A Linguistic Framework for Controlled Language Systems

Alexis Nasr
Laboratoire d'Informatique d'Avignon
alexis.nasr@lia.univ-avignon.fr

Owen Rambow, Richard Kittredge
CoGenTex Inc.
(owen/richard)@cogentex.com

Abstract

In this paper, we discuss the use of the Meaning-Text Theory (MTT) of [Mel88] in a controlled language (CL) application[1]. We show that MTT defines a linguistic framework which is ideally suited for the definition and automation of CL rules. In the paper, we first briefly present a CL system based on MTT. We then discuss MTT in more detail. We show how CL-specific information can be represented so that it can be exploited within an MTT-based CL system, and give an extended example.

1 Introduction

A crucial advantage of the use of a general-purpose linguistic theory and particularly MTT in a CL application is that we can distinguish between two types of knowledge:

- The linguistic knowledge which allows us to generate all possible (meaning-preserving) paraphrases in a given language. This knowledge is the responsibility of the linguistic theory and of the linguistic model of the language within that theory.
- The constraints introduced by a Controlled Language which allow us to choose a CL compliant paraphrase among all possible paraphrases. This knowledge derives from the specification of the CL.

This factorization makes the whole system more easily portable to new CL applications: only the CL-specific modules need change (and maintenance). For example, keeping a CL system updated according to the changing definition of the implemented CL is straightforward. Furthermore, it is possible for someone with only minimal linguistic training to create an entirely new CL system for a new CL specification. In fact, this need not be a controlled language in the traditional sense, but could also correspond to a generic "style sheet". Finally, we observe that some of the generic linguistic paraphrasing rules are in fact valid for many languages and can be re-used when the system is ported to a CL based on a new language.

[1]The use of MTT in a CL application was first discussed in [Nas96]

2 System Overview

The system we propose is based on the architecture for a transfer-based MT system[2]. (The machine translation system was developed as a prototype English-to-French translation system [RNP+97].) The input text is parsed into the Deep-Syntactic Structure of MTT. It is at this level that transfer to the CL occurs. Then, the CL-compliant Deep Syntactic Structure is used as the starting point for generation using the RealPro text generation system [LR97]. The architecture is shown in Figure 1.

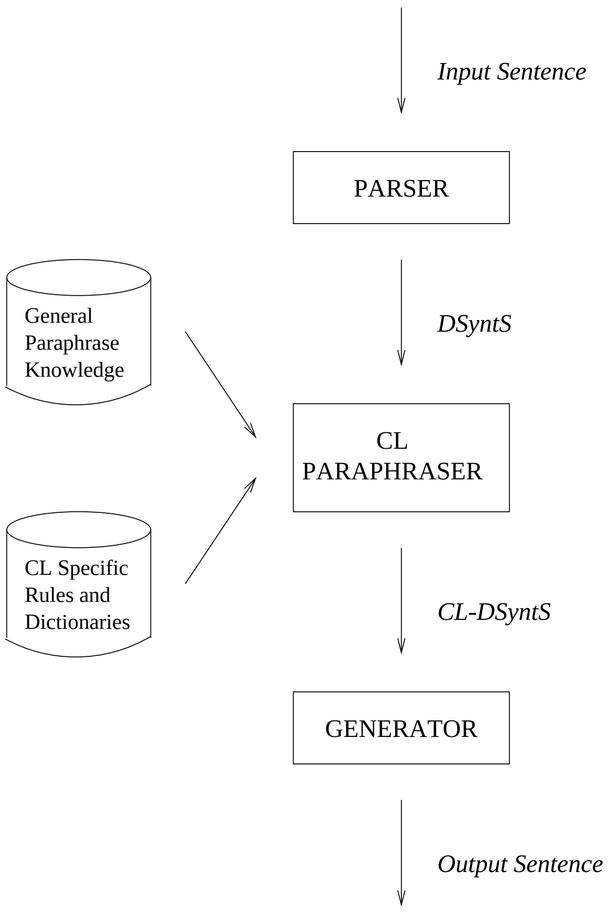

Figure 1: CL Paraphrase System Overview

3 The MTT Paraphrase Framework

According to the division drawn above, between the responsibility of the linguistic theory and the CL constraints, the former must provide a framework for paraphrase. The theory on which our reformulation architecture is based, Meaning Text Theory [Mel88], gives a special place for paraphrase by defining an abstract linguistic representation level allowing for a simple representation of paraphrase phenomena (the Deep Syntactic Level) along with formal means for describing semantic relations between lexemes (the lexical functions) and a formalism for paraphrase rules. These three aspects of the theory are described below.

