THE GENERATION OF COORDINATED NATURAL LANGUAGE AND GRAPHICAL EXPLANATIONS IN DESIGN ENVIRONMENTS

A THESIS SUBMITTED TO THE UNIVERSITY OF SALFORD
FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

January 2000

By
Julio Sergio Santana Sepúlveda
Department of Computer & Mathematical Sciences
TIME Research Institute
University of Salford

Contents

List of Tables

List of Figures

"Praise the Lord, O my soul;
all my inmost being, praise his holy name.
Praise the Lord, O my soul,
and forget not all his benefits –"
(Psalm 103:1,2)

… to my wife Elsa
and to my children Samuel and Abigail
with love …

Abstract

One significant shortcoming of most graphical design systems is that they seldom provide coherent explanations about the current design situation and the steps followed in the problem solving process. The central problem in producing explanations in these systems is how to generate explanations that involve both the graphical and the linguistic modalities. This thesis begins by reviewing the current approaches to explanation generation and developing a taxonomy of systems based on their important characteristics. The taxonomy is then used to motivate the need for a new approach to the generation of coordinated graphical and linguistic explanations which is developed in the thesis. Unlike previous work, the new approach is founded on the formalisation of a linguistic and a graphical languages with a clear and explicit semantics, and a translation relation between them. To generate multimodal explanations that include deictic references, a special kind of communicative act denominated *coordinated linguistic and graphical act* (COLGRA) is defined. A COLGRA is an operationalisation of a speech act in which a deictic component is included. COLGRAs are used as the basic building blocks of composite explanations that include quantified expressions and plural noun phrases that other multimodal explanation systems are not able to handle. A prototype system for the geometrical design of towers for electrical lines is developed to demonstrate the viability of the approach and to evaluate its strengths and weaknesses.

DECLARATION

No portion of the work referred to in this thesis has been submitted in support of an application for another degree or qualification of this or any other university or other institution of learning.
Partial results of this thesis have been presented jointly with the supervisors in the following papers:

- *Textual and graphical explanations for geometric problem-solving systems*, Proc. XII Reunión Nacional de Inteligencia Artificial, Sociedad Mexicana de Inteligencia Artificial, Cuernavaca, México, pp 106–112, 1995.

- *An intelligent design system for overhead transmission line towers*, Proc. The 4th World Congress on Expert Systems (WCES98), International Society for Intelligent Systems, Mexican Society for Artificial Intelligence, and Instituto Tecnológico y de Estudios Superiores de Monterrey (ITESM), pp. 270–277, México, D.F., 1998.

Acknowledgements

The development of this thesis was supported by a grant from CONACYT and IIE under the In-House IIE/SALFORD/CONACYT doctoral programme.

Special thanks are due to my supervisor in México, Dr. Luis Alberto Pineda and my supervisor in Salford, Dr. Sunil Vadera.

I am grateful to Professor F.A. Holland and Dr. E. Wilde for their advice and continuous help during the development of this thesis.

I wish to express my thanks to the Instituto de Investigaciones Eléctricas (IIE), for giving me the opportunity and for supporting my participation in the programme, specially to Dr. Pablo Mulás del Pozo, and Dr. David Nieva. I also wish to express my gratitude to the National Autonomous University of México (UNAM) for its support during the last year of this work, specially to Dr. Ismael Herrera Revilla.

I am grateful to Professor James Allen, from The University of Rochester, U.S.A., for his valuable advice in the research proposal.

I also wish to thank the members of the informatics module of the programme for their continuous advice and encouragement, specially to Dr. Luis Enrique Sucar and Dr. Eduardo Morales. Thanks are also due to Dr. Guillermo Rodríguez and my fellow investigators Andrés Rodríguez, Pablo de Buen, and Pablo Ibargüengoytia.

Thanks are also due to Mrs. M.E. Calderon and to the late Fis. Andrés Estebaranz for their untiring effects to make the In-House IIE/Salford CONACYT programme increasingly successful.

Chapter 1

INTRODUCTION TO GRAPHICAL AND LINGUISTIC COMMUNICATION

Around 1950 Alan M. Turing posed the question: *Can machines think?,* and proposed a test that has driven much of the work in artificial intelligence (AI) towards answering that question [Turing 1950]. The Turing test (see Fig. 1.1) consists in determining if an interrogator, **I**, can distinguish which one of two interlocutors, **A** and **B**, is the computer, given only their answers to his questions. If the interrogator is not able to make such a distinction, then one could say that *the computer thinks*. This test along with the ideas behind it, has provoked an ongoing deep philosophical discussion. Thus, avoiding a pronouncement in favour of any party, let the statement in question be rephrased as *the computer thinks in Turing's sense.*

In making the computers think in Turing's sense, the natural language (NL) understanding and generation tasks play a central role. Since a language is

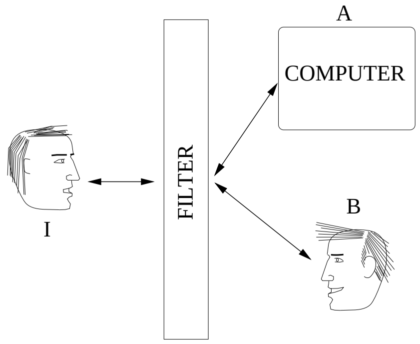


Figure 1.1: The Turing test

the vehicle in which intelligence is conveyed, the more computers' language resemble those of humans, the more they will perform as their human counterpart in Turing's test. Given the capabilities of current systems, however, to make computers understand and speak about any topic seems all but impossible. In view of this, the strategy has been to build systems that deal only with very specific domains [Winograd 1972, McKeown 1982, Buchanan & Feigenbaum 1981, Buchanan & Shortliffe 1984, Chandrasekaran et al. 1989, Maybury 1990, Moore 1989, Lester & Porter 1997]. Many efforts have been devoted to design systems capable of understanding and generating natural language text restricted to these domains. Furthermore, since language is more than natural language speech or text, the advent of multimodal systems has opened new directions in making computers communicate as human beings. Thus, a more complete setting of the Turing test should take multimodality into account. Much attention has been given to developing technology to put together different media at the hardware and basic software levels [Fox 1991]. However, the technology to allow these media to be used as an intelligent communication substratum is still under development and requires a deeper understanding of human-computer interaction [Roth & Hefley 1993, Blattner & Glinert 1996]. One important subject within human-computer interaction is that of the integration of natural language and graphics [Lee & Zeevat 1990, Cohen et al. 1989]. To understand the interaction between human beings and the systems that integrate NL text and graphics, at least two things should be considered, namely: that there are intentions when drawing a picture in the same manner as when uttering a phrase, and that the graphical part is more than an output medium to illustrate the textual part, but it has a semantic force that can be exploited as representative knowledge.

The work presented here extends three aspects of previous developments, namely:

- it enhances multimodal explanation systems with the latest developments in NL explanation generation systems, in particular, by explicitly including intentional structures as a part of their representational structures,
- it enriches NL explanation systems with the addition of the graphical modality, not just as an additional output medium of the system as traditional multimodal systems do, but as a part of the representational knowledge of the system, and
- additionally, the generation of multimodal explanations in design environments (e.g., engineering design), which has received little attention in the past, is addressed in this work.

Although some of these points have been individually dealt with by others, in this thesis a coherent integration of all them is proposed and examined through

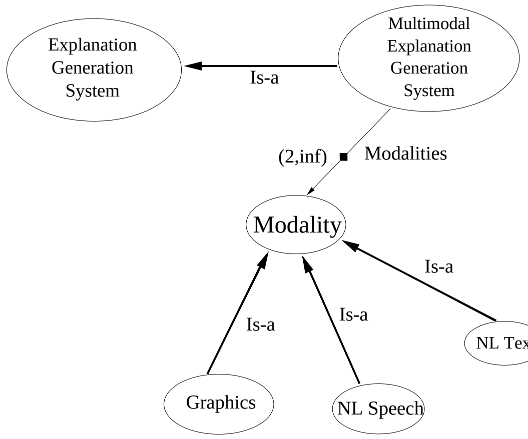


Figure 1.2: Multimodal and explanation generation systems

the exposition of a *prototype* system that has been developed following these lines.

1.1 Explanation Generation Systems and Multimodal Systems

Explanations are elaborated as a response to explicit or implicit questions, such as, *what, why, how, where, when, who,* etc. An *explanation generation system* is one that generates responses of such a kind, typically using NL text, and a *multimodal explanation system* is one that generates such responses using more than one modality (e.g., NL text and graphics). Figure 1.2 shows the relationship between these two categories. The graphical notation used in the figure is based on the one developed by Brachman for his knowledge representation language KL-one [Brachman 1985]. Here, the ovals represent concepts, the bold arrows are used to link a concept with its *superconcept* (*is-a* or *subsumption* link), the arrows with a black box represent *roles* or *relations*, which express relationships between one concept and another concept. In the figure, a multimodal explanation system is defined as a particular case of the general concept of explanation generation systems. The multimodal explanation generation system concept is restricted to have at least two modalities. In the figure, this is indicated by the pair (2, inf), which means that the number of modalities is restricted between 2 and *infinite.* The concept *modality* has many subconcepts. However, here, for the sake of space only three are shown, namely: graphics, NL text, and NL Speech.

Natural language generation systems appeared around the mid-1960s with systems like the primitive system ELIZA [Weizenbaum 1966]. Although it was a very successful experiment, it never pretendend to have any intelligence or reasoning mechanism at all. ELIZA, however, stimulated the imagination of many about having computers able to sustain a conversation with human beings. Probably, the first milestone, not only among natural language but even multimodal systems, was the SHRDLU system [Winograd 1972]. This system was able to interpret natural language expressions and to produce some simple explanations about its actions in a very restricted *blocks* domain. Despite the very limited geometrical blocks world, SHRDLU opened important questions that defined many new directions among natural language processing and multimodal systems. In particular, since then, many wondered if the kind of interactivity and graphics-NL integration exhibited by this system was possible in systems with more complex domains. Although this system was not specifically a *multimodal explanation generation system,* it incorporated many of the features these systems are expected to display. Here are some of its characteristics that started to be included in other systems:

- The system understood NL expressions of different types: questions, statements, and commands.
- It displayed graphical entities (on a separate TV monitor). Its graphical world consisted of geometrical objects (pyramids, boxes, and rectangular prisms) of different colours and sizes.
- It manipulated the objects in its world according to the instructions of the user.
- It could explain the situation depicted in its world.
- It resolved referring expressions of different kinds through a dialogue with the user. Consider, for instance, the following dialogue fragment in relation to Fig. 1.3:

  User: grasp the block.
  System: I don't understand which block do you mean.
  User: find a block which is taller than the one you are holding and put it into the box.
  System: by "it" I assume you mean the block which is taller than the one I am holding.
  User: OK.
  There are several blocks in the system's world so the definite clause *the block* is ambiguous. The reference is solved through the user provided explanation. This explanation, however, contains further references that need to be solved: the pronouns *one* and *it*. Since the system is initially holding the red block, (a) in Fig. 1.3, it identifies the blue block with *the block* the user is intending to refer to, and moves it into the box, (b).

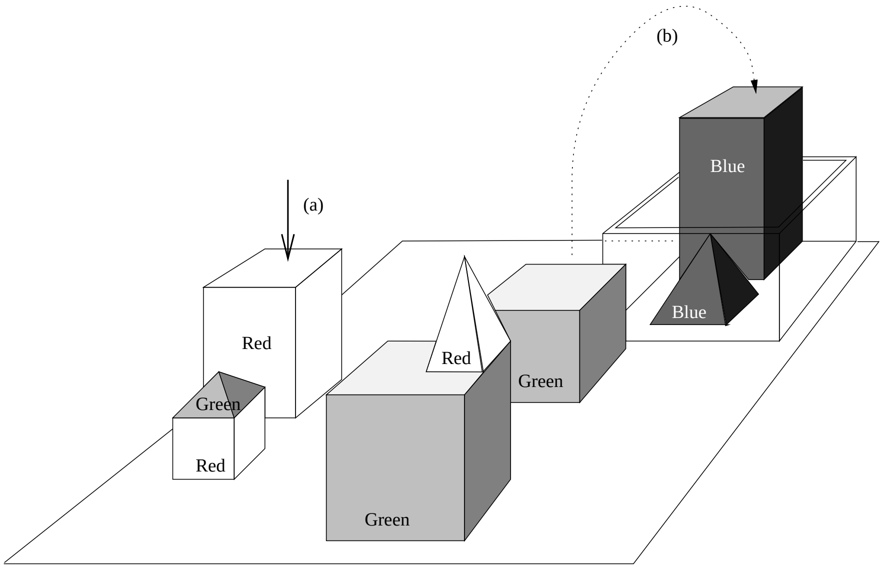


Figure 1.3: A typical arrangement of pieces in the SHRDLU system

- It was able to explain the ambiguity of a user given expression by generating an embedded multiple-option list in the explanatory phrase. See, for instance, the following explanation:
  "I'm not sure what you mean by *on top of* in the phrase *on top of green cubes.* Do you mean
  1 - directly on the surface
  2 - anywhere on top of?"
- In addition to the above capabilities, SHRDLU was also able to learn new words defined in terms of the words the system already knew, to answer yes-no questions as well as different wh-questions, such as *what, how many, why, when,* etc., to understand quantifiers such as *every, any, some,* etc., and to reason about temporal relations.

