Text Structuring in a Multilingual System for Generation of Instructions

Ivana Kruijff-Korbayová and Geert-Jan M. Kruijff

Institute of Formal and Applied Linguistics (ÚFAL)
Faculty of Mathematics and Physics, Charles University
Prague, Czech Republic
{korbay,gj}@ufal.mff.cuni.cz
WWW home page: http://ufal.mff.cuni.cz/

1 Introduction

The approach presented in this paper has been developed in the context of the international project called AGILE (Automatic Generation of Instructions in Languages of Eastern Europe)[1]. The overall aim of the project is to develop a multilingual system for generating continuous instructional texts in Bulgarian, Czech and Russian [2]. The project is concerned mainly with (i) the development and adaptation of linguistic resources for the chosen languages, and (ii) the investigation and specification of text structuring strategies employed in those languages for the given type of texts. In the current paper, we concentrate on the latter issues.

What we are particularly interested in is the following. In the application scenario of the system developed in AGILE, the user provides a definition of a text content in the form of an A(assertion)-box. from this A-box we are then to generate a text, in either of the languages we are concerned with. The problem we address in this paper is how to go about this for relatively simple instructional texts about working with AutoCAD.

To that end, we begin with a brief discussion of the texts (and their language-dependent styles) we would like to be able to generate (§2). As we already pointed out, these texts are generated from (or "realize") a text content specified by an A-box. In §3 we show how we can use the concepts, employed in an A-box, to set up a structure for a text that would realize the content given by that A-box. Finally, we discuss in section §4 how the approach outlined in the previous sections is implemented in the KPML multilingual generation environment [1] that we use in the context of AGILE.

[1] AGILE (Automatic Generation of Instructions in Languages of Eastern Europe) is an international project supported by the European Commission within the COPERNICUS programme, grant No. PL961104. We would like to thank our colleagues from Charles University, the University of Brighton, the University of the Saarland, the Bulgarian Academy of Sciences and the Russian Research Institute of Artificial Intelligence for their cooperation.

2 Texts and Text Styles

The texts considered for generation in the current stage of the AGILE project are simplified versions of routine passages occurring frequently in the user guide for the AutoCAD system, namely descriptions of step-by-step procedures for performing a task. An example of such a description is given in figure 1.

To create a multiline style
First open the Multiline Styles dialog box using one of these methods:

i. **Windows** From the Object Properties toolbar or the Data menu, choose Multiline Style.
ii. **DOS and UNIX** From the Data menu, choose Multiline style.

(1) Choose Element Properties to add elements to the style.
(2) In the Element Properties dialog box, enter the offset of the first line element.
(3) Select Add to add the element.
(4) Choose Color. The the Select Color dialog box appears. Then select the element's color.
(5) Choose Linetype. Then select the element's linetype from the Select Linetype dialog box.
(6) Repeat these steps to define another element.
(7) Choose OK to save the style of the multiline element and to exit the Element Properties dialog box.

Fig. 1. Example "To create a multiline style"

The corpus investigations we carried out for the languages under study in AGILE provided empirical support for the view that in the instructional texts under investigation we can observe alternations that can be classified into two main groups, as follows:

1. *Choice of grammatical means*: Whether lexico-grammatical choices are predetermined in certain ways by the style of text.
2. *Distribution of content and lexical realization*: Whether information is realized explicitly in the most straightforward way given the A-box, (so that there is only a minimal inference load required to interpret a piece of text), or whether information is left more or less implicitl; and, what the realization is of the explicitly conveyed information.

The first type of alterations concerns the following kinds of choices: (i) Expressing or hiding the intentional agent of an action (agency, voice), (ii) ways of addressing the readership (person, number, voice), and (iii) modes of realising instructions (mood, voice). The second type of of alterations is reflected in the level of detail in expressing the content explicitly. A nice example is whether or not the text mentions any side-effects that will occur as a result of performing a particular action.

In the next sections we develop a view on how to deal with such alterations in a principled fashion.