[2]See also [Adr94] for a similar approach.

3.1 Deep Syntactic Representation of an Utterance

The *Deep Syntactic Structure* (DSyntS) specifies the syntactic organization of a sentence in terms of a dependency tree. Such trees are composed of nodes labeled by *generalized lexemes* and directed arcs (dependencies) labeled with *Deep Syntactic Relations*. A generalized lexeme is a full lexeme, a multilexemic unit (idiom), or a lexical function (see Section 3.2 for details). Semantically empty lexemes, such as governed prepositions or auxiliaries are not represented at this level. Generalized lexemes can be enriched with meaning bearing morphological features such as number or gender in nouns and tense and/or aspect in verbs. Syntactically conditioned morphological features, induced by syntactic rules (such as agreement rules), are not represented in the Deep Syntactic Structure.

Each type of Deep Syntactic Relation stands for a family of specific syntactic constructions. There are different arc labels for the different arguments ('I' for subject, 'II' for direct object, 'III' for indirect object, and so on); label 'ATTR' covers all adjuncts. Two additional labels handle coordination and parentheticals and related constructions.

The Deep Syntactic Structure is closely related to a specialized lexicon, the *Explanatory Combinatorial Dictionary* [MP87] which, among other information, defines the subcategorization frames of predicative lexemes, as well as the prepositions introducing the actants of a predicate and the lexical functions linking the different lexical entries.

For the sake of illustration, a Deep Syntactic Structure, corresponding to the sentence *The decontamination of the aircraft must be done in an authorized area*, is represented in figure 2.

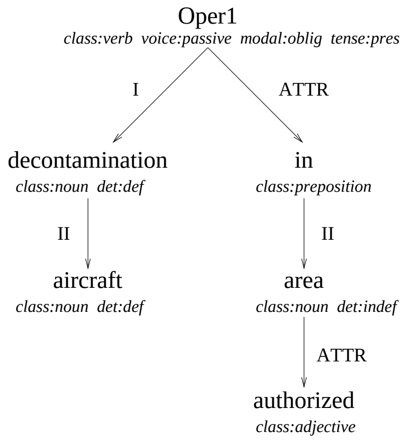

Figure 2: A Deep Syntactic Structure

The DSyntS of figure 2 is represented internally in the following way.

```
Oper1 [class:verb voice:passive modal:oblig tense:pres lf:Oper1]
  (I decontamination  [lexeme:contamination class:noun det:def]
    (II aircraft [lexeme:aircraft class:noun det:def])
   ATTR in [lexeme:in class:preposition]
     (II area [lexeme:area class:noun det:indef]
       (ATTR authorized [lexeme:authorized class:adjective])))
```

In the preceding DSyntS, the articles are not represented as nodes, as well as the prepositions *of*, the auxiliary *be* and the modal *must*. The lexical value of the main verb of the sentence is represented by a lexical function *Oper1* (see 3.2).

Through the use of generalized lexemes and deep syntactic relations, the abstraction achieved at the Deep Syntactic level concerns lexical as well as syntactic aspects of utterances. This level of abstraction allows generation, from a single Deep Syntactic Structure, of several semantically equivalent Deep Syntactic Structures by application of rules of a simple format, called *paraphrase rules*. The generated structures will themselves give rise to a large number of paraphrastic sentences.