SHRDLU seminally synthesised in a single system many of the desirable features of multimodal dialogue systems. From then on, NL systems, multimodal systems, and explanation generation systems evolved in different directions. In the next section, the question of these desirable features is explored in more detail.

1.2 A Desirable Multimodal Explanation System

What is the ideal of a multimodal explanation system? Whatever the answer to this question is, it is at the risk of getting dyed with one's preferences and conceptions. Assuming that risk, however, a set of *desirable* features for this category of systems is proposed with the aid of the dialogue presented below. This dialogue was taken, adapted and translated from a corpus of real dialogues in Spanish compiled by Olmos-Sánchez (1998). These dialogues, whose initial design goals were empirically defined, were captured in videotape from two real interlocutors, and transcribed into Spanish text and graphics as an experimental support for the work presented in this thesis[1].

TEXT	GRAPHICS

[1.1]: Instructor You are going to design a drawing according to the following: The drawing must be composed of: • one square, • one rectangle, • one circle, and • three lines.	

[1]The dialogue is inserted here exactly at the expense of some redundancy.

[1.2]: Instructor They are arranged in the following way:

- the rectangle is inside the square,
- the circle is inside the square,
- the rectangle and the circle do not intersect each other,
- one line is parallel to one of the edges of the square,
- the two other lines are perpendicular to the *parallel* line,
- one extreme of one of the *perpendicular* lines is joined to the circle,
- one extreme of the other *perpendicular* line is joined to the rectangle,
- the other two extremes of both *perpendicular* lines are joined to the *parallel* line.

[1.3]: Designer then ... are they a square, a rectangle, a circle, and three lines?

[1.4]: Instructor yes

[1.5]: Designer is the rectangle inside the square?

[1.6]: Instructor yes

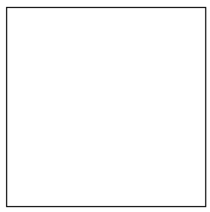
[1.7]: Designer how about the circle ... ?

[1.8]: Instructor yes, it is inside as well

[1.9]: Designer the rectangle and the circle do not intersect each other, do they?

[1.10]: Instructor no, they shouldn't intersect each other

[1.11]: Designer good *(• draws the square)*



[1.12]: Designer next, I've got three lines

[1.13]: Instructor that's right, you've got three lines

[1.14]: Designer two of them are perpendicular ...

[1.15]: Instructor ... to one parallel

[1.16]: Designer to one parallel. Well, hum –

[1.17]: Instructor one parallel to one edge of the square

[1.18]: Designer right

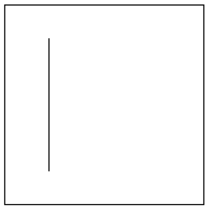
[1.19]: Designer and the constraint of the two perpendicular lines is that one of them touches ...

[1.20]: Instructor ... the circle

[1.21]: Designer yep, the circle

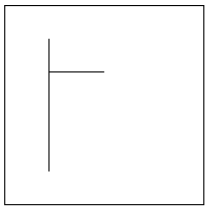
[1.22]: Instructor the other touches the rectangle

[1.23]: Designer OK *(• draws a line inside the square)*

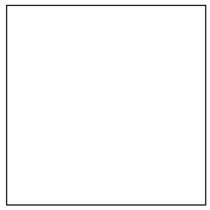


[1.24]: Designer they are two perpendicular lines *(• adds a line perpendicular to the other)*

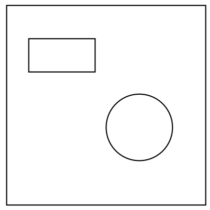
[1.25]: Instructor but the parallel goes outside the square



[1.26]: Designer *(• erases both lines)*



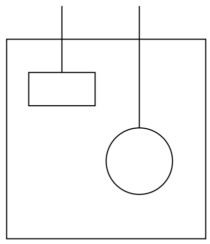
[1.27]: Designer *(• draws the rectangle and the circle)*



[1.28]: Designer then, one of the perpendicular lines touches the circle and the other touches the rectangle, don't they?

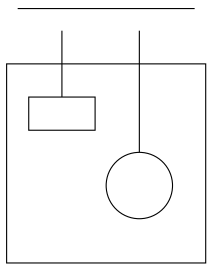
[1.29]: Instructor yep

[1.30]: Designer *(• draws the two lines)*



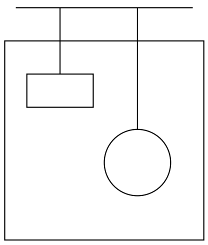
[1.31]: Designer *(• draws the line on the top)*

[1.32]: Instructor they also touch the parallel



[1.33]: Designer (• *joins the two lines to the line on the top*)

[1.34]: Instructor right!



This simple dialogue reflects the interaction between a designer and an instructor. Interactions of this sort are typical of the ones that occur in design environments, say, between an architect and his or her client for instance. To see this, suppose that the square represents a room, and the circle and the rectangle represent some items of the room's furniture, and that the dialogue participants are trying to agree about the relative location of all these objects. Although this dialogue is somehow similar to a tutoring session, there is a crucial difference that should be noted. In the latter case there is a perfectly well defined curriculum, while in the former, the *curriculum* is dynamic, so to speak, since the *instruction matter* is created as the dialogue progresses. There is also a difference between the purpose of a design session and a tutoring session. In the latter, the instructor goal is for the apprentice to understand a subject, whereas in the former, it is for the designer to create an object guided by some set of specifications and some set of constraints [Brown & Birmingham 1997].

Without pretending to be general, it can be said that the previous dialogue reveals many features that are desirable in explanation generation systems with multimodal capabilities. Here, some of these features are expounded:

1. The system must be able to play any of the interlocutor roles. In this particular case, it must be able to play either the role of instructor or designer. This implies that both the system and the user must be given the ability to manipulate the graphical part. Note, for instance, that the SHRDLU system manipulated the graphical part but it did not allow the user to do it directly.

2. The objects and the constraints that constitute the dialogue matter can be predefined in a static knowledge base or they can be dynamically introduced during the session. Observe, for instance, that in dialogue portions **[1.1]** and **[1.2]** the linguistic terms corresponding to the design objects and its constraints are introduced at the beginning of the dialogue. The graphical representations of these objects, however, are introduced later in the dialogue.

3. The system must be able to accept input from different modalities. In the example dialogue, the graphical input is mentioned in the textual part with italics enclosed in parentheses, in portions **[1.11], [1.23], [1.24], [1.26], [1.27], [1.30], [1.32],** and **[1.33]**, and its depictions are shown in the *graphics* column.

4. The system must be able to generate definite noun phrases to identify the objects in the discourse. For example, in dialogue portion **[1.2]**, there are phrases such as *the rectangle, the parallel line, one of the perpendicular lines,* and *the other perpendicular line,* to identify the different geometrical design objects.

5. As soon as the graphical objects are introduced, there must be the possibility to put them in correspondence with the linguistic terms of the previous discourse. This establishes a model against which the interlocutors' assertions and actions are tested. In portion **[1.11]**, the designer draws a square and the instructor assumes that it is the square they were talking about in the previous phrases. Again, in portion **[1.23]**, the designer draws a line that the instructor assumes is one of the three straight lines, without identifying it with any of them in particular. When the next line is added, in portion **[1.24]**, the instructor detects that something is going wrong because, no matter which one of the two lines is assigned to *the parallel line* definite clause, one of them should go outside the square[2]. Thus, in portion **[1.25]**, the instructor utters the phrase *but the parallel goes outside the square*, intending that the designer corrects his mistake. This portion also illustrates that design tasks have to deal many times with incomplete or underspecified problems. In this case, that the parallel line must lie outside the square is a constraint that was not mentioned in the original definition of the problem. The incorporation of this knew knowledge to the problem, however, did not alter the ability of the designer to solve it.

6. The objects and even the constraints can be removed in the same way they were added. In portion **[1.26]**, the designer erases two lines when he learns they do not conform to the intended design.

7. A structure of the underlying intentions must be maintained. It should be noted that the whole dialogue is driven by the intentions of both the instructor and the designer. There are intentions behind the NL text utterances as well as behind the drawing of a graphic depiction. For instance, in portions **[1.1]** and **[1.2]**, the intention of the instructor is that the designer recognizes the specifications and constraints that will serve as a guide to the design. Also, when the square is drawn in dialogue portion **[1.11]**, the designer has the implicit intention that the instructor identifies it with *the*

square of the previous discourse. In portion **[1.23]**, the designer intends that the line is identified as one of the three lines mentioned at the beginning of the dialogue. In portion **[1.27]**, the designer intends that the figures drawn are identified with the rectangle and the circle of the previous discourse and that their actual location complies with the constraints stated for them. In portion **[1.32]**, when the instructor utters the phrase *they also touch the parallel*, he has the intention that the designer joins the two lines with the so called parallel line.

8. The possibility of accepting or making *deictic references*[3]to graphical objects is a very useful feature that could help to disambiguate NL or graphical expressions and to simplify the overall dialogue. In the example dialogue this sort of reference is not used, but it is evident that some of the interlocutors' intentions could be stated more clearly using this technique. For instance, between dialogue portions **[1.11]** and **[1.13]**, the instructor easily assumes that the designer intended him to recognise the figure just drawn as the square because of its shape. No deictic reference was used here, and there was no need to use it. What happens between portions **[1.23]** and **[1.26]** is different. In this case, when the first line is introduced in portion **[1.23]**, a question arises: which one of the three mentioned lines is the designer intending to refer to by his drawing? Here, a utterance like *"this is the parallel line"* along with a pointing gesture towards the line, could have helped the instructor to disambiguate the message conveyed by the figure.

9. The system must be able to answer yes-no questions as well as different wh-questions. Between dialogue portions **[1.3]** and **[1.10]**, the designer poses several yes-no questions which are immediately answered by the instructor. There are no examples of other types of question in this example, however some of the dialogue utterances could be reformulated as wh-questions. For instance, the expression in portion **[1.14]** is implicitly understood by the instructor as the question: *"What are two of the lines perpendicular to?"*.

10. The system must resolve quantifying expressions. There is no such situation requiring the solution of a quantified expression in the above example dialogue. However, solving quantifying expressions is one of the most common activities in design. Consider, for instance, the following specification in relation to the Fig. 1.4:
    Any dot in the design must be within a distance of 50 units measured from the black ball.
    To satisfy this specification, the system must be capable of evaluating a quantifying expression like:

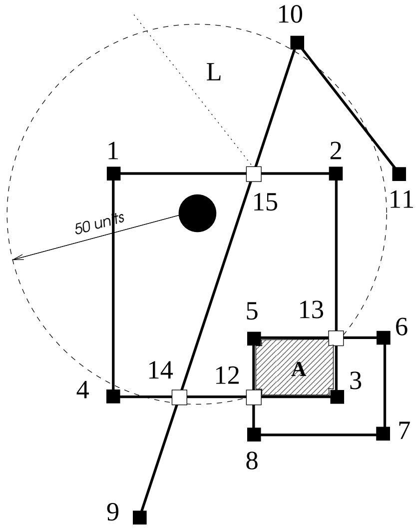


Figure 1.4: A quantification example

$\forall x, \text{dot}(x) \rightarrow \text{distance}(ball,\ x) \leq 50$

In the example shown in Fig. 1.4, the distance from the black ball to every particular dot, symbolised here by black or white boxes, must be evaluated to test if it is less than or equal to fifty units. This is the same as testing if all the dots are inside the circle with a radius of 50 units and with the centre in the black ball.

