Comparing PostGraphe with the RAGS architecture

Guy Lapalme
Département d'informatique et de recherche opérationnelle
Université de Montréal, CP 6128, Succ Centre-Ville
Montréal Québec Canada, H3C 3J7
lapalme@iro.umontreal.ca

1 Presentation of PostGraphe

For the past few years, we have studied the automatic generation of graphics from statistical data in the context of the PostGraphe system [4, 3]. PostGraphe is given data in tabular form as might be found in a spreadsheet; also input is a declaration of the types of values in the columns of the table. The user then indicates the intentions to be conveyed in the graphics (e.g. compare two variables or show the evolution of a set of variables) and the system generates a report in LaTeX with the appropriate PostScript graphic files. PostGraphe also generates an accompanying text.

The input to PostGraphe consists of special annotations followed by the raw data. See figure 1 for an example of their use and figure 2 for an example of the output generated. The input annotations indicate the names, the description and the types of the variables, some of which may be of special interest to the user. There is also an indication of how to determine the relational keys for the data and a series of predicates describing the writer's intentions. The justification for these annotations and their Prolog syntax are presented in detail in [3]. This format corresponds to columns in a spreadsheet having as first elements the name of the variable, as second its type, and as third an indication if the variable can be a key or not; the rest of the columns are the data values of the variable.

In order to recast PostGraphe within the RAGS architecture, one should first pay a close attention to the following:

types which associate a set of properties and a unit to every variable of the input. The properties are organized as a multiple inheritance graph divided into a number of sub-graphs, each corresponding to a specific feature: organization (nominal, ordinal, quantitative, ...), domain (enumeration, range, ...), temporal (month, year, ...), format (integer, real, ...), measurements (distance, duration, ...), and specific objects (countries, ...). Units are organized in a parallel inheritance graph.

relational keys which are similar to the notion of the same name in relational databases. They help determine which variables depend on which others. They are also used for ordering variables in some graphics so that the more important ones (usually the keys) are given the more visible positions.

writer's intentions describe what to say and up to a certain point, how to say it. This information is organized in lists that correspond to sections of the report. Intentions are constraints on the expressivity of the chosen text and graphics. PostGraphe tries to find the smallest set of graphics that covers the writer's intentions. The following basic intentions are covered in our model: the *presentation* of a variable, the *comparison* of variables or sets of variables, the *evolution* of a variable along another one, the *correlation* of variables and the *distribution* of a variable over another one. Some of these intentions are further divided into subjective subtypes.

```
data(% names of the variables
     [année,compagnie,profits],              % [years, companies, profits]
     % 2-description of variables
     [année,compagnie,profits],              % [years, companies, profits]
     % 3- types of the variables
     [année/[symbolique],                    % year/[symbolic]
      etiquette,                             % label
      dollar/[pluriel_de(profit)]],          % dollar/[plural_of(profit)]
     % 4-description of types of variables
     [année,compagnie,profits],              % [years, companies, profits]
     % 5- data of special interest to the user (none in this case)
     [ ],
     % 6- candidates for relational keys
     [année,compagnie],                      % [years, company]
     % 7- non-candidates for relational keys
     [profits],                              % [years, companies, profits]
     % 8- the writer's intentions
     [ % section 1
      [presentation(année),                  % presentation(year)
       presentation(compagnie),              % presentation(company)
       presentation(profits)],               % presentation(profits)
      % section 2
      [comparaison([profits],[compagnie]),   % comparison([profits],[year])
       evolution(profits,année)]],           % evolution(profits,year)
     % the raw data
     [[1987,'A',30],
      [1988,'A',35],
      [1989,'A',40],
      [1990,'A',35],
      [1987,'B',160],
      [1988,'B',165],
      [1989,'B',140],
      [1990,'B',155],
      [1987,'C',50],
      [1988,'C',55],
      [1989,'C',60],
      [1990,'C',95]])).
```