3.2 Lexical Functions

The heart of lexical paraphrase in MTT lies in the lexical functions. Such functions formalize a semantic relation between two lexemes L_1 and L_2 ($F(L_1) = L_2$) (L_1 is called the *key* of the lexical function and L_2 its *value*). The semantic relations represented by lexical functions are extremly varied, as the function *Magn* which expresses intensity (*Magn(rain) = heavy*) or the function *Syn* which relates a lexeme and its synonyms. The paraphrastic richness of a Meaning Text Model[3] depends on the number and variety of the lexical functions represented in the model. As shown in 3.1 some lexemes of a Deep Syntactic Structure are not represented directly, but by means of a lexical function, like the verb *do* which is represented as a support verb of the noun *decontamination* by means of the lexical function *Oper1*[4]. The lexical functions are hosted by the Explanatory Combinatorial Dictionary, each lexical entry L of the dictionary lists the values of the different lexical functions defined for L (for example *Syn(L) Magn(L) . . .*).

3.3 Paraphrase Rules

The transformation of a Deep Syntactic Structure into a synonymous one is carried out by application of paraphrase rules. They represent semantic equivalences expressed in terms of lexical functions. More precisely, a paraphrase rule relates together two paraphrastic lexico-structural structures (portions of Deep Syntactic Structure), some lexemes of the two structures being related by lexical functions. We have represented below a simple rule, relating two structures reduced to a node, linked by the synonymic lexical function (*Syn*).

```
PARA-RULE:Syn_substitution

[$X] | [($X [lexeme:$X_lex])]
<-->
[$X] | [($X [lexeme:Syn($X_lex)])]

END:
```

[3]A Meaning Text Model is an actual collection of linguistic rules along with a lexicon, both expressed in the formal apparatus defined by the Meaning Text Theory.

[4]Lexical functions represent actually two quite different phenomena, *syntagmatic lexical cooccurrence* (Oper1, Magn . . .) where the value and the key of the lexical function appear in the DSyntS and *paradigmatic lexical cooccurrence* (Syn, Conv31 . . .) where the two lexemes do not cooccur in the DSyntS

This rule replaces the lexical value of node `$X` (`$X_lex`) by a synonym of the latter (`Syn($X_lex)`). It must be noted that the rule does not have to worry about the possible difference in number between `$X_lex` and `Syn($X_lex)` if they are nouns or the nature of preposition introducing their actants if they are predicative lexemes, these "details" being taken care of at a later stage in the generation of the sentence, illustrating the advantages of performing paraphrase at an abstract level.

A slightly more complex rule, which replaces a verb `$V_lex` by a converse `Conv31($V_lex)` (for example: Conv31(receive) = give) and swaps the first and the third actants (respectively `$N1` and `$N2`) of the two verbs is represented in figure 3:

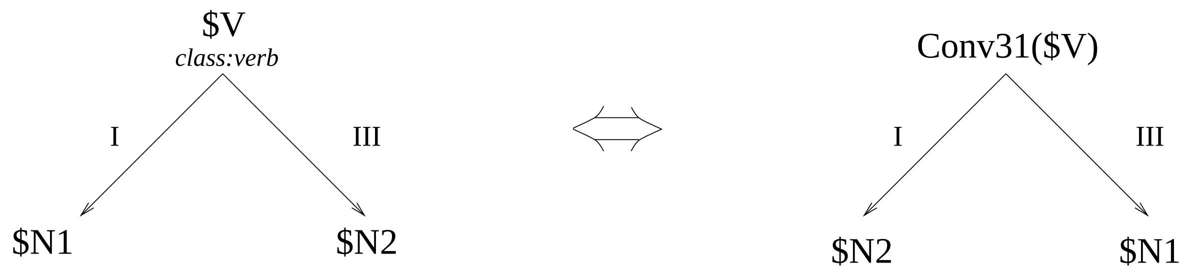

Figure 3: A conversive paraphrastic rule

Internally, the rule of figure 3 is represented in the following way:

```
PARA-RULE:Conv31_permutation
[($V I   $N1)
 ($V III $N2)] | [($V [lexeme:$V_lex class:verb])]
<-->
[($V I   $N2)
 ($V III $N1)] | [($V [lexeme:Conv31($V_lex)])]
END:
```

During the application of a paraphrase rule on a given Deep Syntactic Structure, only the elements of the latter which appear in the rule will be modified, leaving the rest of the structure unchanged.

According to MTT, a paraphrase rule should always provoke at least one lexical transformation through the use of lexical functions. We will relax this constraint and allow for *Syntactic paraphrase rules* as well as *Lexical parpahrase rules*, described below.