11. The system must be able to exploit the semantic contributions of all the modalities involved. This is particularly important in the case of the graphics modality. In its discrete domain, judged from its external behaviour, SHRDLU performed very well: it could extract and present information about the relative situation of the partaking geometrical pieces. For instance, when the system was asked, in the situation depicted in Fig. 1.3, *"is there a large block behind a pyramid?"*, it responded: *"Yes, three of them: a large red one, a large green cube and the blue one"* [Winograd 1973]. However, a deeper look at the system reveals that the graphical part of it is used only as an *illustrative* vehicle for those facts in the knowledge base that are graphically depictable. Consequently, the information presented to the user in the visual depiction is richer than that retrievable from the knowledge base: while the actual domain the system operates on is discrete, the visual depiction occurs in a continuous space. In Fig. 1.3, for instance, it is impossible for the system to say that the box's volume is greater than the blue pyramid's, though it is evident from the graphical visualization. Using a discrete domain again, this same behaviour is exhibited with a more formal approach in the Tarski's World System [Barwise & Etchemendy 1989], though using the language of first-order logic instead of the NL component,

and without the problem-solving capability of SHRDLU.

The example dialogue shows a situation that does not occur in systems that do not have all pictorial information in their knowledge base. When a rectangle and a circle are drawn by the designer in the dialogue portion **[1.27]**, if it is assumed that the instructor is understanding the message conveyed by the drawing, he must be aware not only that the figures drawn correspond to the linguistic terms *the rectangle* and *the circle*, but that these objects comply with the constraint specified for them, namely: that they must both lie inside the square. This is possible only if he is able to carry out a semantic evaluation of the geometrical information implicit in the graphical depictions. If the system is to play the role of the instructor, this implies that, in order to perform such a semantic evaluation, it must have access to a *graphical language* with its formal syntax and semantics, and to the graphical language sentences that represent the particular situation.

1.3 A Scheme for a Multimodal Explanation System Oriented Towards a Design Task

In view of the desirable characteristics of a multimodal explanation system (MMES), one such system for the generation of explanations in the context of geometrical design problem-solving tasks is sketched here. First, the question of the selection of design tasks as a domain for the MMES is addressed and, then, the architecture of the system is briefly reviewed.

1.3.1 Why a design task?

Most explanation systems have been developed as output interfaces to expert systems based on static knowledge bases (e.g, diagnostic expert systems). Their basic architecture can be summarised as depicted in Fig. 1.5. In these systems the interaction cycle is such that the whole explanation is generated as a response to a user query or to an expert system communicative goal, and then the user can pose a new query or give feedback to the system. As shown in the example dialogue of Section 1.2, a dialogue oriented towards a design task requires an interaction that is more fine grained.

[2]This is a clear example of how the problem-solving process is strongly attached to the explanation generation process. Here, if the system plays the role of the instructor, it cannot determine whether the introduced graphical objects are part of the problem solution until it has assigned names to them.

[3]For the purposes of this thesis, an utterance will be assumed to be a *deictic reference* when, to identify the object it refers to, it is accompanied by a *deictic act*, i.e., by the pointing of the finger, a nod of the head or some other demonstrating gesture towards such an object.

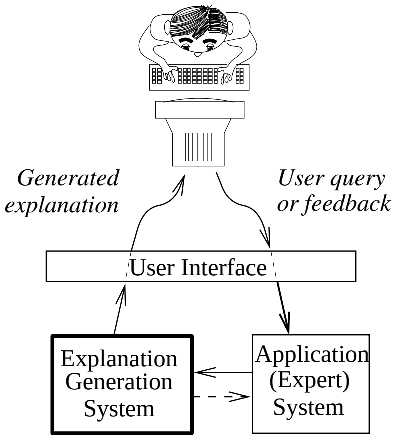


Figure 1.5: Typical architecture of the expert-system/explanation-system combination

By distinguishing different types of *building blocks* as the basis for the construction of different knowledge based systems, and by exposing how these blocks are attached to explanations of a specific kind, B. Chandrasekaran and his colleagues have shown that the problems posed to the explanation generation process by design tasks contrast greatly with those that sprout from systems whose groundwork are static knowledge bases [Chandrasekaran 1986, Chandrasekaran et al. 1989, Brown & Chandrasekaran 1986]. Consider, for instance, typical multimodal explanation generation systems. Their architecture follows the same pattern presented in Fig. 1.5. In this case, the user poses a query which is translated into a language that an application or expert system understands. The application system, then, elaborates on the user query to produce an answer which will constitute the contents of the presentation. This answer, typically, a plan to do or to describe something, is in turn passed to a multimodal presentation system. This system works out the answer provided, to build up a multimodal presentation. Two problem-solving processes might take place in this architecture: the first is dependent on the application and takes place in the application or expert system; the second has to do with the multimodal presentation design and it is the main task of the multimodal presentation system. This process has to solve problems like: what information should be communicated in which modality, when some piece of information should be exhibited, how information presented in one modality should refer to the information presented in the others, etc. This independence of tasks is possible precisely because the underlying knowledge base is static: no new information is added to it or deleted during the system operation. Design systems, however, require to provide the user with the possibility of introducing new objects and constraints to the knowledge base at any time. This would correspond with enabling a user to update the knowledge base of a natural language based question-answering system, which is a characteristic that not many of the current explanation systems allow.

Consider, for instance, a design system capable of maintaining a dialogue like the one presented in Section 1.2. In this case, in dialogue portions **[1.24]** and **[1.25]**, the problem of relating a referring expression, the definite clause *"the parallel line"*, to a graphical object is strongly attached to the design problem-solving process itself. That is, the course of the problem solution depends upon which graphical object is assigned a particular name! This problem has been named *the problem of multimodal reference resolution* by Pineda & Garza (1997). In their work they argued that its solution is one central issue of multimodal inferences. In a design dialogue, the output of this resolution is crucial for the course that the dialogue and the problem solutions will take in subsequent steps.

Furthermore, Pineda (1993) has shown that design is a highly creative process in which the design objects appear, disappear, emerge, and are modified as its primary activity. This, in turn, reveals that the keyword in design is *create* as opposed to *discover* or *explore* which better describe other AI systems[4] [Simon 1981, Brown & Birmingham 1997, Brown & Chandrasekaran 1986, Iwasaki 1997, Goel 1997]. As a consequence, different challenges are faced when generating explanations for intelligent design systems than those for other AI systems. For instance, the question of how the newly created objects can be unambiguously referred to by the system is central in design while it is less important in other disciplines.

The example dialogue presented in Section 1.2 makes it evident that the graphical modality plays a more active role in design dialogues than in other multimodal presentation discourses. To have a system that is able to design or to aid in design activities and at the same time be able to explain its reasoning processes and situations, is a challenge that involves the development of one language for each modality and their *coordination*.

The architecture overviewed below is specifically designed to deal with some of the problems posed by design processes in the generation of multimodal explanations.

---

[4]Notice that the word *design* is used here in a broader sense than that of engineering design. As Iwasaki (1997) has said:

> "In English, the word *design* applies to engineering as well as artistic creative endeavors. One can say 'to design' machines and buildings, as well as dresses, drugs, policies, and cancer treatment protocols."

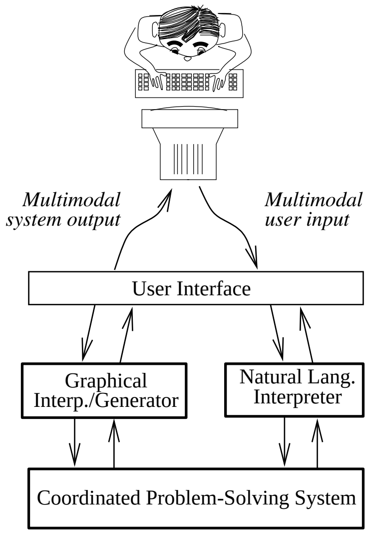


Figure 1.6: Architecture overview

1.3.2 Architecture overview

In the previous section it was argued that multimodal design systems need to (i) have fine grained interaction, (ii) reckon with a good mechanism to resolve multimodal references, (iii) manage the creation of new objects during the dialogue process, (iv) exploit the semantics of the graphical part, and (v) coordinate the NL text and graphic components. These requirements lead to an outline architecture shown in Fig. 1.6. In contrast to the architecture given in Fig. 1.5, the main features of this architecture are as follows:

1. Each modality is assigned a formal language with its own syntax and semantics.

2. In the same manner that the system can elaborate a multimodal output, it can accept a multimodal message as input. The graphical input is of particular importance in design systems, since the design problem to be solved could be directly expressed as a graphic depiction.

3. Since this system is aimed at design tasks, every change in the target model or new state must be immediately fed back to the user, though the problem-solving process has not finished yet.

4. The problem-solving process takes advantage of the semantic contributions of each modality.

5. Since, in the work presented in this thesis, the multimodal presentation is about the problem-solving process itself, the *multimodal presentation*

system has to look inside the problem-solving module. Consequently, the multimodal presentation generation and the problem-solving process are intermixed activities that are managed by the *coordinated problem-solving system*.

In this model, the user poses a problem in the form of a multimodal input (graphics and NL text). The user interface passes on the information to each modality interpreter to be interpreted accordingly. A *coordinated* message, containing both the graphical and the linguistic information is built up and passed to the coordinated problem solving system. This system works out the posed problem using the semantic contributions of both modalities, and issues a coordinated message composed, again, of graphical and linguistic information. The fine grained interaction of the system allows a collaborative design dialogue in which the user can divert from one solution proposed by the system to another one.

1.4 A Guide to the Following Chapters

A guide to the subsequent chapters is presented here.

In Chapter 2 a taxonomy of multimodal and explanation systems is developed to present the evolution of these systems. There, the work developed in this thesis is also located within the map of that characterisation.

Chapter 3 presents the basic representational structures used as a basis for the work presented in this thesis, namely: the *coordinated linguistic and graphical semantic unit* (COLGSU). The two formal languages, a linguistic and a graphical, are introduced in this chapter along with a translation relation between them. A state of interpretation is defined in relation to the introduced structure, the COLGSU. Finally, an analysis of the quantified expressions within the context of the developed languages is presented with a special emphasis on the determination of the quantification domain.

The next chapter presents an operationalisation of the speech acts which are rendered as specific sentences. These acts constitute the surface rendering of the text and the graphics of the explanations.

Chapter 5 develops an operationalisation of the rhetorical relations that will produce the structure of the discourse of the explanation. Examples of explanations of the design of electric towers are presented in this chapter.

Chapter 6 presents the prototype system developed under the principles established in this thesis and its architecture.

Finally, in Chapter 7 several conclusions are drawn, and some directions for future work are defined.

Chapter 2

A CONCEPTUAL SURVEY OF NATURAL LANGUAGE AND MULTIMODAL EXPLANATION SYSTEMS

The evolution of multimodal[1] and explanation systems can be examined from different perspectives. One of these perspectives is to trace the chronological events that led to the *more evolved systems* up to the present days. The perspective adopted in this work, however, is more taxonomic: the different systems are examined in the context of a classification elaborated in terms of their salient features independently of the time when they were developed. This does not mean that the chronological view is not important. In fact, there is an interrelationship between the events as they occurred in time and the introduction of features that have led to new system classes.

The different classes of explanation and multimodal systems emerged as responses to questions of particular types, e.g., *How was a problem solved?, How were determinate conclusions reached?, How was an artifact designed?, What is a certain object?, How is a particular object composed?, What happens during a determinate process?, Why was a problem solved in a certain way?, Why is a particular operation required?,* etc. The possibility of answering each kind of question depends not only upon the information included or omitted in the system, but upon the organization of that information as well [Chandrasekaran & Swartout 1991]. These are the ingredients that make up a class in the taxonomy presented here.

The taxonomy developed aims to provide a comprehensive summary of work in the area and an indication of the main research contributions of the existing

[1]Note that when speaking of *multimodal* systems in this thesis, the term refers specifically to multimodal explanation systems.

