Remaining Issues that Could Prevent UNL to be Accepted as a Standard

Gilles Sérasset and Étienne Blanc

GETA-CLIPS, IMAG, Université Joseph Fourier,
BP 53, 38041 Grenoble cedex 9,
Gilles.Serasset@imag.fr

Abstract. This paper presents practical issues when dealing with UNL (Universal Networking Language) documents. Some of these issues are at a purelly syntactical level, others are at a semantic level. Some of these issues introduce unnecessary difficulties when developing tools to handle UNL documents when others introduce unnecessary difficulties when encoding natural language utterances into UNL graphs.

1 Introduction

After several years of development, UNL (Universal Networking Language, [1, 2]) has proved its viability as a cross lingual data exchange format. Its expressive power makes it very useful for the development of multilingual information systems where it serves as a way to represent utterances in a language free manner. However, in order to be adopted as a standard, the UNL definition should be clarified or corrected in order to avoid common errors and misunderstandings.

As a UNL partner since 1998, the GETA (Groupe d'Étude pour la Traduction Automatique) group of the CLIPS (Communication Langagière et Interaction Personne-Système) lab develops and maintains a UNL deconverter for French. For this development, we are one of the few groups that decided to use our own existing tools (namely the ARIANE-G5 translator generator, [3–6]). As such, we had to develop several tools to parse and handle UNL documents and went accross some of the problems that will arise when UNL will be used by third party developers.

This paper presents some of the issues we faced and suggests some solutions. Our goal is to give UNL the opportunity to be largely adopted by third parties as a de-facto standard. After briefly presenting the UNL language and an example of an UNL document, we will begin by low level problems posed by the UNL syntax. After that, we will focus on middle level aspects involved when interpreting the UNL language at its computational level. Finally, we will present some of the higher level issues arising when we interpret UNL utterances as linguistic structures.

2 Motivations

The UNL (Universal Networking Language) provides a language independent knowledge representation formalism. Such a formalism allows for the development of Information Systems where all computation may be performed on UNL expressions and where natural language is only considered as a interface medium for humans. In this vision, enconversion and deconversion may be considered as the interface layer of such a Information System.

Such an information infrastructure is only possible if the UNL expressions are represented and interpreted in a coherent way. As such, the UNL has to be accepted as a standard by the developers of such Information Systems and by the developers of enconverters and deconverters.

However, even between the persons in charge of deconvertion and enconversion, we can see some discrepancies in the way natural language utterances may be encoded in UNL expressions are in the way UNL expressions may be interpreted.

Some of these discrepancies are coming from errors that are not detected by current tools and shows that the UNL infrastructure is still lacking a practical validation routine. Others are coming from the UNL specifications themselves. This papers focuses on the latter case, where the UNL specification should be corrected or clarified. We will also present some unnecessary difficulties we faced when developing our own French deconverter using standard java tools. We claim that such difficulties will slow the adoption of UNL as a standard by independant Information System developers.

3 Issues in UNL syntax

3.1 Overview of UNL

The purpose of this section is to briefly present the syntax of UNL documents. For more details, refer to [2].

UNL documents A UNL document is a set of UNL utterances, structured by proprietary tags ([D] for documents, [P] for paragraphs, [T] for titles and [S] for sentences). Each sentence in this structure contains a UNL utterance.

UNL utterances Hence, UNL utterances are interpreted as graphs where the nodes are annotated "Universal Words (UW)" and arcs are labeled by a relation.

Such graphs are to be denoted using a specific syntax that (usually) represents the list of arcs of the graphs.

Example of a UNL document The following one sentence document

```
[D: dn=sample document, on=French]
[S:1]
```

```
{org:fr}
Le chat attrape une souris.
{/org}
{unl}
agt(catch(agt>thing,obj>thing).@entry,   cat(icl>feline).@def)
obj(catch(agt>thing,obj>thing).@entry,   mouse(icl>rat).@indef)
{/unl}
[/S]
[/D]
```

shows the UNL representation of a French document containing the single sentence "Le chat attrape une souris" ("The cat catches a mouse"). Fig. 1 shows the graphical representation of the corresponding graph (where attribute `.@def` and `.@indef` have been hidden).