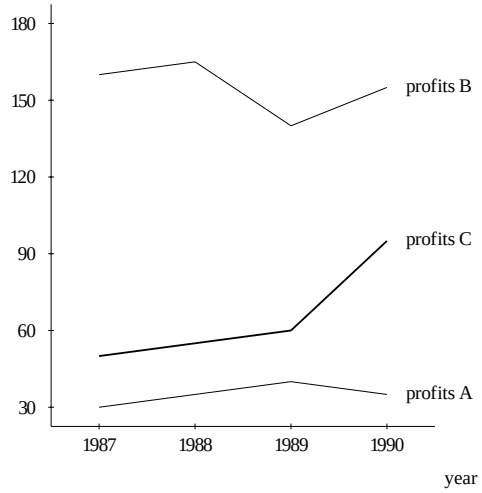
Figure 1: Example of input to PostGraphe

Nouvelle section (3 intentions à traiter).
(*New section: (3 intentions to process)*)
nouvelles intentions: présentation de année. présentation de compagnie. présentation de profits.(*new intentions: presentation of year. presentation of company. presentation of profits*)

year company	1987 profits	1988 profits	1989 profits	1990 profits
A	30	35	40	35
B	160	165	140	155
C	50	55	60	95

[Schéma: *table1*]. présentation de année (100). présentation de compagnie (100). présentation de profits (100).(*presentation of year(100). presentation of company (100). presentation of profits (100).*)

Nouvelle section (2 intentions à traiter).
(*New section: 2 intentions to process*)
nouvelles intentions: comparaison de profits entre compagnie. évolution de profits par rapport à année.(*new intentions: comparison of profits between companies. evolution of profits by year*)



[Schéma: *curve3*]. comparaison de profits entre compagnie (69). évolution de profits par rapport à année (95).
[Schéma: *evolution2*]. évolution de profits par rapport à année (100).

Les profits de la compagnie B se situent à 155 comparativement à 160 en 1987. Les profits de B sont plus élevés que les autres compagnies en 1990. (*Company B's profits are 155 compared to 160 in 1987. B's profits are higher than the ones of other companies in 1990.*)

Figure 2: Report generated by PostGraphe from input of figure 1. English translations given here in (*this form*) were not generated by the system.

comparison([A,B],[X])	columns3	80	[X|Ys]	A $\in$ Ys, B $\in$ Ys
comparison([Y],[X])	bars1	100	[X,Y]	
evolution(A, X)	curve2	100	[X|Ys]	A $\in$ Ys

Table 1: Inference table excerpt

PostGraphe uses a schema-based planning mechanism to generate both text and graphics. The planner uses the types and values of the data as well as the relational keys but it is mainly goal-driven. Because we have decided to use text and graphics for every message, planning is done in 2 phases: the graphical schemas are chosen and then the textual schemas are adapted to the contents and structure of the graphics. This separate planning of text and graphics might be questionned because it is often thought and said in the multimedia generation folklore and in some graphic generation texts that to obtain a good interaction between text and graphics, that text should give informations that the graphics does not show. But Corio [1, 2] observed that most often the text merely reinforces what already appears in the graphic. For example, 29% of texts associated with a comparison intention, there is a mention of the highest value as to say to the reader: "Yes, what you see in this graphic is really what is important".

From this information, our system determines which schemas best satisfy the writer's intentions. It starts from a set of intentions to satisfy; in doing so, no a priori ordering of the variables is given; all types of schemas have been assigned a weight for each intention; we can thus build a global quality function to be maximized. But instead of trying all groups of intentions to find the smallest subgroups of variables that best covers the writer's intentions, we use a set of heuristics.

We first find the intentions that are "compatible" so that each schema takes into account as many intentions as possible while keeping each one "readable". Then we check if each group is feasible and determine the best schema to express it. This step is based on a table which associates the type of a variable with the most efficient graphical methods to express it. The table entries are weighted, and the result of this phase is a list of candidates sorted from the most to the least efficient for the current goals. Table 1 shows a 3-entry excerpt from the table. The first entry indicates that the *columns3* schema (the third schema dealing with column graphs) satisfies the goal of comparing *A* and *B* along *X* with efficiency 80. The variable *X* and the list of variables *Ys* are involved in the process and *A* and *B* must be members of *Ys*.

The next step is the low-level generation of graphic primitives and text. If this stage determines that a figure cannot be generated because of physical reasons, such as being either too large or not having enough grey levels, the next best candidate is tested. This low level work is quite involved because it has to take into account the 2-D constraints and the limitations of the media. For this we developed our own Postscript generation system in Prolog to determine and get access to the exact position of each element (character, line, axis, ...) of a generated graph. These informations are necessary for the references made within the text part of PostGraphe.