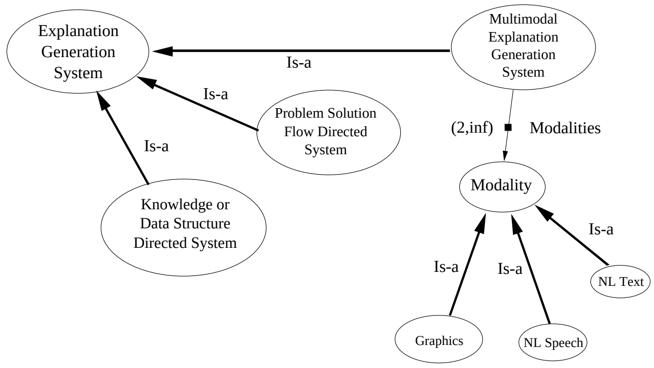


Figure 2.1: Code and data traversing systems as explanation systems

systems. To facilitate this, a Brachman style semantic network is used[2] [Brachman 1985] to present the taxonomy. A prototypical example from each class of systems is described and then added to the semantic network. Once all the classes of systems are described, the semantic network represents the current state of the field of natural language explanation and multimodal explanation systems. The system and research described in this thesis can then be placed in the context of other work by creating a new node in the semantic network. Sections 2.1 to 2.5 describe the various systems and section 2.6 places the system developed in this thesis in the context of other work.

2.1 The First Attempts: Explaining the Problem Solution Trace

Possibly the simplest explanations produced by a computer are those that trace over the steps in the solution process to a given problem. Figure 2.1 gives the initial taxonomy of systems that includes this simple class of systems which is labelled *problem solution flow directed systems*, because the explanation generation is chiefly guided by following the flow in the solution process.

Systems like the Mission Planning Assistant (MPA)[3] [Tanner & Keuneke 1991] achieve this sort of explanation by utilising textual templates with variables that can be instantiated with the data of the particular case.

Figure 2.2 depicts the format of a planning operator showing, among others[4], the slots that will serve as the embedded textual templates for explanation

[2]Some of the elements of this graphical notation were introduced in section 1.1 (p. 3 of this document).

[3]The Mission Planning Assistant (MPA) was a prototype system that designed a strategic plan for a particular kind of Air Force mission.

[4]One such operator was selected in the course of a planning session according to a decision

```
(Plan
    (Name <plan name>)
    (Sponsor <sponsor name>)
    (Purpose <purpose string>)
    (Achieved <achieved string>)
    (Body
        <step 1>
        . . .
        <step i>
        . . .
        <step n> ))
```

Figure 2.2: Format for a plan operator of the MPA system

generation: <*purpose string*>, and <*achieved sting*>. The explanation template, shown in Fig. 2.3, uses the information written down in these slots to build up an explanation.

For example, taking the actual planning operator that considers the feasibility of an A-10 plane for a certain mission (Fig. 2.4) as the basis for a user required explanation when the planner is at the *ChooseSquadron* step, the application of the template (Fig. 2.3) would yield the explanation shown in Fig. 2.5.

Although the MYCIN system [Shortliffe 1976, Davis et al. 1977, Scott et al. 1984] used several strategies to build up its explanations, it followed a strategy similar to the one presented above when answering questions about how it reached a particular conclusion.

Another more elaborated approach to this sort of explanation is to directly translate the code using an interpreter that transforms it into an English-like form. The Digitalis Therapy Advisor (DTA) [Gorry et al. 1978, Swartout 1977] was one of the first systems that used this approach. The DTA procedures, written in a special language, were translated into an English-like form when the user asked how a task was accomplished. Using a similar technique, the MYCIN rules [Scott et al. 1984] were translated into English as shown in Fig. 2.6. Now, generally, when a user asked MYCIN a question, he or she was not just interested in knowing what a particular rule said, but in knowing how it solved a problem. To enable it to produce such "how" explanations, MYCIN maintained a

---

table computed by the operator's *sponsor* with the available data at runtime [Chandrasekaran et al. 1989].

In the context of <*context of containing specialist*>, I:

- <*achieved string from step 1*>
- ...
- <*achieved string from step* $i - 1$>

I was in the process of:

- <*purpose string from step i*>

and was about to do the following:

- <*purpose string from step* $i + 1$>
- ...
- <*purpose string from step n*>

Figure 2.3: Plan explanation template from the MPA system

```
(Plan
    (Name A-10)
    (Sponsor A-10-Sponsor)
    (Purpose "considering the feasibility of an A-10 for the mission")
    (Achieved "chose an A-10 for the mission")
    (Body
        AssignAircraftType
        ChooseSquadron
        AssignBase
        GetRange
        (Design Configure-A-10) ))
```

Figure 2.4: MPA's A-10 planning operator

In the context of selecting an appropriate aircraft for the mission, I:

- Assigned A-10 as the aircraft type.

I was in the process of:

- Selecting an squadron for the mission.

and was about to do the following:

- Select a base for the mission.
- Determine the range for the mission.
- Choose a configuration for the A-10 on this mission.

Figure 2.5: Explanation for a MPA's plan

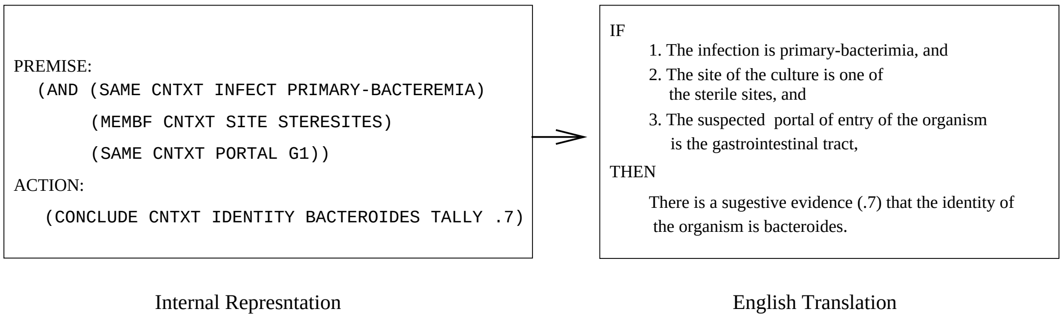


Figure 2.6: Translation of MYCIN rules to English

history tree. This tree was traversed to collect the rules applied in the solution of a particular problem. Then, the relevant parts of these rules were translated, with a mechanism like the one used in Fig. 2.6, and, along with other information, their translation constituted the desired explanation.

A singular case in this category is the Graflog system [Pineda 1989, Pineda 1992*b*, Pineda 1993]. This system, aimed at a 2D geometrical design problem-solving task, produced an explanation following the steps in the problem solution path, but, instead of using any of the natural language devices examined so far, it uses a sequence of graphical depictions that act as snapshots of these steps. The process of translating from the graphical language, which is part of the knowledge representation groundwork of the Graflog system, into the graphical depictions is a more complex process than that of translating from the programming code into an English-like explanation of the systems described above.

For a trained user, this graphical explanation could say more than a natural language textual explanation. The sequence of illustrations produced by this system, however, was more difficult to understand as an explanation by a novice user.

The systems under this category were able to answer implicit or explicit questions about *how a problem was solved,* or *what steps were followed in solving a problem*. Some of these systems could even tell *what different options were available to it at a specific decision point*. However, because of their lack of information on the reasons behind the decisions taken during the problem solving process, they could not answer *why the system decided to follow one option and not another*.

2.2 Traversing the Knowledge Base

Generating explanations about the contents of a knowledge base can be a task that is primarily driven by traversing the graph implicit in the knowledge base. For this reason, in Fig. 2.1, the systems that follow this strategy have been grouped under the category named *knowledge or data structure directed system*.

Again, to cope with the problem of extracting a view of a knowledge or data structure and presenting it in the more readable form of NL, the easiest way was to use *templates*. For instance, the legendary SHRDLU system used templates[5] when answering questions about the situation of its blocks world [Winograd 1972, Winograd 1973]. Having anticipated the sort of questions its users wanted to ask, this system counted on a set of predesigned templates to inspect its knowledge base and present a NL explanation. Resting upon simpler structures than those of knowledge bases, some relational database management systems with NL interfaces, provided, among their adornments, NL presentations of their data views using the same sort of templates [Codd et al. 1978, Waltz 1978].

An important evolutionary step in this class of systems was the identification of specific patterns when describing the structure of objects or processes and the envisaging of the possibility of applying these patterns in conjunction with the graph traversal of the knowledge structure describing these objects or processes. A salient example of this type of work is that taken by Paris (1988). She developed the strategy shown below to follow a chain of causal links[6]:

Given a chain of causal links, for each object:

[5] The templates used by SHRDLU and other contemporary systems to present information about their knowledge or data structures were similar to those illustrated in section 2.1 on p. 20 of this thesis.

[6] In this description, the items enclosed in *curly brackets*, "{ }", are optional.

1. Follow the next causal link
2. {Mention an important side link}
3. {Give attributive information about a part just introduced}
4. {Follow the substeps if there are any
   (These substeps can be omitted for brevity.)}

Go back to 1.

This process can be repeated for each subpart of the object.

Lester & Porter (1997) presented a more elaborate strategy for explaining the contents of a knowledge base that extends the work of McKeown (1985), and Paris (1988). Since this work incorporates a rhetorical structure, it is reviewed below in Section 2.3.

The systems under this category were able to answer implicit or explicit questions about *how an object was composed or structurally defined*, or *how an object was functionally defined*, or *what the steps in a process were (as described in a knowledge-base)*, or *what happened during a process*. Since these systems basically explained a view of a knowledge-base, they could not go beyond it. Their explanations are basically definitional.

2.3 Imposing a Discourse Structure to Explanations

Many philosophers and linguists have observed that human discourses follow specific patterns that depend upon the discourses' purposes [Aristotle 1954, Williams 1893, Shepherd 1926, Grimes 1975]. There are a number of systems whose explanations are generated by using *rhetorical* patterns. Figure 2.7 shows the revised taxonomy when a node *rhetorical structure directed system* is added to represent this class of systems.

Although the observation about the patterns that occur in natural language discourses can be traced back to Aristotle, one of the first works that synthesised a set of *rhetorical predicates and schemas* specifically directed to automatic discourse generation was that of McKeown (1982). She elaborated on the work of Shepherd (1926) and Grimes (1975), and developed a computational model of her *schemas,* that embody standard patterns of discourse structure. These structures were the core of the TEXT system that she developed to produce NL explanations in response to questions about a database structure. Figure 2.8 shows the identification schema with an example of the text produced by its application. In this figure, the *slash*, "/", separates alternatives, the items enclosed

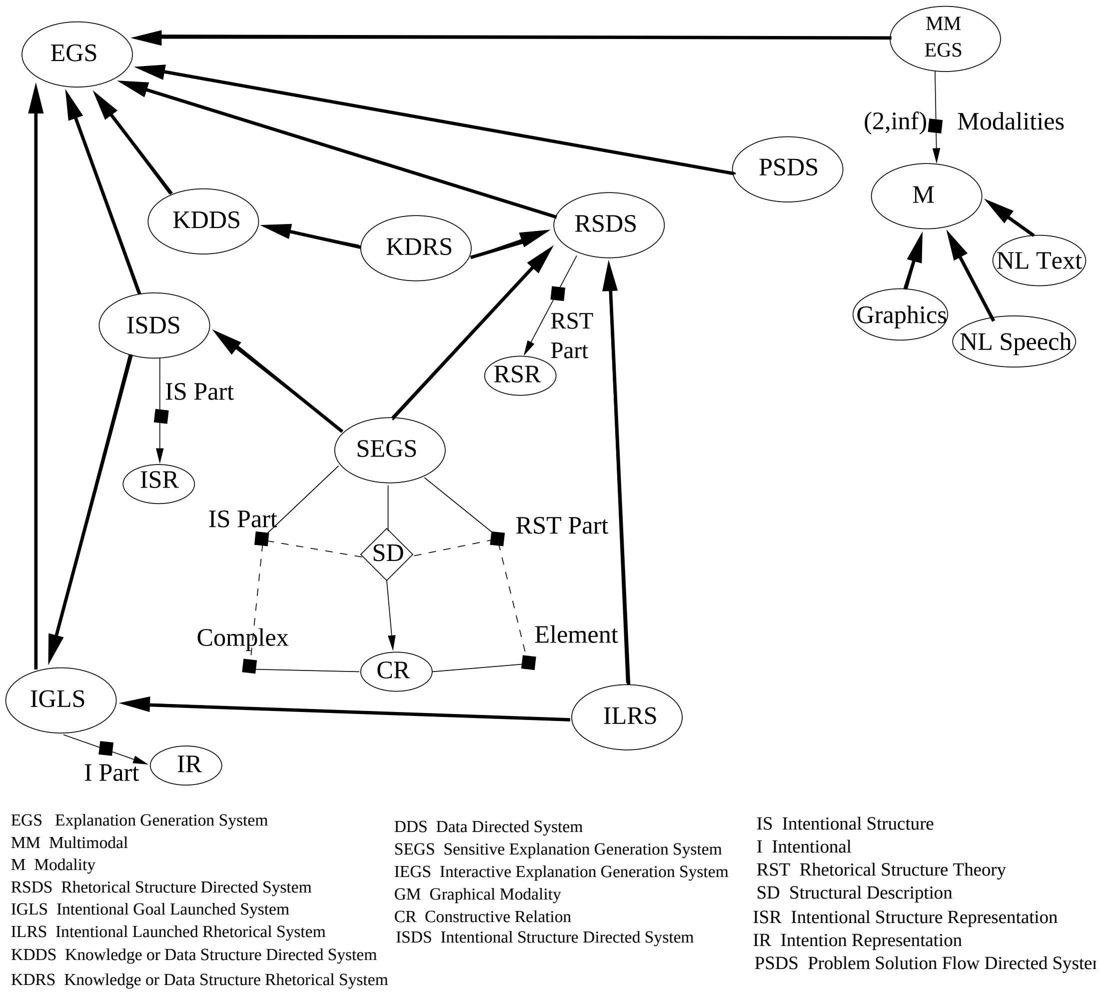


Figure 2.7: Adding intentional and rhetorical structures to explanation systems

Identification Schema

Identification (class & attribute/function) {Analogy/Constituency/Attributive/REnaming/Amplification}* Particular-illustration/Evidence+ {Amplification/Analogy/Attributive} {Particular-illustration/Evidence}

