M"))
   :ARG4 (b2 / binding-affinity-91
      :ARG1 (i2 / inhibitor-constant)
      :ARG2 (a / approximately
           :op1 (c / concentration-quantity
                :quant 35
                :unit (n3 / nanomolar))))
   :ARG4 (t / tight-05)).
```

Reference	TKI258 binds tightly to the FGFR1 KD V561M (Ki 35 nM)
Γω-Φ (3)	That is bound TKI258 FGFR1 knocks down that mutates into it V561M binding affinity inhibitor constant approximately 35 nanomolar tight.
JAMR	inhibitor-constant binding-affinity approximately 35 nanomolar tightly binds tki258 - with knock-down mutating v561m fgfr1 fibroblast_growth_factor_receptor_1
ISI-MT	bound small-molecule :wiki doesn't tki258 enzyme :wiki fibroblast_growth_factor_receptor_1 fgfr1 by knock-down of mutating v561m to binding-affinity inhibitor-constant approximately 35 concentration-quantity nanomolar to tight
Generate	bind that was knock-downed and was mutated tight .
Baseline	small - TKI258 name bind enzyme Fibroblast_growth_factor_receptor_1 FGFR1 name knock mutate V561M inhibitor binding concentration nanomolar approximately tight

Table 11: An AMR for a biology text as generated by different systems (ex: `isi_0002.786`)

```
(k / know-01
      :ARG0 (i / i)
      :ARG1 (t / thing
            :ARG0-of (c / cause-01
                  :ARG1 (c2 / cross-02
                        :ARG0 (c3 / chicken)
                        :ARG1 (r / road)))))
```

Reference	I know why the chicken crossed the road.
Γω-Φ (4)	I know the thing that causes that the chicken crosses the road.
JAMR	i know the things that cause the chicken cross the road
ISI-MT	know why the chicken cross the road?
Generate	I know thing that caused chicken crossing road .
Baseline	i know thing chicken cross road cause

Table 12: AMR and their realization with different systems (ex: `isi_0002.766`)

```
(a / and
     :op1 (r / remain-01
           :ARG1 (c / country :wiki "Bosnia_and_Herzegovina"
                  :name (n / name :op1 "Bosnia"))
           :ARG3 (d / divide-02
                  :ARG1 c
                  :topic (e / ethnic)))
     :op2 (v / violence
           :time (m / match
                  :mod (f2 / football)
                  :ARG1-of (m2 / major-02))
           :location (h / here)
           :frequency (o / occasional))
     :time (f / follow-01
            :ARG2 (w / war
                   :time (d2 / date-interval
                          :op1 (d3 / date-entity :year 1992)
                          :op2 (d4 / date-entity :year 1995)))))
```

Reference	following the 1992-1995 war, bosnia remains ethnically divided and violence during major football matches occasionally occurs here.
Γω-Φ (3)	Bosnia remains under that it is divided about ethnic, the violence the football match that is major in here occasional and when follows the war from 1992 to 1995
JAMR	following the 1992 1995 war , ethnic divides bosnia bosnia_and_herzegovina remains , and the occasional major football match violence in here
ISI-MT	following the war between 1992 and 1995 countries :wiki bosnia_and_herzegovina bosnia remain divided on ethnic and violence at football matches by major here from time to time.
Generate	Bosnia remains Bosnia divided about ethnic , and violence in here football match that was majored following war from 1992 to 1995 .
Baseline	country Bosnia_and_Herzegovina Bosnia name remain divide ethnic and violence match football major here occasional follow war date date date.
RIGOTRIO	following the 1992 1995 war, bosnia has remained an ethnic divide, and the major football matches occasionally violence in here.
CMU	following the 1992 1995 war , bosnia remains divided in ethnic and the occasional football match in major violence in here
FORGe	Bosnia and Herzegovina remains under about ethnic the Bosnia and Herzegovina divide and here at a majored match a violence.
ISI	following war between 1992 and 1995 , the country :wiki bosnia and herzegovina bosnia remain divided on ethnic and violence in football match by major here from time to time
Sheffield	Remain Bosnia ethnic divid following war 920000 950000 major match footbal occasional here violency

Table 13: Example used at SemEval-2017:Task 9; the last 5 lines show the output from participants given by the task organizers [16]

.

9.2 Prolog Implementation of Lambda Application

As our generation algorithm is based on lambda application, it might be interesting to some readers to see how this process can be simply implemented in Prolog. Of course, this does not imply that this cannot be done in other programming languages. But given the fact that unification, which is an *elementary* operation in Prolog, can easily implement β-reduction, (line 1) of Listing 1, the whole process is simplified.

λ-expression, e.g. $\lambda a.b$ are *conventionally* represented in Prolog as `a^b`; b usually contains one or more occurrences of a. But given the fact that `a` is a free variable, there is no way to link it with its external name. So we create pairs whose first value is the external name of the parameter and the second the Prolog variable that will appear in the body of the λ-expression. For example, if the first variable is `:ARG0`, we will represent it as `(':ARG0':a)^b`[13]. See examples of dictionary entries in the listing below (line 14).

Each λ-expression is evaluated in an environment represented as a list of `key:value` pairs such as `[':ARG0':value0, ':ARG1':value1,...]`.

The evaluation is performed by the predicate `applyEnv` which has as first argument the λ-expression to evaluate in the context of the second argument; this predicate creates a relation with the fourth argument, the λ-expression as the fourth argument and the rest of the environment as the third. `applyEnv` first checks (using `select`) that the key of the variable appears in the environment (line 4), if so it replaces it in the body of the expression using `reduce`, otherwise it replaces its value by `null` (line 8). This process is applied recursively until there are no more variables in the λ-expression (line 11).

The example call (line 20) corresponds to the expression of Table 4, the resulting Deep Syntactic Representation is the value of `Exp`.

[13]Parentheses around pairs are needed because in the default Prolog priority, `^` is more binding than `:`.