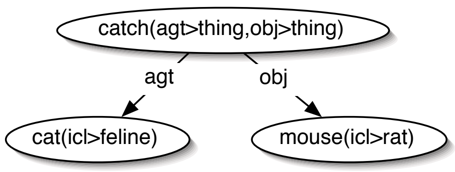

Fig. 1. UNL graph for the example sentence

3.2 Issues in UNL syntax

The current UNL syntax has several issues that leads to unnecessary complexity when parsing an UNL document.

Lexical lookahead is necessary In a UNL graph, the nodes are annotated UWs. For example, the node
`cat(icl>feline).@def`
consists in the UW `cat(icl>feline)` annotated by the `.@indef` attribute.

A UW consists in a headword (`cat` in our example) representing an English language word followed by an optional list of constraints (`(icl>feline)` in our example) that is sufficient to distinguish the correct sense of the UW among the word senses of the headword.

Some headword may contain the "`.`" character, as, `etc.` or `P.O. Box`. As some UWs with these headwords may not be constrained, a UNL graph may contain a node like `etc..@parenthesis` where the first "`.`" character is part of the headword and the second one introduces an attribute.

In order to correctly parse such graphs, one has to distinguish between both usages of the "`.`" character. This distinction may only be done with a 1 character lookahead in the parser, either at the lexical, or at the syntactic level. Doing this at the syntactic level makes the grammar significantly more complex and some of the standard tools that may be used don't properly handle such lookahead.

Encoding issues As most lexical items of the UNL are expressed using lower ASCII characters, most of the document may be parsed provided that the encoding used for the document is compatible with lower ASCII. This forbids the use of UTF-16 or EBCDIC encodings.

Free natural language occurrences may occur in a UNL utterance. Notably between the `{org}...{/org}` tags. Encoding of these parts may be indicated via the `{org:<l-tag>=<charcode>}` tag. However, each language may use it's, own encoding. Hence, when parsing a UNL document, a developer cannot treat a UNL document as a standard stream of characters but rather as a stream of bytes, as the byte to character transformation may vary within the document.

Moreover, there is no way to specify the encoding used for a UNL graph, but knowing this encoding is necessary. The reason is that headwords may contain characters that are not in the lower ASCII character set. We can find such UWs in the UNL specification document itself, as `soufflé(icl>food)`. Hence, to ensure the correct handling of a UNL document, the encoding should be made explicit in the graph opening tag (`{unl:...}`).

4 Computational Interpretation of UNL graphs

4.1 Use of Hyper-nodes

The introduction of UNL specification states that "*the UNL expresses information or knowledge in the form of semantic network with hyper-node*". Hence UNL is dealing with the computational model of hyper-graphs.

In the standard form of the UNL syntax, hyper-nodes are represented by a hyper-node ID, used to label relations appearing inside this node. For example, the English sentence "*The cat who caught the mouse eats*" will be encoded by the UNL graph:

```
agt:01(catch(icl>do).@present,  cat(icl>animal).@def.@entry)
obj:01(catch(icl>do).@present,  mouse(icl>animal))
agt(eat(icl>do).@present.@entry, :01)
```

which is interpreted as in Fig. 2.