Example of application.-

An aircraft carrier is a surface ship with a DISPLACEMENT between 78000 and 80800 and a LENGTH between 1039 and 1063. Aircraft carriers have a greater LENGTH than all other ships and a greater DISPLACEMENT than most other ships. Mine warfare ships, for example, have a DISPLACEMENT of 320 and a LENGTH of 144. All aircraft carriers in the ONR database have REMARKS of 0, FUEL TYPE of BNKR, FLAG of BLBL, BEAM of 252, ENDURANCE RANGE of 4000, ECONOMIC SPEED of 12, ENDURANCE SPEED of 30 and PROPULSION of STMTURGRD. A ship is classified as an aircraft carrier if the characters 1 through 2 of its HULL NO are CV.

Figure 2.8: McKeown's Identification Schema.

in *curly brackets*, "{ }", are optional, the *asterisk*, "*", indicates that the preceding item may appear 0 or more times, the *plus sign*, "+", indicates that the preceding item could appear 1 or more times.

TEXT schemata were implemented using the *augmented transition network* (ATN) formalism [Woods 1970, Woods 1987]. A graph representing the ATN corresponding to the identification schema of Fig. 2.8 is depicted in Fig. 2.9[7].

[7]The actual ATNs implemented in the TEXT system to produce explanations were slightly different from those analysed as the original basis of McKeown's work. In Fig. 2.9, this difference is shown in the dashed edges, which were eliminated from the actual implementation.

Given an input question[8], to produce an explanatory response, TEXT dispatches two independent selective processes. The first one uses the input question as a *mask* to delimit the information to be extracted from the knowledge base into the, so called, *relevant knowledge pool*. The second one uses the input question again, but now to select a set of schemata suitable for the answer. Then, from the available information in the relevant knowledge pool, a decision is made to select just one of the schemata to be worked out. The ATN of the selected schema is then traversed from its initial node to the ending node producing the discourse structure and the propositional contents of the explanation. Every time a "FILL" edge is traversed the corresponding predicate (e.g., ATTRIBUTIVE in the case of the edge labelled "FILL ATTRIBUTIVE" in Fig. 2.9) is used to extract the propositions that match against it from the relevant knowledge pool. Whenever there are several options among the edges to choose or the propositions matching a predicate, focusing constraints [Grosz 1977] are used to define both the edges to be considered to output predicates and the propositions to be included in the final discourse.

Following a similar strategy, Lester & Porter (1997) developed their KNIGHT project. This system uses two organisative structures to build an explanation from a knowledge base, namely: a set of *knowledge-base accessors*, and a library of *explanation design packages*.

A *knowledge-base accessor* (KBA) is a structure that brings out a *view* of the knowledge base when it is superimposed on it. A *view* is a coherent subgraph of the knowledge base describing the structure and function of objects, the changes made to objects by processes, and the temporal attributes and temporal decompositions of processes. Some of KNIGHT's knowledge-base accessors are shown in Table 2.1. Given their arguments, these accessors locally mask out all the knowledge-base information that is not relevant for the explanation. The result of this operation is an *instantiated knowledge-base accessor*.

An *explanation design package* (EDP) is a tree-like structure that defines how the explanation will be organised. Figure 2.10 shows an example EDP designed to generate explanations about processes. The root node of this structure is an *exposition node*, which identifies the main task of the EDP. This node can be decomposed into several *topic nodes*, which subdivide the explanation matter into different *non-overlapping* topics. In the example, a process is explained by showing:

1. how a process fits into a taxonomy, its actor roles, and where it occurs (process overview),

[8]The original TEXT system of McKeown (1982) responded to three types of question, namely: *What is a <entity>?, What do you know about <entity>?,* and *What is the difference between <entity1> and <entity2>?* These questions could be asked to the system in the form of the three corresponding functions: (definition *<entity>*), (information *<entity>*), and (differense *<entity1>* *<entity2>*)(sic).

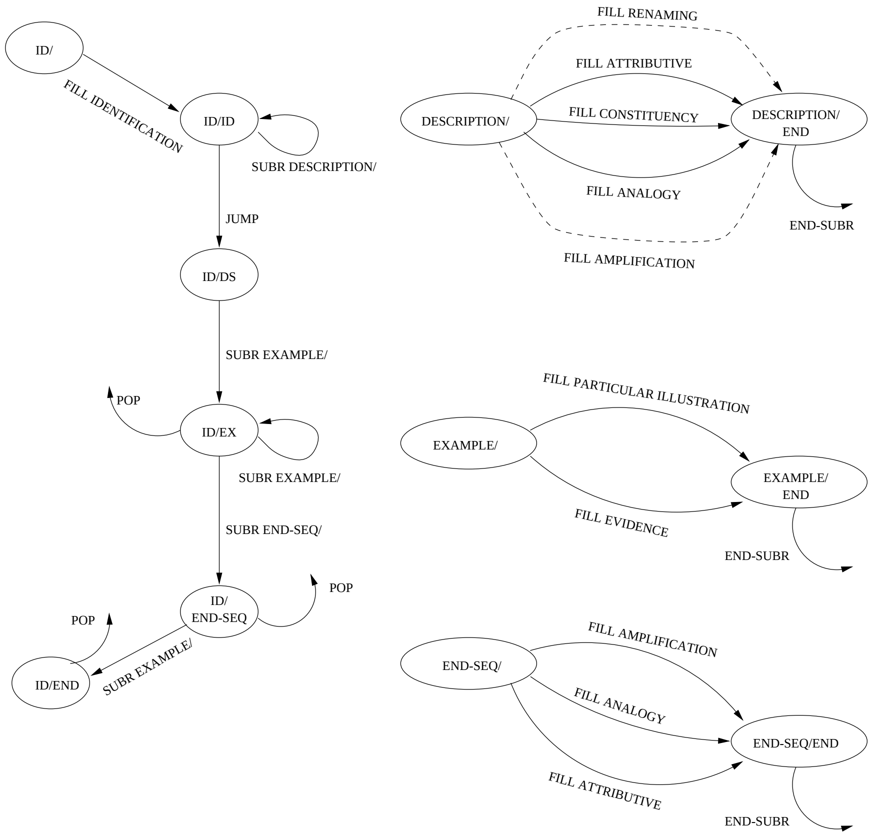


Figure 2.9: ATN corresponding to the identification schema

Table 2.1: KNIGHT's knowledge-base accessors

KB accessor	Arguments	Description of View
As-Kind-Of	*concept* *reference*	Finds a view of *concept* as a kind of *reference* concept.
Auxiliary-Process	*process* *view-type*	Finds temporal, causal, or locational information about *process* as specified by *view-type*.
Participants	*process* *reference*	Finds an "actor-oriented" view of *process* as viewed from the perspective of *reference* process.
Core-Connection	*process*	Finds the connection between *process* and a "core" process.
Functional	*object* *process*	Finds functional view of *object* with respect to *process*.
Partonomic-Connection	*object*	Finds the connection from *object* to a "superpart" of the object in the "partonomy".
Subevent	*process*	Finds a view of "steps" of the *process*.
Substructural	*object*	Finds a structural view of the parts of *object*.
Temporal-Step	*process*	Finds a view of *process* with respect to another process of which *process* is a "step".

2. how the "products" of a process are used by other processes (output actor fates topic),
3. how the process is related temporally to other processes (temporal info. topic), and
4. the steps of the process (process details).

A *topic node* can be decomposed into several *content specification nodes*. One of these nodes can contain a *knowledge-base accessor* or introduce a decomposition into several *knowledge-base accessors,* and additionally it can refer to *elaboration nodes*. The *elaboration nodes* are used to introduce additional *content specification nodes*. This hierarchical decomposition of an EDP goes down to the leaves, always *knowledge-base accessors*, which are used to define the final content of the explanation.

When an EDP is used to produce an explanation, the inclusion of a particular *instantiated knowledge-base accessor* on it is determined according to the centrality of the topic, the level of detail required, and specific implementation factors. These conditions are managed at the level of the *topic nodes*, so that a whole branch of the tree can be discarded from this level without further processing.

To integrate the final explanation, the leaves of the tree are developed into a linear structure of message specifications that serve as input to the realization component of the system. This component, based on a *functional unification formalism* grammar [Elhadad 1989, Elhadad 1993], translates the message specifications into a coherent natural language English text.

The systems under this category were able to answer basically the same sorts of question as the two preceding categories. The contribution of systems of this sort has to do more with the quality of the explanations than with the types of question they can answer, since issues like coherence, cohesion, and semantic unity of the explanations were addressed by them.

2.4 Acknowledging Intentionality in Explanations

Even though the category of system reviewed in the previous section recognised the centrality of a discourse purpose in the making up of its structure, the bringing to light of the participant dialogue actors was not made explicit. Now, in any dialogue, the purpose of the communication is for one person to alter the cognitive state of the other. Thus, the systems that explicitly take into account the actors participating in a dialogue must represent their cognitive states, which can be defined in terms of their beliefs, desires, and intentions [Hintikka 1969, Allen

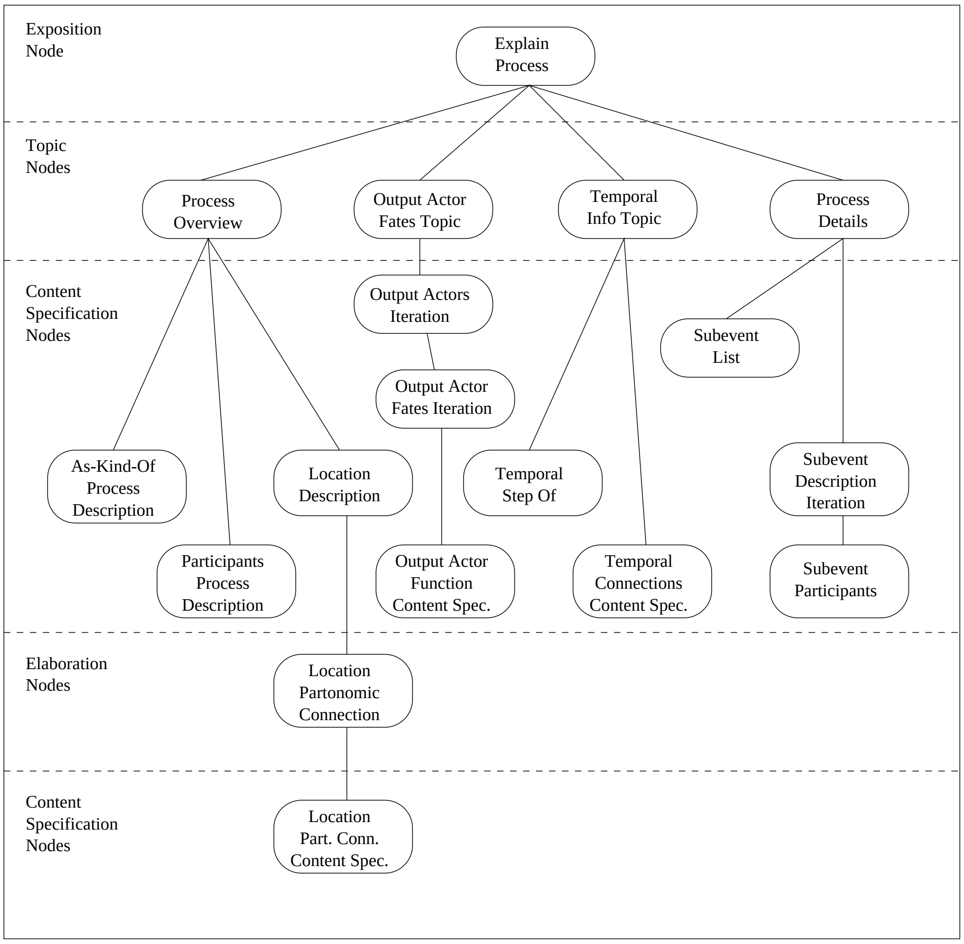


Figure 2.10: An EDP designed for explaining processes

1983, Allen 1995]. In Fig. 2.7 the node corresponding to systems of this sort is named *intentional goal launched system*, because the way explanations are generated by these systems is mainly guided by the intentions of changing the cognitive state of the hearers.