3.3.1 Syntactic Paraphrase Rules

A syntactic paraphrase rule does not introduce any lexical transformation to a DSyntS on which it applies, as in the passive to active following rule.

```
PARA-RULE:passive-to-active
[($V I   $N1)
 ($V II  $N2)] | [($V [class:verb voice:passive])]
<-->
[($V I   $N2)
```

```
 ($V II  $N1)] | [($V [class:verb voice:active])]
END:
```

This rule merely changes the value of the feature `voice` of the verb (node `$V`) from `passive` to `active` and permutates the first and second actant of the verb. The rule does not contain any condition on the `lexeme` feature, the reason being that in this case it is a syntactic configuration which is transformed, independently of the lexical nature of the verb.

3.3.2 Lexical Paraphrase Rules

Lexical paraphrase rules provoke lexical changes but do not embed any lexical function, relating directly two lexico syntactic structures which do not fall under a productive paraphrase pattern.

```
PARA-RULE:assess->make_estimate_of
[($A II $B)] | [($A [class:verb lexeme:assess])]
<-->
[($A II $C)
 ($C II $B)]| [($A [class:verb lexeme:make])
              ($C [class:noun lexeme:estimate])]
```

The preceding rule replaces the verb *assess* by the multi lexemic structure *make estimate*. This equivalence cannot be represented by a “normal” support verb expansion since there is no lexical function relating the verb *assess* and the noun *estimate*.

4 The Controlled Language System

The previous section introduced the formal means MTT proposes to perform paraphrase: the definition of the deep syntactic level, paraphrase rules and lexical functions. We will describe in this section the organization of the linguistic knowledge needed for a CL reformulation application. Three different sources of knowledge will be distinguished: *general linguistic knowledge*, *CL linguistic knowledge* and *CL control information*. The nature of the first two sources is the same: they represent paraphrase rules and linkage of lexemes by lexical functions. They differ by their level of generality. The purpose of this distinction is to avoid redefining in the CL specific part some generic linguistic knowledge. The third source of knowledge, the control information, implements the writing rules of a Controlled Language.

4.1 General Linguistic Knowledge

The general linguistic knowledge comprises paraphrase rules and lexical functions that are of a general usage. For sake of illustration, we have represented below a complex support verb expansion rule:

```
PARA-RULE:Oper1_expansion
[($V II   $N1)
 ($V ATTR $ADV)]| [($V   [class:verb lexeme:$V_lex])
                   ($ADV [class:adverb lexeme:$ADV_lex])]
```

```
<-->
[($V  II  $N2)
 ($N2 II  $N1)
 ($N2 ATTR $ADJ)] | [($V [lexeme:Oper1(N0($V_lex)) class:verb lf:oper1])
                     ($N2 [lexeme:N0($V_lex) class:noun])
                     ($ADJ [lexeme:Adj0($ADV_lex) class:adjective])]

END:
```

Such a rule transforms a simple verb (`$V`) into a support verb structure, (`($V II $N2)`) where `$N2` is a nominal derivative of the verb `$V` (`N0($V_lex)`) and `$V` a support verb of `$N2`. The rule also moves the direct object (`$N1`) of the original verb to become the second actant of the nominalisation. Finally, it transforms the adverb attached to `V` into an adjective, using the lexical function *Adj0* and attaches it to the nominalisation. The application of such a rule to the sentence *Functionally test warning system* will give rise to *Do a functional test of the warning system*.

Lexical functions are represented as a list of triples (LF, lex1, lex2) where LF is the name of a lexical function and lex1 and lex2 two lexemes such that (LF(lex1) = lex2). Further conditions can be added to any of the two lexemes. They are related by the lexical function only if the conditions are fulfilled. In the following list, the noun *rain* is linked to the adjective *heavy* by the lexical function *Magn* expressing the idea of intensity. The condition on lexeme *rain* prevent the establishment of the link between the verb *rain* and *heavy*.

```
[(Magn, rain [class:noun], heavy [])
 (Conv31, give [], receive [])]
```

4.2 CL Linguistic Knowledge

CL linguistic knowledge is composed of paraphrase rules and lexical functions[5] which are particular to a given textual genre and cannot be seen as generic linguistic knowledge.