Finally, a post-optimization phase eliminates redundancies which can occur because the heuristics sometimes miss a compatible grouping of intentions.

Surface text generation is handled by a subsystem that we call SelTex [1, 2] which captures the subtle nature of the interaction between text and graphics. Before designing SelTex, we did a corpus study of 411 French texts associated with graphics from such diverse sources as "Tendances sociales" published every three months by Statistics Canada, books on statistics, investment funds reports, governmental reports, etc.

Table 2 shows the levels of "RAGS defined" representations that are used in PostGraphe. We see that currently PostGraphe is not so modular as would be desirable but the next section explains why this is the case.

Level	Abstract		Concrete	
	RAGS	PostGraphe	RAGS	PostGraphe
conceptual			4	data, types, intentions
semantic	7	schemas	8	Prolog code
rhetorical	5.2.1	schemas	5.4	Prolog code
document	6	graphic selection rules		LaTeX and postscript
syntax	9	text selection rules		DCG rules

Table 2: Levels of representation used in PostGraphe with the defining section in [5]

2 "Ideal" architecture for a report generator

An ideal architecture of a multimedia report planner could be cast in "RAGS term" as follows using the terminology of figure 1.1 of [5].

- content determination
- medium selection
- sentence planning / graphic planning
- text realization / graphic realization

This way of presenting the steps is too simple because it does not take into account the links that occur between texts and graphics which are of the utmost importance in a report. In long reports, ordering and structure are most important: sequencing of information can be temporal (e.g. by season in a business report), geographical (e.g. unemployment statistics by region) or by content. When less information is given, links between the informations become more important because they alone determine the organization of the report. For example, a short two pages report will deal with only one topic but the layout of the data on the two pages becomes more important because it is the only visible structure.

There are many possible types of links between units of a report, the most obvious being the one which links an explanation with a graphic: the graphic being used for showing an overview and the text to pinpoint some important facts.

So now the problem is to determine where in this ideal pipeline would fit exactly the choice of a type of graphic. One would expect that it should be done between the medium selection and the final realisation. But this choice does not occur at a single place because for selecting the best graphic, one must know the medium, the structure of the report and the realisation constraints. One should know if we are dealing with a graphic alone, a combination of a text and a graphic; some graphics combine themselves better to give a better report organisation and some physical constraints must also be taken into account: starting from how many columns should a bar chart be transformed into a curve? So we see that the modular pipeline architecture must be revisited at least to find a way of informing the different modules of global constraints that must be taken into account. It should be possible to find a way of "backtracking" in the case of a choice that would eventually be not appropriate.

3 "Realistic" architecture for a report generator

In PostGraphe, we devised a set of heuristics to try to guess early what will eventually be good decisions: for example, we associate a size with each graphic (curves are quite compact while tables take more space).

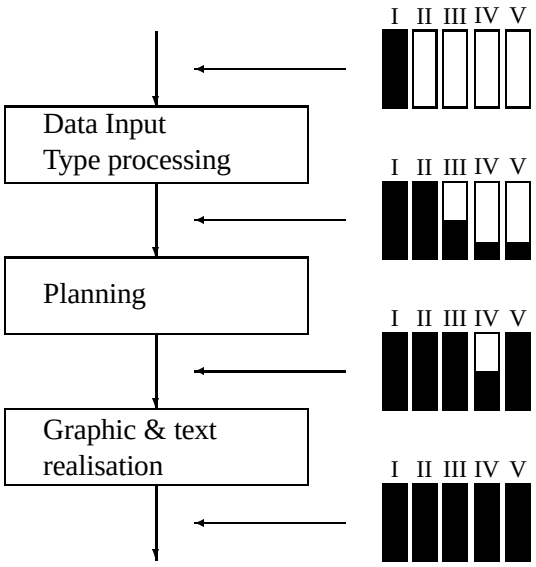


Figure 3: A RAGS view of the PostGraphe system. The labels for the RAGS representation refer to the following: I = conceptual; II = semantic; III = rhetorical; IV = document; V = syntactic