```
reduce(Arg^Expr,Arg,Expr).          % one-level beta-reduction

%%% Application process
applyEnv((Key:Var)^Expr0,Env0,Env2,Expr2):-  % value found
    select(Key:Val,Env0,Env1),!,
    reduce(Var^Expr0,Val,Expr1),
    applyEnv(Expr1,Env1,Env2,Expr2).
applyEnv((_Key:Var)^Expr0,Env0,Env1,Expr2):- % no value found
    reduce(Var^Expr0,null,Expr1),
    applyEnv(Expr1,Env0,Env1,Expr2).
applyEnv(Expr,Env,Env,Expr).

%% dictionary
verb('like-01',(':ARG0':X0)^(':ARG1':X1)^s(X0,vp(v("like"),X1))).
verb('desire-01',(':ARG0':X0)^(':ARG1':X1)^s(X0,vp(v("desire"),X1)))
    .
noun('boy',('D':D)^('A':A)^np($D/d("the"),A,n("boy"))).
noun('girl',('D':D)^('A':A)^np($D/d("the"),A,n("girl"))).

%%%% example
:- verb('like-01',Like),verb('desire-01',Desire),noun('boy',Boy),
   noun('girl',Girl),
   applyEnv(Boy,[],_,Boy1),
   applyEnv(Girl,['D':d("a")],_,Girl1),
   applyEnv(Like,[':ARG0':Boy1,':ARG1':Girl1],_,Exp).
```

Listing 1: Core of the λ-expression implementation in Prolog.

9.3 Description of the corpora

9.3.1 Statistics on data sources (verbatim from the Data section of LDC2017T10[13])

The source data includes discussion forums collected for the DARPA BOLT AND DEFT programs, transcripts and English translations of Mandarin Chinese broadcast news programming from China Central TV, Wall Street Journal text, translated Xinhua news texts, various newswire data from NIST OpenMT evaluations and weblog data used in the DARPA GALE program. The following table summarizes the number of training, dev, and test AMRs for each dataset in the release. Totals are also provided by partition and dataset:

Dataset	Training	Dev	Test	Totals
BOLT DF MT	1,061	133	133	1,327
Broadcast conversation	214	0	0	214
Weblog and WSJ	0	100	100	200
BOLT DF English	6,455	210	229	6,894
DEFT DF English	19,558	0	0	19,558
Guidelines AMRs	819	0	0	819
2009 Open MT	204	0	0	204
Proxy reports	6,603	826	823	8,252
Weblog	866	0	0	866
Xinhua MT	741	99	86	926
Totals	36,521	1,368	1,371	39,260

Table 14: Number of examples in each corpus of LDC2017T10

For those interested in utilizing a standard/community partition for AMR research (for instance in development of semantic parsers), data in the *split* directory contains 39,260 AMRs split roughly 93%/3.5%/3.5% into training/dev/test partitions, with most smaller datasets assigned to one of the splits as a whole. Note that splits observe document boundaries.

9.3.2 Description of the data sources (verbatim from the README.txt in the distribution)

The sentences that have been AMR annotated in this release are taken from the following sources (their dataset shorthand appears in parentheses).

BOLT Discussion forum MT data (bolt) This discussion forum MT data comes from the Bolt Astral team's 2012p1 Tune dataset, and was selected for AMR annotation because it is rich in informal language, expressions of sentiment and opinion, debates, power dynamics, and a broader spectrum of events (e.g. communication events) all of which are not typically found in traditional newswire data. It also illustrates how AMR is applied to machine translation.

CCTV Broadcast conversation (cctv) These transcripts and English translations of Mandarin Chinese broadcast news conversation from China Central TV (CCTV) were selected for AMR annotation as they contain a mixture of news content and conversational features.