4.2 Cross-scope relations

Example of UNL graphs using hyper-nodes, aka scopes, are numerous and the UNL syntax also allows the encoding of relations linking nodes across different scopes. For example, the English sentence "*The cat who caught the mouse eats it*" may be encoded by the the UNL graph:

```
agt:01(catch(icl>do).@present,  cat(icl>animal).@def.@entry)
obj:01(catch(icl>do).@present,  mouse(icl>animal))
agt(eat(icl>do).@present.@entry, :01)
```

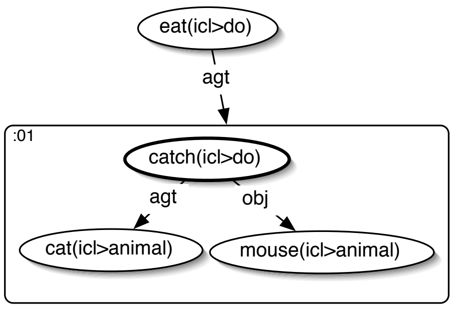

Fig. 2. Sample UNL hyper-graph

`obj(eat(icl>do).@present.@entry,  mouse(icl>animal))`
which is interpreted as in Fig. 3.

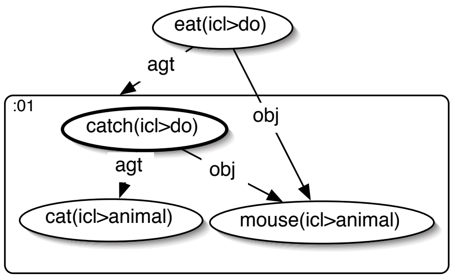

Fig. 3. A UNL graph with a cross-scope relation

One will note that the supplementary `obj` relation links a node (`eat`) at the top level to a node (`mouse`) which is inside the scope `:01`.

Hence, we conclude that the UNL syntax may be used to define classical graphs with Hyper-nodes, and may also be used to define cross-scope relations. However, this conclusion seems not to be shared by all UNL partners and such a possibility should be either explicitly forbidden, or illustrated, with a particular example in the UNL specification.

5 Linguistic Interpretation of UNL graphs

5.1 Scope of the "@not" attribute

The UNL specification chose to encode the speaker's view of aspect using particular attributes. As an example, the English sentence "*the car is about to work*"

will be encoded as
`agt(work(agt>thing).@entry.@begin.@soon,   car(icl>automobile))`

The negation is also expressed as an attribute. For example, the English sentence "*the car does not work*" will be encoded as
`agt(work(agt>thing).@entry.@not,   car(icl>automobile))`

But, the linguistic interpretation of the UNL graph
`agt(work(agt>thing).@entry.@begin.@soon.@not, car(icl>automobile))`
is not clear and may be:

– *the car is about not to work*
– *the car is not about to work*

The scope of the negation should be more clearly stated.

5.2 Encoding of predicates/arguments

One of the greatest difficulty when enconverting or deconverting a UNL graph is that relations are not always easy to select. This difficulty is at its highest when working with some predicate verbs or nouns. The reason is that relations between a predicates and its arguments are generally defined with respect to the predicate whereas UNL relations are defined independently of the UWs they connect.

As an example, in the English sentence "*I strive to work*", there is no real need to characterize the relation between "strive" and "work" as it is simply defined as the second argument of "strive". However in order to encode this sentence in UNL, one has to select, among UNL relation, the one that may represent this particular relation.

In this particular example, users usually hesitate between 2 relations:

– `obj` when the encoder considers that the action of working is directly affected by the action of strive or
– `pur` when the encoder considers that the agent of strive performs some actions which purposes are "to work".

Frequently, the final choice is adopted as a convention. As such, the chosen convention should be documented somehow. Hence, if the partners chose to encoded this relation as an `obj`, this choice should be reflected in the way the UW for "strive" is encoded, as in `strive(agt>human,obj>action)`.

If the other choice is made (which we personally prefer), another problem occurs, as the `pur` relation will be ambiguous in the context of "strive" as it will represent the second argument of the predicate and it may also be used as a circumstantial.