Possibly the most influential theoretical work among systems that take intentionality into account is that of speech acts theory [Grice 1957, Austin 1962, Searle 1969, Searle 1979]. Though with different perspectives, the philosophers who developed the theory, postulate that three kinds of act occur when something is uttered, namely:

the locutionary act: the act of uttering a sequence of words, and

the illocutionary act: the act that the speaker performs in saying the words, i.e., the speaker's intention behind the utterance.

Several implemented explanation systems have roots in this theory. The first attempt to operationalise the speech acts theory was due to P. R. Cohen, who developed an axiomatization of the theory, and, based on it, created a system which was able to reason about beliefs to produce a plan for the hearer to recognise the speaker's intentions [Cohen 1978, Cohen & Perrault 1979]. However, his system did not produce utterances, but rather descriptions of the actions to be performed. Later on, Appelt & Kronfeld (1987), elaborating on Cohen & Perrault (1979) work, developed their KAMP system. This system was based on a set of axioms of the theory, and a set of *action summaries*, which acted as plan operators, with preconditions and effects, to produce a plan that was always verified against the axioms. Appelt distinguished four kinds of linguistic action that were organised in a hierarchy, as shown in Fig. 2.11. By means of these actions, KAMP explicitly represents the relationship between speaker intentions and speech acts. Appelt's work, however, did not explicitly take into account the rhetorical patterns that occur in natural discourses.

Working towards linking the rhetorical structure with the speaker and hearer intentions, Mann & Thompson (1988) developed their *rhetorical structure theory* (RST). This theory defines the relationship between two spans of text, the *nucleus* (N) and the *satellite* (S), by establishing constraints on N, constraints on S, constraints on the combination of N and S, and the effect that the speaker is trying to achieve on the hearer's beliefs or inclinations. See, for instance, the RST relation shown in Fig. 2.12. The example shown in the figure exhibits how the *evidence* in the satellite is used to increase the hearers belief in the declaration stated in the nucleus.

The work of Mann & Thompson (1988) introduced a new category of systems that take into account both the intentions of the interlocutors and the rhetorical structure of the discourse. In Fig. 2.7 this sort of system is named *intentional*

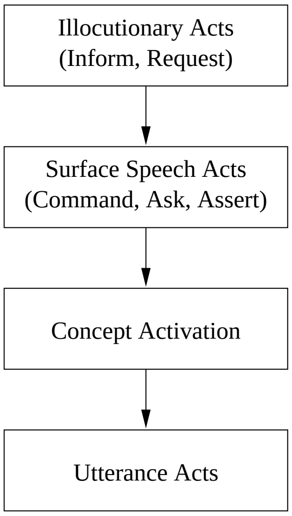


Figure 2.11: KAMP's hierarchy of linguistic actions

relation name:	EVIDENCE
constraints on N:	The hearer might not believe N to a degree satisfactory to the speaker
constraints on S:	The hearer believes S or will find it credible
constraints on the N+S combination:	
	The hearer's comprehending S increases the hearer's belief on N
the effect:	The hearer's belief on N is increased
locus of the effect:	N

Example:

Nucleus: The program as published for calendar year 1980 really works.

Satellite: In only a few minutes, I entered all the figures from my 1980 tax return and got a result wich agreed with my hand calculations to the penny.

Figure 2.12: A RST relation

launched rhetorical system, because the intentional launching of the system is rendered as a rhetorical structure in the final discourse.

The RST was directly operationalised in the works by E. H. Hovy [Hovy 1991, Hovy 1993]. The RST relations were implemented as planning operators, like the one shown in Fig. 2.13. The RESULTS field of this operator guarantees that both the speaker and the hearer will mutually believe that the relationship SEQUENCE-OF holds between two input entities (?PART and ?NEXT). The NUCLEUS requirements and the SATELLITE requirements fields specify that both entities are believed to be TOPICs by the speaker and the hearer. The NUCLEUS+SATELLITE requirements field states that ?PART must precede ?NEXT. The NUCLEUS growth points and the SATELLITE growth points fields suggest additional material to be included by expanding other operators whose RESULTS field unifies with the propositions in the growth points. The ORDER field establishes the order in which the NUCLEUS and the SATELLITE should be rendered. Finally, the RELATION-PHRASES field introduces cue words to relate the NUCLEUS and SATELLITE resulting texts.

The main difference between the operationalisation of the *speech acts theory* and that of the RST is that, while in the former the surface text is defined starting from the intentions of the speaker (illocutionary acts), in the latter, it is the textual structure, given by RST relations, that defines the effects in terms of the cognitive states of the interlocutors. This subtle distinction imposes the restriction that when using a particular RST relation to produce NL text, only one intention can be considered, namely, the one stated in the RESULTS filed of the RST planning operator (or the effects part of a RST relation). Notice that the key idea behind Searle's speech acts theory is that the same utterance is used to convey different intentions. To overcome this problem, Moore (1995) developed an interesting combination of *speech acts theory* and Mann & Thompson's RST. Although several variants to this combination have been proposed for the generation of NL explanations [Maybury 1992, Wahlster 1991, Pilkington & Grierson 1996], the one that best illustrates it, is the work of Moore. In her work, she established a relation between *communicative goals*, which embody the speaker's intentions, and *linguistic goals*, which establish the linguistic means for achieving the communicative goals. There are two classes of linguistic goal, namely: rhetorical goals, and speech acts. A speech act can be directly rendered as part of the surface text of the explanation, while a rhetorical goal can introduce further decompositions by the application of other plan operators which comply with the rhetorical goal.

Figures 2.14 and 2.15 show two operators developed for Moore's *Program Enhancement Advisor* (PEA) system. The NUCLEUS of the first operator (Fig. 2.14), is the speech act RECOMMEND, which can be directly rendered as output text of the explanation The operator contains two optional satellites, both communicative goals (intentional goals), PERSUADED and COMPETENT. Any of these

```
Name:  SEQUENCE

Results:
    ((BMB SPEAKER HEARER (SEQUENCE-OF ?PART ?NEXT)))

Nucleus requirements/subgoals:
    ((BMB SPEAKER HEARER (TOPIC ?PART)))

Satellite requirements/subgoals:
    ((BMB SPEAKER HEARER (TOPIC ?NEXT)))

Nucleus+Satellite requirements/subgoals:
    ((NEXT-ACTION ?PART ?NEXT))

Nucleus growth points:
    ((BMB SPEAKER HEARER (CIRCUMSTANCE-OF ?PART ?CIR))
     (BMB SPEAKER HEARER (ATTRIBUTE-OF ?PART ?VAL))
     (BMB SPEAKER HEARER (PURPOSE-OF ?PART ?PURP)))

Satellite growth points:
    ((BMB SPEAKER HEARER (ATTRIBUTE-OF ?NEXT ?VAL))
     (BMB SPEAKER HEARER (DETAILS-OF ?NEXT ?DETS))
     (BMB SPEAKER HEARER (SEQUENCE-OF ?NEXT ?FOLL)))

Order:
    (NUCLEUS SATELLITE)
Relation-phrases:
    ("" "then" "next")
```

Figure 2.13: Hovy's RST planning operator SEQUENCE

EFFECT:
```
  (GOAL ?hearer (DO ?hearer ?act))
```
CONSTRAINTS:
```
  (NUCLEUS)
```
NUCLEUS:
```
  (RECOMEND ?speaker ?hearer (DO ?hearer ?act))
```
SATELLITES:
```
  (((PERSUADED ?hearer (DO ?hearer ?act)) *optional*)
   ((COMPETENT ?hearer (DO ?hearer ?act)) *optional*))
```

English paraphrase

To make the hearer to do an *act*,

IF

This text span is to appear in nucleus position

THEN

1. Recommend the *act*

AND optionally,

2. Achieve a state where the hearer is persuaded to do the *act*
3. Achieve a state where the hearer is competent to do the *act*

Figure 2.14: Moore's plan operator for recommending an act

EFFECT:
```
  (PERSUADED ?hearer (DO ?hearer ?act))
```
CONSTRAINTS:
```
  (AND (STEP ?act ?goal)
       (GOAL ?hearer ?goal)
       (MOST-SPECIFIC ?goal))
```
NUCLEUS:
```
  (FORALL ?goal
       (MOTIVATION ?act ?goal))
```
SATELLITES:
  nil

English paraphrase
To achieve a state in which the hearer is persuaded to do an *act*,
IF
  The *act* is a step in achieving some *goal(s)* of the hearer, and
  the *goal(s)* are the most specific along any refinement path,
THEN
  Motivate the *act* in terms of those *goal(s)*.

Figure 2.15: Moore's plan operator for persuading a user to do an act

communicative goals can be pursued using another operator whose EFFECT field matches it. For instance, the operator shown in Fig. 2.15 can be used as an attempt to satisfy the goal stated in the first satellite of the operator shown in Fig. 2.14. Notice, then, that to satisfy a particular goal there can be many operators available. The selection of one particular operator is made by different heuristics that take into account the discourse generated so far, the user model, and other parameters that depend on the expected quality of the explanation.

The application of operators like the ones shown in Figs. 2.14 and 2.15, will define a plan for the explanation in the form of a tree whose leaves are the speech acts to be rendered as the surface text of the explanation. Figure 2.16 shows a fragment of a tree corresponding to a plan generated by the PEA system with plan operators like these. Nodes **1, 2** and **3** are the rendition of some of the goals of the operator displayed in Fig. 2.14, while nodes **3** and **4** come from the operator displayed in Fig. 2.15. In the leaves of the tree, nodes **2, 7** and **9**, the output text is displayed at the bottom of the box, along with the goal. Any branch labelled with **N** represents the *nucleus* of a rhetorical relation, while any of those labelled with **S** represents a *satellite*. Both the intentional structure and the rhetorical structure are embedded in trees of this sort. Note, for instance, that the branching starting from nodes **2** and **3** defines a rhetorical relation *nucleus/ satellite*, and that the decomposition of node **5** into node **6** defines a structural relation between two intentional goals. The text generated by this fragment of a plan, which intends to convince the user to perform a replacement of Lisp code in his or her input program, is as follows:

> *You should replace (SETQ X 1) with (SETF X 1). SETQ can only be used to assign a value to a simple variable.* ***In contrast****, SETF can be used to assign a value to any generalized variable.*

In the preceding text, the phrase marked in bold type is introduced by the CONTRAST rhetorical relation, shown in node **8** in Fig. 2.16.

Since in systems like Moore's, intentional goals are expressed in terms of rhetorical goals and speech acts, a new category of system is introduced in Fig. 2.7 as the node named *sensitive explanation generation system*[9]. This name arises from the fact that this sort of system is more sensitive to the user's feedback because of the explicit combined representation of the intentional structure and the rhetorical structure.