GALE Weblog and Wall Street Journal data (consensus) This GALE weblog data in this dataset was selected for AMR annotation because it contains informal language, as well as event phenomena of interest to events researchers (e.g. causal relations, different levels or granularities of events, irrealis events, fuzzy temporal information, etc.)

The Wall Street Journal newswire data in this dataset was selected for AMR annotation because these sentences contain an interesting inventory of financial and economic events, and have been widely annotated within the NLP community. Of the 200 sentences in this dataset, 100 are from WSJ news, and 100 are GALE Weblog data.

BOLT Discussion forum English source data (dfa) This discussion forum data was selected from LDC's BOLT - Selected & Segmented Source Data for Annotation R4 corpus (LDC2012R77) for AMR annotation because it is rich in informal language, expressions of sentiment and opinion, debates, power dynamics, and a broader spectrum of events (e.g. communication events) all of which are not typically found in traditional newswire data.

DEFT Discussion forum English source data (dfb) This discussion forum data was selected from Multi-Post Discussion Forum (MPDF) files collected by LDC, and were selected for AMR annotation given their selection for annotation in other tasks (ERE, BeSt, RED, etc) within the DARPA DEFT program. These selected MPDFs included several high priority documents that were also chosen for exploratory event annotation in DEFT.

NOTE: For purposes of AMR annotation, these MPDFs were automatically segmented prior to production. Other DEFT tasks did *NOT* use this segmentation, as they annotate at the document rather than sentence level.

Guidelines AMR sentences (guidelines) This data consists of constructed, example sentences that are used to for AMR training, and which also appear in the ./docs/amr-guidelines-v1.2.pdf file. They were not selected from an LDC dataset.

Open MT Data (mt09sdl) This data was selected from the NIST 2008-2012 Open Machine Translation (OpenMT) Progress Test Sets corpus (LDC2013T07) for AMR annotation because it is rich in events and event-relations commonly found in newswire data, and illustrates how AMR is applied to machine translations.

Narrative text "Proxy Reports" from newswire data (proxy) This data was selected and segmented from the proxy report data in LDC's DEFT Narrative Text Source Data R1 corpus (LDC2013E19) for AMR annotation because they are developed from and

thus rich in events and event-relations commonly found in newswire data, but also have a templatic, report-like structure which is more difficult for machines to process.

GALE-era Weblog data (wb) The GALE-era weblog data in this dataset was selected for AMR annotation because it contains informal language, as well as event phenomena of interest to events researchers (e.g. causal relations, different levels or granularities of events, irrealis events, fuzzy temporal information, etc.)

Translated newswire data from Xinhua (xinhua) This data was selected from LDC's English Chinese Translation Treebank v 1.0 corpus (LDC2007T02) for AMR annotation because it is rich in events and event-relations commonly found in newswire data, and illustrates how AMR is applied to machine translation.