The following paraphrase rule which transforms into active an agentless passive verb while introducing the pronoun *you* as subject cannot be considered generic, it is specific to procedural texts where the agent is the mechanic reading the manual.

```
PARA-RULE:passive-to-active-no-agent
[($V II  $N2)] | [($V [class:verb voice:passive])]
<-->
[($V I   $N2)
 ($V II  $N1)] | [($V  [class:verb voice:active])
                  ($N1 [class:pronoun pers:2nd number:sing])]
END:
```

The CL lexical functions are grouped together in a list. In the following list example, the verb *abandon* is linked to the verb *stop* through the lexical function *Syn* which links

[5]The expression CL lexical function can be misleading: it is not the nature of semantic relation represented by the lexical function which is CL specific, but the decision to relate two particular lexemes by a given lexical function.

two synonymic lexemes. The linking of these two lexemes cannot be seen as general: in everyday's discourse, *abandon* is not a synonym of *stop*.

```
[(Syn, abandon [class:verb], stop [class:verb])
 (V0, decontamination [], clean [class:verb])
 (Syn, authorized [class:adjective], approved [class:adjective])
 (Oper1, decontamination [], do [])]
```

4.3 Control Information

The control information describes the way linguistic knowledge (CL specific and generic) must be used to enforce the writing rules of the controlled language. It is important to notice that at this level, no paraphrase rules, nor lexeme linkage through lexical functions are defined. Two kinds of control information are distinguished: the *lexical control list* and the *syntactic control list*.

The lexical control list is composed of couples made of a lexeme and a paraphrase rule name. This couple indicates that the lexeme must be replaced using the associated paraphrase rule. The lexeme will be called a trigger. The firing of the rule associated to a lexeme is also conditioned by (optional) constraints (appearing between brackets) on the lexeme . The paraphrase rule associated to a trigger can be either a CL specific or a generic rule:

```
Lexical Control List:
[(decontamination [], support_verb_contraction)
 (assess [], assess->make_estimate_of)
 (authorized [class:adjective], syn-substitution)]
```

The first couple of the list indicates that the noun *decontamination* must be replaced, using the `support_verb_contraction` rule, the second couple states that lexeme *assess* must be replaced by the multi lexemic structure *make an estimate of*. Finally, the third couple shows that the adjective authorized must be replaced by a synonym.

The syntactic control information is composed of an ordered list of non lexical rule names.

```
Syntactic Control List:
[active-to-passive
 active-to-passive-no-agent
 modal-elimination]
```

4.4 The CL Correction Algorithm

The algorithm takes as input a DSyntS, a syntactic control list and a lexical control list. It proceeds in two steps:

1- The DSyntS is visited top down. For each node N, the rules of the syntactic control list in their order of appearance are checked. The order in which the rule names appear in the list is therefore important. When several rules can apply on a given node, it is the first one in order of appearance which will be applied first. Its application can modify the DSyntS, preventing possible application of other rules.

2- The DSyntS is visited a second time top down. For each node N visited, a lookup in the lexical control list is performed. If N appears as a trigger of a couple in the lexical control list, its associated rule is checked.

To show how the algorithm works, we now discuss an example taken from the AECMA Simplified English Guide:

- Bad: The decontamination of the aircraft shall be done in an authorized area.
- Good: Clean the aircraft in an approved area.

The input DSyntS is represented in 3.1 and the control lists are the ones of 4.3.

The system first visits the root node *Oper1* which stands for the verb *do*. It tries to apply the `active-to-passive` rule which fails since the verb does not possess a second actant. The `active-to-passive-no-agent` rule is then checked, it succeeds transforming the voice of the sentence and introducing the pronoun *you* as a subject. This results in the DSyntS for *You shall do the decontamination of the aircraft in an authorized area* represented below.

```
Oper1 [class:verb voice:active modal:oblig tense:pres lf:Oper1]
  (I XX [class:personal_pronoun pers:2nd number:sing]
   II decontamination [lexeme:contamination class:noun det:def]
     (II aircraft [lexeme:aircraft class:noun det:def])
   ATTR in [lexeme:in class:preposition]
      (II area [lexeme:area class:noun det:indef]
         (ATTR authorized [lexeme:authorized class:adjective])))
```