The systems under the two categories introduced in this section were potentially able to answer a wider range of questions. In particular the system developed by Moore (1995), was able to answer follow-up questions, like the following:

1. Why are you trying to achieve *<goal>*?
2. Why are you using *<method>* to achieve *<goal>*?
3. Why are you doing *<act>*?
4. How do you achieve *<goal>*?
5. How did you achieve *<goal>*?
6. What is *<concept>*?
7. What is the difference between *<concept1>* and *<concept2>*?

To answer questions like these, Moore's system had to rely not only on the representation of its plan operators, but on the rich knowledge representation of the underlying expert system, which allowed it to justify the actions taken [Swartout et al. 1991].

[9]It should be noted that a new graphical device is introduced in Fig. 2.7 along with this node: the diamond labeled SD, which introduces a *structural description* as defined in Brachman-style semantic networks. In this case, this particular *structural description* is defining a structural relationship between the intentional structure and the rhetorical structure roles of the node, namely, that the intentional part is expressed in terms of the rhetorical part.

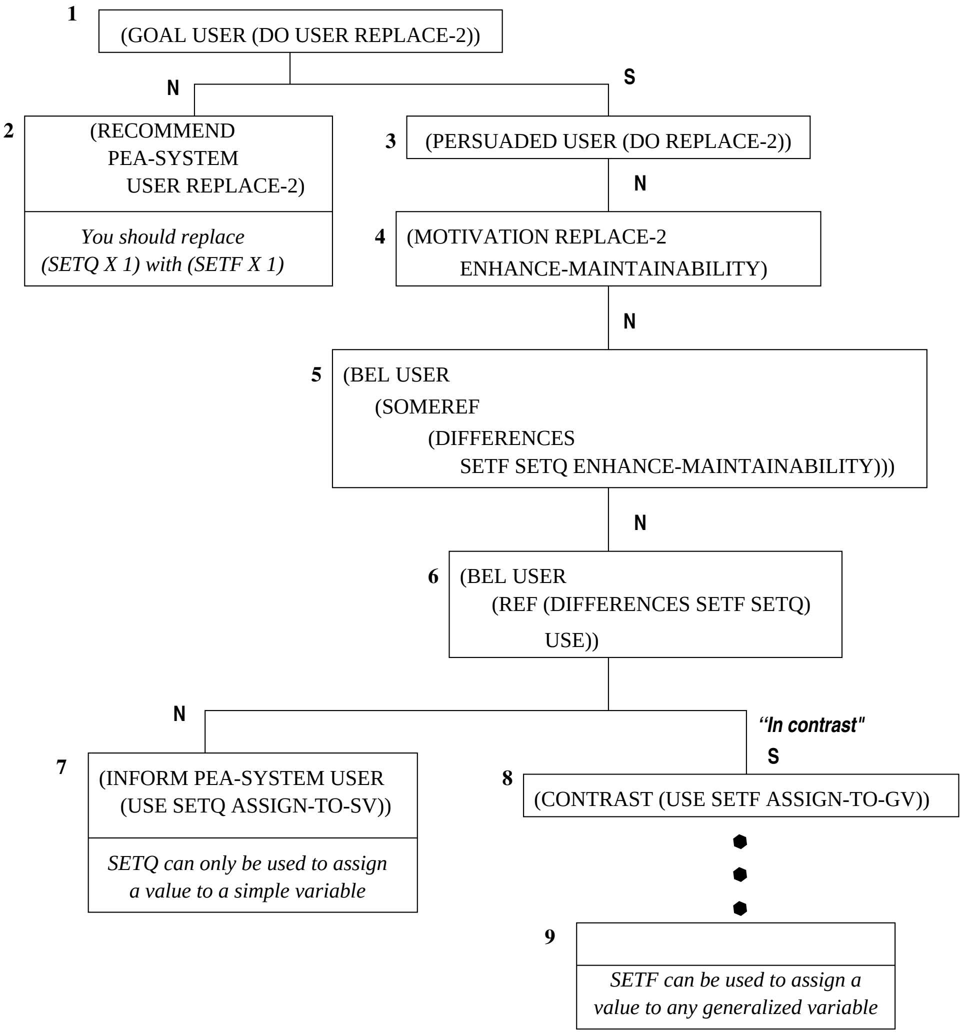


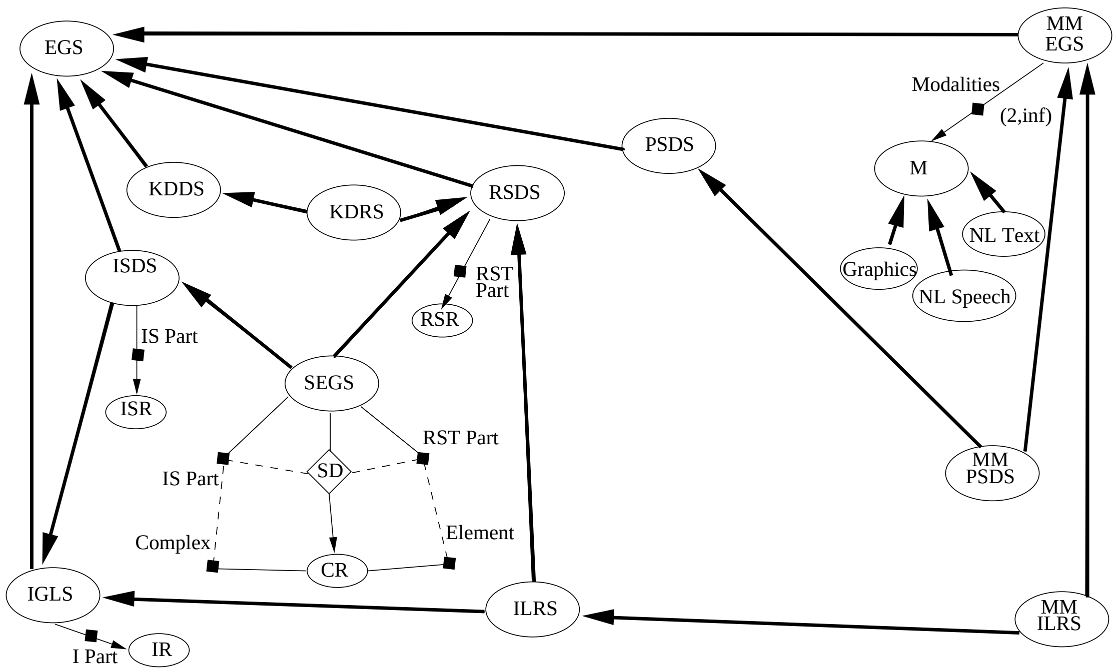
Figure 2.16: A fragment of a plan for an explanation generated with the PEA system

2.5 Incorporating Multimodality in Explanation Systems

The arrival of multimodality in computing opened a new dimension to explanation generation systems. The natural trend in this newly introduced category of system, was to adapt the existing approaches to explanation generation to accept other modalities. This was the case, for instance, of the COMET system which adapted McKeown's schemas [McKeown 1982, McKeown 1985] to accept the graphical modality [Feiner & McKeown 1991, McKeown et al. 1990]. Mark T. Maybury did the same with his plan operators [Maybury 1990, Maybury 1992, Maybury 1993*a*]. In these cases, the fundamental characteristics of the resulting systems are directly derived from their ancestors. Others have built on the foundation provided by other authors, like Wahlster et al. (1993), who elaborated on the research line taken by Moore & Paris (1989). In these cases, again, the fundamental characteristics are inherited from the parent line of research. To cover these multimodal explanation generation systems , two new nodes are introduced in the explanation and multimodal systems taxonomy being developed, namely: the *multimodal intentional launched rhetorical system* (MM-ILRS), and the *multimodal problem solution flow directed system* (MM-PSDS), as shown in Fig. 2.17. The first category is devoted to explain information as it is stored in a knowledge-base, while the second is dedicated to explain the trace of a problem solution thread.

The Graflog system constitutes a very singular case within multimodal systems, because it implemented a multimodal input, while its output is just graphical [Pineda 1989, Pineda 1992*b*, Pineda 1993]. If this system incorporated the NL text modality for its output, it could easily fit into the MM-PSDS category. This system has been briefly described in Section 2.1 on p. 23.

Below, some salient examples of multimodal systems are described. Most of these systems have been posed as interfaces to expert systems. Hence, their main task is to explain how to do something based on the rules of the expert system. For instance, the XTRA system explains how to fill in a tax form [Wahlster 1991]. The COMET system explains how to repair military equipment [Feiner & McKeown 1991, McKeown et al. 1990]. Maybury's system explains how to find a route in a cartographic map [Maybury 1993*a*]. The WIP system explains how to operate or assemble different items of equipment [André et al. 1993]. All these systems, take the predefined rules applied for a particular problem, and with their instantiations explain them to the final user.



EGS Explanation Generation System
MM Multimodal
M Modality
RSDS Rhetorical Structure Directed System
IGLS Intentional Goal Launched System
ILRS Intentional Launched Rhetorical System
KDDS Knowledge or Data Structure Directed System
KDRS Knowledge or Data Structure Rhetorical System

DDS Data Directed System
SEGS Sensitive Explanation Generation System
IEGS Interactive Explanation Generation System
GM Graphical Modality
CR Constructive Relation
ISDS Intentional Structure Directed System

IS Intentional Structure
I Intentional
RST Rhetorical Structure Theory
SD Structural Description
ISR Intentional Structure Representation
IR Intention Representation
PSDS Problem Solution Flow Directed System

Figure 2.17: Multimodality in explanation systems

2.5.1 The XTRA system

XTRA (eXpert TRAnslator) is an intelligent multimodal interface for expert systems that combines natural language, graphics and pointing for input and output [Wahlster 1991]. The system provides natural language access to an expert system, which assists the user in filling out a tax form. The user can query the system by means of a combination of natural language and deictic references. The system answers with a combination of a natural language text and pointing gestures to the graphical material already on the display.

As far as input is concerned, XTRA is one of the most interesting systems, since it incorporates the ability to handle *deictic references* (see definition on the footnote of p. 14) embedded in its natural language inputs. Although there are several other systems which combine natural language and pointing [Carbonell 1970, Brown et al. 1979, Thomson 1986, Neal & Shapiro 1991], XTRA resolves deictic references in a very singular fashion. Since a document can embody a complex graphical structure, there is, almost always, a difference between the object at which the user actually points and the object at which he or she attempts to point. To determine the precise object being pointed by the user, XTRA takes into account several considerations, including document structure, discourse context and pointing mode. XTRA supports four pointing modes, namely: exact pointing with a pencil, standard pointing with the index finger, vague pointing with the entire hand, and encircling regions. Each of these modes is associated with a granularity value (i.e. the number of pixels surrounding the pointing icon).

Since XTRA is an interface to an expert system, its whole content is decided by this expert system. Regarding presentation planning, XTRA incorporates two relevant components. The first one is the simultaneous production of deictic descriptions and pointing actions in its presentations. The second is a mechanism called anticipation feedback, which consists of checking the realised plan against the user model. If this realisation does not have the desirable effect relative to the assumed knowledge of the user, the system has to replan its utterance.

2.5.2 The COMET system

COMET (COordinated Multimedia Explanation Testbed) is a knowledge-based system that produces coordinated, interactive explanations that combine text and three dimensional graphics, all of which are generated on the fly [Feiner & McKeown 1991]. COMET's domain is the maintenance and repair of a military radio receiver-transmitter. The user interacts with the system by means of a menu to request instructions for a procedure or a diagnostic. In the latter case, an underlying expert system is invoked and it can determine if a set of tests ought to be performed in order to ascertain the cause of the problem. In

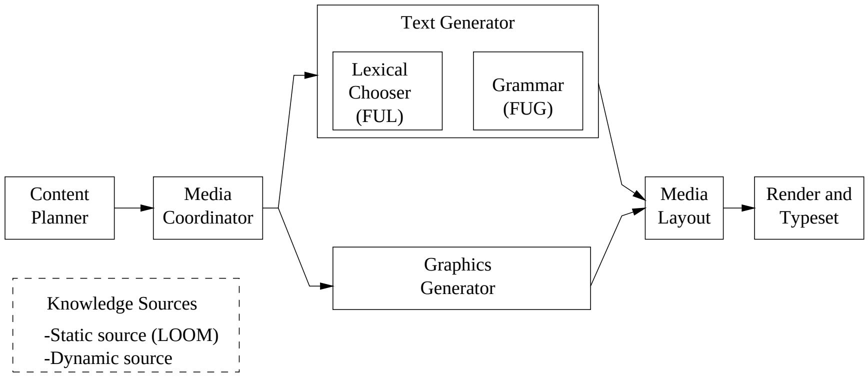


Figure 2.18: The architecture of the COMET system

such a case, it calls an explanation component to tell the user how to carry out these tests. An explanation consists of one or more steps presented in a series of displays. Each display can contain synthesised graphical information as well as textual explanations.