As an example, let's encode the English sentence "*John strives to work to survive*". First, let's get rid of the solution consisting to attach "survive" as the purpose of "work" as in

```
agt(strive(agt>person,pur>action).@entry, John)
pur(strive(agt>person,pur>action).@entry, work(agt>person))
pur(work(agt>person), survive(agt>person))
```
illustrated in fig. 4

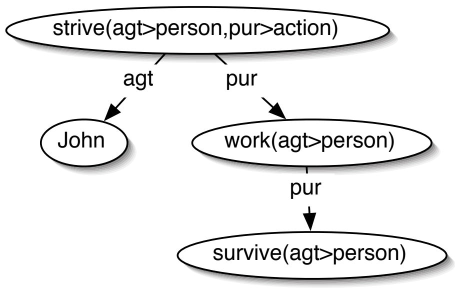

Fig. 4. Incorrect solution to encode "*John strives to work to survive*"

This solution is incorrect as it means that "work to survive" is the purpose of John's actions , as if, most of the time he was working, but not to survive.

Hence, "survive" has to be considered as the purpose of John when he strives to do anything.

Which means that the correct graph should be
```
agt(strive(agt>person,pur>action).@entry, John)
pur(strive(agt>person,pur>action).@entry, work(agt>person))
pur(strive(agt>person,pur>action).@entry, survive(agt>person))
```
as illustrated in fig. 5

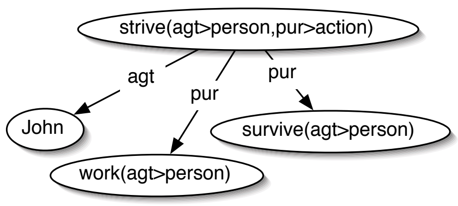

Fig. 5. Correct but problematic solution to encode "*John strives to work to survive*"

But this solution is problematic as no deconverter may decide between "*John strives to survive to work*" and "*John strives to work to survive*".

Hence, the UNL specification should provide a way to distinguish in a graph

- relations that links a predicate and its argument and
- relations that links predicates to circumstantials.

5.3 Encoding variables or untranslatable entities

According to the UNL specification, all UWs that are used in UNL graphs need to be defined in the UNL Knowledge Base (KB). However, several sentences contains elements that do not represent any concept and are not translatable. As an example, the English sentence "*Computers with part number 'ABCD' should be returned to the factory*" will be encoded by a graph containing a node labeled `ABCD` which has no reason to appear in any knowledge base.

In some cases, such untranslatable entities may even be ambiguous with legal UWs.

As there is no way to syntactically distinguish between a legal UW and an untranslatable entity, correct deconversion of such graphs is impossible.

6 Conclusion

As we have shown in this paper, several issues, of varying importance, are still to be addressed as the main ambition of UNL is to become a standard for representing knowledge and information in a language independent way. As such, any point that may be subject to user's interpretation is counter productive.

Also, the initial ambition of UNL is to represent information in an unambiguous way. Such an ambition is fully justified when the information is available without any ambiguity. However, much information systems that may use UNL will have to deal with ambiguous informations. Hence, UNL should define a way to encode ambiguous information.

References

1. Uchida, H., Zhu, M., Della Senta, T.: UNL: A Gift for a Millennium. Institute of Advanced Studies, The United Nations University, Tokyo, Japan (2000)
2. UNL Center: The Universal Networking Language (UNL) specifications Version 3 Edition 2. UNDL Fondation (2003)
3. Boitet, C.: Software and lingware engineering in recent (1980-90) classical mt : Ariane-g5 and bv/aero/f-e. In: ROCLing-III (tutorials). (1990) 21
4. Boitet, C.: Handling texts and corpuses in ariane-g5, a complete environment for multilingual mt. In Belguith, L., ed.: ACIDCA'2000, Corpora and Natural Language Processing, Monastir, Université de Sfax (2000) 7–11
5. Sérasset, G., Boitet, C.: Unl-french deconversion as transfer and generation from an interlingua with possible quality enhancement through offline human interaction. In Tsujii, J.I., ed.: MT Summit VII, Singapore, Asia Pacific Ass. for MT (1999) 220–228
6. Sérasset, G., Boitet, C.: On unl as the future "html of the linguistic content" and the reuse of existing nlp components in unl-related applications with the example of a unl-french deconverter. In Uszkoreit, H., ed.: COLING-2000, Saarbrücken, ACL (2000) 768–774