3 Text Structuring

In this section we discuss how we can set up a structure for a text that is to realize a given A-box, by looking at how the concepts in the A-box are organized in order to specify a particular content. We begin by making a distinction between *text structure elements* being the elements from which a (task-oriented) text is built up, and *text templates*, which condition the way text structure elements are to be linguistically realized (§3.1). Subsequently, we focus on the relation between concepts on the one hand, and text structure elements on the other hand. We are specifically interested in those T(erminology)-box concepts that are used to configure the content specified in an A-box. This is because we want to have a close connection between how the content can be defined in an A-box and how that content is to be spelled out in a text using a particular organization, so as to ensure that the intended content is indeed reflected by the text (§3.2).

3.1 Structuring and Styling

Above we already alluded to the idea that texts may vary in their distribution and lexico-grammatical realization of content. Here we propose two notions that should enable us to capture these differences in a principled fashion. These two notions are the notion of **text structure element** and the notion of **text templare**.

A text structure element is a predefined component that is to be filled by one or more specific parts of the user's definition of the content to be spelled out by the text. Using the reader-oriented terminology common in technical authoring guides, we will distinguish a small (recursively defined) set of text structure elements, given in figure 2. Orthogonal to the notion of text structure element is the notion of text template. Whereas text structure capture *what* needs to be realized, the idea of a text template captures *how* that content is to be realized. Thus, a template defines a style in which content is to be realized. As we shall discuss below (§4), we will define text templates in terms of constraints on the realization of specific (individual) text structure elements. For example, whereas in Bulgarian and Czech headings (to which the TASK-TITLE element corresponds) are usually realized as nominal groups, the Russian AutoCAD manual realizes headings as partial purpose clauses.

3.2 Relating Domain Model Concepts and Text Structure

Our aim is to create a structure for a text that is to realize the content specified by an A-box. We already have defined the elements that build up such a text

Task-Document A TASK-DOCUMENT has two slots:

i. TASK-TITLE (obligatory)
ii. TASK-INSTRUCTIONS (obligatory), being alist of at least one INSTRUCTION.

Instruction An INSTRUCTION has three slots:

i. TASKS (obligatory), being a list of at least one TASK.
ii. CONSTRAINT (optional)
iii. PRECONDITION (optional)

Task A TASK has two slots:

i. INSTRUCTIONS (obligatory)
ii. SIDE-EFFECT (optional)

Fig. 2. Text Structure Elements

structure (cf. figure 2). In this section we begin with defining the concepts that are used to specify content in an A-box. Thereafter we relate these concepts to the text structure elements so as to enable us to build a text structure by going by the way content has been structured in an A-box.

An A-box, short for "Assertion-box", instantiates concepts defined in a T-box, or "terminology box". In KPML, and AGILE, these concepts are generally referred to as *domain model concepts*. For AGILE we employ four concepts to define the admissible A-box configurations. These configurations provide the structured contexts in which actions and events "take place". The concepts are given in figure 3[2].

The following mapping between text structure elements (figure 2) and concepts (figure 3) enables us to provide a flexible definition of rules for text structuring of a content specified by an A-box:

TASK-TITLE ↔ GOAL of topmost PROCEDURE
TASK-INSTRUCTIONS ↔ METHODS of topmost PROCEDURE
SIDE-EFFECT ↔ SIDE-EFFECT of a PROCEDURE
TASK ↔ GOAL of a PROCEDURE
CONSTRAINT ↔ CONSTRAINT of a METHOD
PRECONDITION ↔ PRECONDITION of a METHOD
INSTRUCTIONS ↔ METHODS
INSTRUCTION ↔ METHOD

4 Text Structuring Module

In AGILE we are -ultimately- aiming at so-called *end-to-end generation*. That is to say, a user specifies an A-box using an authoring interface, and obtains as

[2] These concepts were defined by Richard Powers.