Then, the modal-elimination rule paraphrases the *"you shall V"* construction with an imperative, resulting in the DSyntS for *Do the decontamination of the aircraft in an authorized area.*

```
Oper1 [class:verb voice:active mode:imper tense:pres lf:Oper1]
  (II decontamination [lexeme:contamination class:noun det:def]
     (II aircraft [lexeme:aircraft class:noun det:def])
   ATTR in [lexeme:in class:preposition]
      (II area [lexeme:area class:noun det:indef]
        (ATTR authorized [lexeme:authorized class:adjective])))
```

The rest of the DSyntS nodes are visited but no syntactic paraphrasing rule applies. The second visit begins, using the lexical control list. Here, we find that "decontamination" is entered for support verb contraction, with Oper1(*decontamination*) = *do* (thus, the rule matches the input) and V0(*decontamination*) = *clean.* The resulting application of the general support-verb contraction rule yields the DSyntS for *Clean the aircraft in an authorized area.* (Note that the rule applies irrespective of the verb form.)

```
clean [lexeme:clean class:verb voice:active mode:imper tense:pres]
  (II aircraft [lexeme:aircraft class:noun det:def]
   ATTR in [lexeme:in class:preposition]
     (II area [lexeme:area class:noun det:indef]
       (ATTR authorized [lexeme:authorized class:adjective])))
```

When finally the node *authorized* is reached, the lookup in the lexical control list shows that it should be replaced using the synonymic substitution rule. *authorized* is therefore replaced by *approved*, and we obtain the desired result.

```
clean [lexeme:clean class:verb voice:active mode:imper tense:pres]
  (II aircraft [lexeme:aircraft class:noun det:def]
   ATTR in [lexeme:in class:preposition]
     (II area [lexeme:area class:noun det:indef]
       (ATTR approved [lexeme:approved class:adjective])))
```

5 Discussion

We have presented a CL system based on the clear separation between general linguistic paraphrasing knowledge and CL-specific paraphrasing knowledge. We have argued that this separation makes it easier to maintain the system and port it to new languages and/or new controlled languages.

We have not addressed the issue of the fluency of the entire text: the proposed approach is based on a sentence-level paraphrase. There are several dimensions to this problem, but one major contribution could be made by including in the parahrase rules restrictions on the theme (the part of the DSyntS that the sentence is about) and the rheme (the part of the DSyntS that represents the information communicated about the theme). For example, if a passive-to-active paraphrase and a conversive paraphrase are both possible, the conversive paraphrase could be chosen because it eliminates the passive but maintains the theme as the subject. We intend to investigate this issue further.

References

[Adr94] Geert Adriaens. Simplified english grammar and style correction in an MT framework : the LRE SECC project. In *Proceedings of the 16th Conference on Translating and the Computer*, pages 78–88, London, 1994.

[AEC89] Association Européenne des Constructeurs de Matériel Aérospatial, Paris. *A guide for the preparation of aircraft maintenance documentation in the international aerospace maintenance language*, 1989.

[LR97] Benoit Lavoie and Owen Rambow. RealPro – a fast, portable sentence realizer. In *Proceedings of the Conference on Applied Natural Language Processing (ANLP'97)*, Washington, DC, 1997.

[Mel88] Igor A. Mel'čuk. *Dependency Syntax: Theory and Practice.* State University of New York Press, New York, 1988.

[MP87] Igor A. Mel'čuk and Alain Polguère. A formal lexicon in the meaning-text theory. *Computational Linguistics*, 13(3-4):13–54, 1987.

[Nas96] Alexis Nasr. *Un modèle de reformulation automatique fondé sur la Théorie Sens Texte: Application aux langues contrôlées.* PhD thesis, Université Paris 7, 1996.

[RNP$^+$97] Owen Rambow, Alexis Nasr, Martha Palmer, Tonia Bleam, Michael Collins, Karin Kipper, Dan Melamed, Jong Park, Joseph Rosenzweig, William Schuler, and B. Srinivas. Machine translation of battlefield messages using lexico-structural transfer. Technical report, CoGenTex, Inc., 1997.