Although, the exact size is only known after the final realisation, it is useful to have an approximation during the planning phase. Unfortunately, the use of heuristics distributes the knowledge of a module and thus can be harder to maintain. For example, if we modify the realization of a type of graphics, then one must check the consequences within the heuristics about it in the other modules. In PostGraphe, we also decided to skip the medium selection module and to always generate both a text and a graphics, but this choice can only be justified in our limited research context where we want to explore the integration of text and graphic. But the content of either text or graphic can depend on the other. For the realizer, we tried using existing graphic rendition systems such as X-Lisp-Stat or the graphics primitives of LaTeX but we realized that many high level choices depend on low-level details such as the number of colors available or the exact positionning of textual labels in the graphics. By implementing our own postscript generator, we were able to integrate it within our decision process, give better heuristics and have a possibility of backtracking in the case of major difficulty.

4 PostGraphe vs the RAGS architecture

PostGraphe implementation is about 4500 lines of Prolog divided into the following five main modules for the graphic generation. In order to measure the relative importance of each module, numbers have been added to give the approximate number of lines in each. Figure 3 give an overall view of the process and the perceived importance of each type of representation defined in RAGS.

Data input (250) reads the input data such as shown in figure 1 and applies some simple transformations to

speed up further processing. The data is quite analogous to the *Concrete Conceptual Representation* of RAGS.

Type processing (150) involves the inheritance tree walking predicates in order to find all the properties associated with each object. This process, which builds on the data of the previous step, does not seem to be addressed by the current definition of RAGS.

Planning (300) is the heart of the system and its implementation can be seen as the processing of a *Concrete Rhetorical Representation*. It is further divided into four parts:

Grouping of intentions is a heuristic that combines intentions in a smaller number of graphics.

Evaluation of the composition of the combined intentions using different schemas in order to find the most appropriate one.

Checking and realisation that the proposed graphics are indeed feasible

Postoptimisation possibly combines the resulting graphics

Schemas (900) are data used by the planning step. This is a form of *Concrete Syntactic Representation*

Graphic rendition system (2000) is quite involved for reasons given above and maps the *Document Abstract Representation* to a *Document Abstract Representation*.

Utilities (1000) are not strictly addressed by RAGS but would certainly be necessary in order to implement the operations on the whiteboard.

The text generation part is essentially a realizer of about 4300 lines of Prolog that implements a set of text selection rules that were obtained from a corpus analysis[2, 1]. This corresponds to the mapping of the *Abstract Syntactic Representation* to the *Concrete Syntactic Representation*.

5 General comments

This exercise of a rational reconstruction using RAGS has been instructive because it forced us to reconsider some points that had more or less taken for granted. But it also raised some important points about RAGS itself:

- how useful is an NLG framework if it does not address the processing steps? for the moment, the description of RAGS is at a very high (often meta) level of abstraction.
- how is backtracking dealt with? on one hand, there seems to be some possibility of having partially specified structures but, on the other side, the whiteboard seems to freeze some choices.
- how should medium selection be addressed ?

But the team of RAGS is to be congratulated for putting out such a work in a domain where even the input is not always well specified.

References

[1] M. Corio. Sélection de l'information pour la génération de texte associé à un graphique statistique. Master's thesis, Département d'informatique et de recherche opérationnelle, Université de Montréal, 1998.

[2] M. Corio and G. Lapalme. Generation of texts for information graphics. In *7th European Workshop on Natural Language Generation, EWNLG'99*, pages 49–58, Toulouse, May 1999.

[3] M. Fasciano. *Génération intégrée de textes et de graphiques statistiques*. PhD thesis, Université de Montréal, 1996.

[4] M. Fasciano and G. Lapalme. Intentions in the coordinated generation of graphics and text from tabular data. *to appear in Knowledge and Information Systems*, page 33p., Sept 1999.

[5] The RAGS project. Towards a reference architecture for natural language generation systems. Technical Report http://www.itri.brighton.ac.uk/projects/rags, ITRI, University of Brighton and Division of Informatics, University of Edinburgh, 1999.

Copies of documents [1] thru [4] can be obtained at the following URL:

`http://www.iro.umontreal.ca/~scriptum/scriptum-english.html`