Procedure A procedure has three slots:

i. GOAL (obligatory, to be filled by a USER-ACTION)
ii. METHODS (optional, to be filled by a METHOD-LIST)
iii. SIDE-EFFECT (optional, to be filled by a USER-EVENT)

Method A method has three slots:

i. CONSTRAINT (optional, to be filled by an OPERATING-SYSTEM)
ii. PRECONDITION (optional, to be filled by a PROCEDURE)
iii. SUBSTEPS (obligatory, to be filled by a PROCEDURE-LIST)

Method-List A METHOD-LIST is essentially a list of METHOD*'s:

i. FIRST (obligatory, of type METHOD*)
ii. REST (optional, of type METHOD-LIST)

Procedure-List A PROCEDURE-LIST is essentially a list of PROCEDURE's:

i. FIRST (obligatory, of type PROCEDURE)
ii. REST (optional, of type PROCEDURE-LIST)

Fig. 3. Domain Model concepts defining admissible A-box configurations

output a text realizing that A-box, in a particular text style. Crucial to this enterprise is the Text Structuring Module that we describe in this section. The Text Structuring Module takes as input a user-specified A-box, and generates a set of specifications of the content of individual sentences (SPLs). These SPLs then serve as input to a language-specific tactical generator that actually generates these sentences. At the basis of the Text Structuring Module (or TSM, hereafter) are the ideas regarding text structure elements, domain model concepts, and text templates that we discussed in the aforegoing sections.

4.1 General Architecture of the TSM

The general architecture of the TSM we conceive of is as follows:

Systemic networks for text structuring. By means of the text structuring networks, an appropriate text structure is formulated in terms of hierarchically related text structure elements, given the A-box.

A program dividing the A-box's content over smaller A-boxes. Given the A-box and the text structure that has been generated, this program divides the A-box into parts that are associated with the text structure elements from which the text structure is composed. In the setting of KPML [1] this can easily be done using so-called ID-inquiries.

A program that generates SPL-code from a given A-box. The aim here is to generate SPLs, using the parts of the A-boxes individuated by the previous program and the realization constraints imposed by text templates.

Once we have individual SPLs, each specifying a sentence, we can use a tactical generator to generate the sentences, which altogether make up the text realizing the content specified in the A-box.

4.2 Systemic Networks for Text Structuring

The major component of the TSM is formed by the systemic networks for text structuring, and therefore we will spend a few more words on describing what these networks look like. In the spirit of [1], we construct a region that defines an additional level of linguistic resources for the level of *genre*. The region enables the composition of text structures in a way that is very similar to the way the lexico-grammar builds up grammatical structures. In fact, by using KPML to implement the means for text structuring, it easily facilitates smooth interaction between global level text generation (strategic generation) and lexico-grammatical expression (tactical generation)[3].

The region, called CADCAM-INSTRUCTIONS, follows out the viewpoint that text templates and text structure elements are essentially orthogonal ideas. Therefore it consists of two parts. One part deals with interpreting the A-box in terms of text structure elements. By traversing the network that the systems of this part make up, we obtain a text structure for the A-box conforms the definitions we provided above. The other part of the region imposes constraints on the realisation of the text structure elements that are being introduced by traversing the other part of the region. Naturally it will depend on our choice of a particular text template (style) which constraints will be imposed. These choices are made through interaction between the user and the system.

References

1. John A. Bateman. Enabling technology for multilingual natural language generation: the KPML environment. *Natural Language Engineering*, 1(1), 1997.
2. Ivana Kruijff-Korbayová. Generation of instructions in a multilingual environment. In Petr Sojka, Václav Matoušek, Karel Pala, and Ivan Kopeček, editors, *Text, Speech and Dialogue (Proceedings TSD'98)*, Brno, Czech Republic, September 1998.

[3] Thus, we overcome the notorious problem known as the *generation gap* in which a text planning module lacks control over the fine-grained distinctions that are available in the grammar. In our case, both text planning and sentence planning are integrated into one and the same system (albeith organized in a stratificational manner).