In COMET, content selection occurs before the content is distributed between the different modalities. Once the user submits a goal, COMET's content planner (Fig. 2.18) selects a rhetorical *schema* [McKeown 1982] from its schemas library to determine which information from the underlying knowledge sources should be included in its presentation. It uses three different domain knowledge sources: a static representation of domain objects and actions, written in the knowledge representation language LOOM [MacGregor & Brill 1992], a diagnostic rule-base, and a detailed geometric knowledge base needed for generating graphics. Each LOOM item is consumed as the selected schema, implemented as an augmented transition network [Woods 1987], is traversed. The output of this process is a hierarchy of logical forms [Allen 1995] which constitutes the propositional content of the presentation.

After the content has been selected, the media coordinator(Fig. 2.18) takes the logical forms from the preceding stage and refines them by annotating directives that indicate which portions are to be produced by the text generator and which by the graphics generator. To this aim, COMET's media coordinator unifies the input logical forms with a *media coordination grammar* in which the media selection strategies are embedded. The logical forms can also be enriched with the accomplishments of each generator. COMET emphasises two aspects of media coordination. First, the use of a common content description language makes it possible for each generator to query and reference other generators. The same annotated description of what is to be described is passed to each generator. This allows each generator to use information about what the other generators present to influence its own presentation. Second, the use of bidirectional interaction between the media specific generators allows COMET to generate explanations that are structurally coordinated and that contain cross-references between media.

2.5.3 Maybury's explanation planner

Maybury's ideas were embodied in the text explanation system TEXPLAN (Textual EXplanation PLANner) [Maybury 1992] that originally focused on the production of natural language text explanations. This system evolved to a more general explanation planner which supports explanations composed of natural language text and graphics, which was tested as the explanation module of the Map Display System [Hilton & Anken 1990, Maybury 1993*a*]. The Map Display System is focused on the design of dynamic, narrated animation of routes over an object-oriented cartographic information system. The system can be queried with questions such as: Where is London? or How do I get from London to Manchester? and it replies depicting the appropriate map and giving a relevant natural language explanation. The map highlights the route, and the explanation consists of a procedure to get from one place to another.

This system has been built around the so called *communicative acts*, whose taxonomy is shown in Fig. 2.19. Communicative acts include rhetorical acts, linguistic acts, graphical acts, non-linguistic auditory acts (e.g., snap, ring), and physical acts (e.g., facial expressions). Of special interest for this work are rhetorical acts. From the point of view of Maybury's work, they can be defined as sequences of linguistic or graphical acts which are used to achieve certain media-independent rhetorical goals. In Fig. 2.19, several examples of linguistic and graphical acts are mentioned.

Here, communicative acts are formalised as plans, which in turn are implemented as *plan operators*. A plan operator, implementing a rhetorical act, must include constraints and preconditions, the refinement or decomposition of the act into subacts, and its intended effects or postconditions. Plan operators are encoded in an extension of first order logic which allows for optionality within decomposition. When a user queries the system, a goal is posted to the explanation planner. Then it selects those operators that can potentially achieve the given goal. The selected operators are prioritised using preference metrics. At once, the system tries to execute the decomposition of the weighted operators until one succeeds, i.e. until the operator is decomposed into primitive acts that can be directly realised as natural language text or as a graphical depiction. An example of such operators is given below:

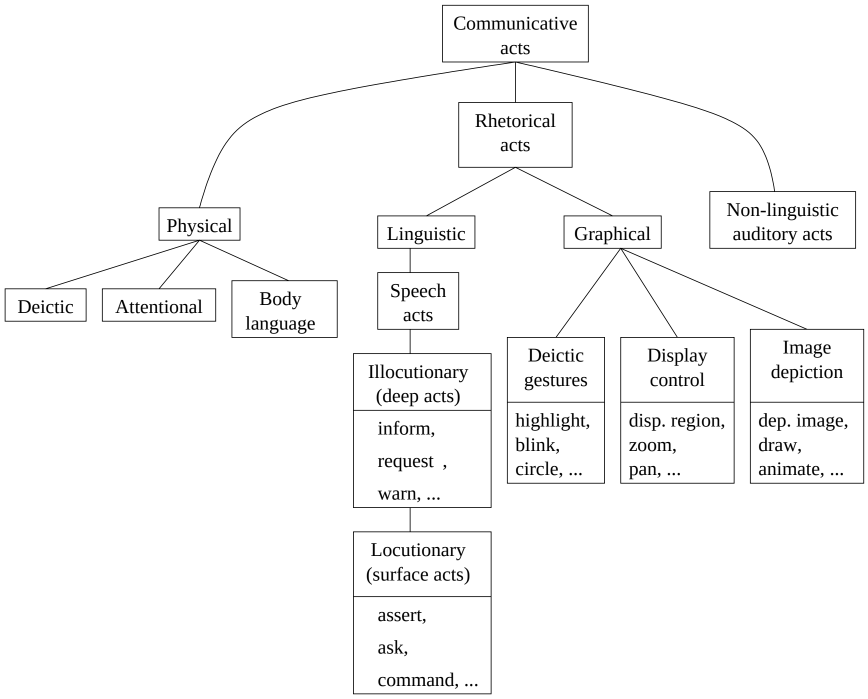


Figure 2.19: Communicative acts taxonomy

Frame 1. Uninstantiated linguistic plan operator

NAME	Identify-location-linguistically
HEADER	Identify(S, H, *entity*)
CONSTRAINTS	Entity?(*entity*)
PRECONDITIONS	WANT(S, KNOW(H, Location(*entity*)))
EFFECTS	KNOW(H, Location(*entity*))
DECOMPOSITION	Assert(S, H, Location(*entity*))

Here variables are printed in *italics*, predicates are ended with a question mark ('?'), functions are written in lower case with except for the first letter which must be a capital letter (e.g., Location), and intentional operators are written in uppercase letters (e.g., KNOW). Then, what does this operator say? It says that:

- provided that *entity* is in fact an **entity**, and
- provided also that the speaker (S) wants the hearer (H) to know its location,
- the speaker will inform the hearer what the location of *entity* is, and

- this will have the intended effect that the hearer knows that location.

Another example involving both linguistic and graphical acts is the following:

Frame 2. Plan operator for graphical and textual display

NAME	Identify-location-linguistically&visually
HEADER	Identify(*S*, *H*, *entity*)
CONSTRAINTS	Cartographic-entity?(*entity*)
PRECONDITIONS	Visible(*entity*) $\wedge$ WANT(*S*, KNOW(*H*, Location(*entity*)))
EFFECTS	KNOW(*H*, Location(*entity*))
DECOMPOSITION	Indicate-deictically(*S*, *H*, *entity*) Assert(*S*, *H*, Location(*entity*))

A cartographic entity is, among other things, a graphical entity that can be displayed. A difference between constraints and preconditions is that nothing can be done to satisfy an unsatisfied constraint, whereas an unsatisfied precondition can be posted as a subgoal to the explanation planner. This could be, for instance, the case of the **Visible** precondition in the previous operator. If *entity* was not visible, the Visible(*entity*) precondition turned into a new goal could trigger the Make-entity-visible operator shown below

Frame 3. Plan operator for map display control

NAME	Make-entity-visible
HEADER	Make-visible(*entity*)
CONSTRAINTS	Cartographic-entity?(*entity*)
PRECONDITIONS	Displayed(*entity*)
EFFECTS	Visible(*entity*)
DECOMPOSITION	Display-region(*entity*)

As an example, suppose that a user asks the system "Where is Chemnitz?". This question is posted to the system as the goal 'Identify(SYSTEM, USER, #<Chemnitz>)'. The plan library is then consulted and all plan operators whose header unifies with this goal and which satisfy their constraints are put aside. To decide which one is to be applied, *preference metrics* is used. In the reviewed example both Frame 1 and Frame 2 are applicable. Frame 2 is selected because operators that utilize both text and graphics are preferred over simple textual operators. The result of the unification process on the selected operator brings out an instantiated plan operator like the following:

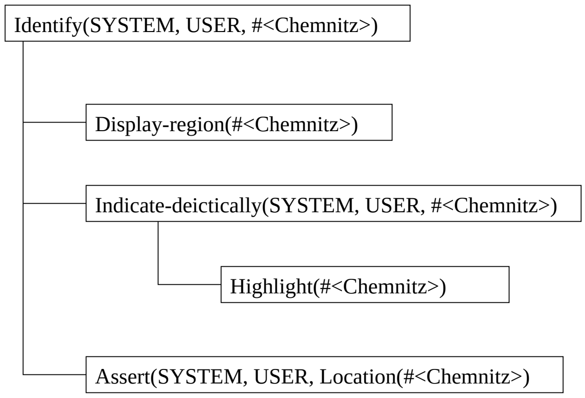


Figure 2.20: Multimedia plan to identify Chemnitz

Frame 4. Instantiated Identify plan operator

NAME	Identify-location-linguistically&visually
HEADER	Identify(SYSTEM, USER,#<Chemnitz>)
CONSTRAINTS	Cartographic-entity?(#<Chemnitz>)
PRECONDITIONS	Visible(#<Chemnitz>) ∧
	WANT(SYSTEM,
	KNOW(USER, Location(#<Chemnitz>)))
EFFECTS	KNOW(USER, Location(#<Chemnitz>))
DECOMPOSITION	Indicate-deictically(SYSTEM,
	USER, #<Chemnitz>)
	Assert(SYSTEM, USER, Location(#<Chemnitz>))

In this operator, all variables have been instantiated according to their values in the original goal. Several new goals can result from the application of an operator like this, either because it has some preconditions unsatisfied or because it is decomposed into several simpler subacts. In this example, according to some context, the application of the operator could result in the hierarchical decomposition shown in Fig. 2.20. The leaves of this tree are primitive acts which can be directly realised either graphically or linguistically.

A more complex task could be that of explaining a route, answering a question like "How do I get from Wiesbaden to Frankfurt?" The system uses the following operator to build up the answer to that question:

Frame 5. Explain-route plan operator

NAME	Explain-route-linguistically&visually
HEADER	Explain-route(*S, H, from, to*)
CONSTRAINTS	Cartographic-entity?(*from*) $\wedge$ Cartographic-entity?(*to*) $\wedge$ *path*
PRECONDITIONS	Visible(*from*) $\wedge$ WANT(*S*, KNOW-HOW(*H*, Go(*from, to*)))
EFFECTS	KNOW-HOW(*H*, Go(*from, to*)) $\wedge$ $\forall$ *seg* $\in$ *path* KNOW(*H*, Subpath(*seg, path*))

DECOMPOSITION	$\forall$ *seg* $\in$ *path*
	    Indicate-deictically(*S, H*, source(*seg*))
	    Command(*S, H*,
	        Do(*H*, Go(source(*seg*),
	                link(*seg*),
	                destination(*seg*))))
	    Indicate-deictically(*S, H*, link(*seg*))
	    Indicate-direction(*S, H*, source(*seg*),
	                    link(*seg*),
	                    destination(*seg*))
	Identify(*S, H, to*)

WHERE	*path* = cartographic-path(*from, to*)

Once this operator is applied it could bring back the hierarchical plan shown in Fig. 2.21, which linearised could be realised as the following multimedia explanation (graphical acts are enclosed in parenthesis and are written in italics):

> *(Display a map around Wiesbaden)(highlight Wiesbaden)* From Wiesbaden take Autobahn A66 Northeast for thirty-one kilometers to Frankfurt-am-Main. *(highlight Autobahn A66)(indicate direction with blinking arrow)(highlight Frankfurt-am-Main)* Frankfurt-am-Main is located at 50.11° latitude and 8.66° longitude.

2.5.4 The WIP system

This system is able to generate simple German or English explanations for using an expresso machine, assembling a lawn-mower or installing a modem [André & Rist 1993]. WIP's input consists of a formal description of the communicative

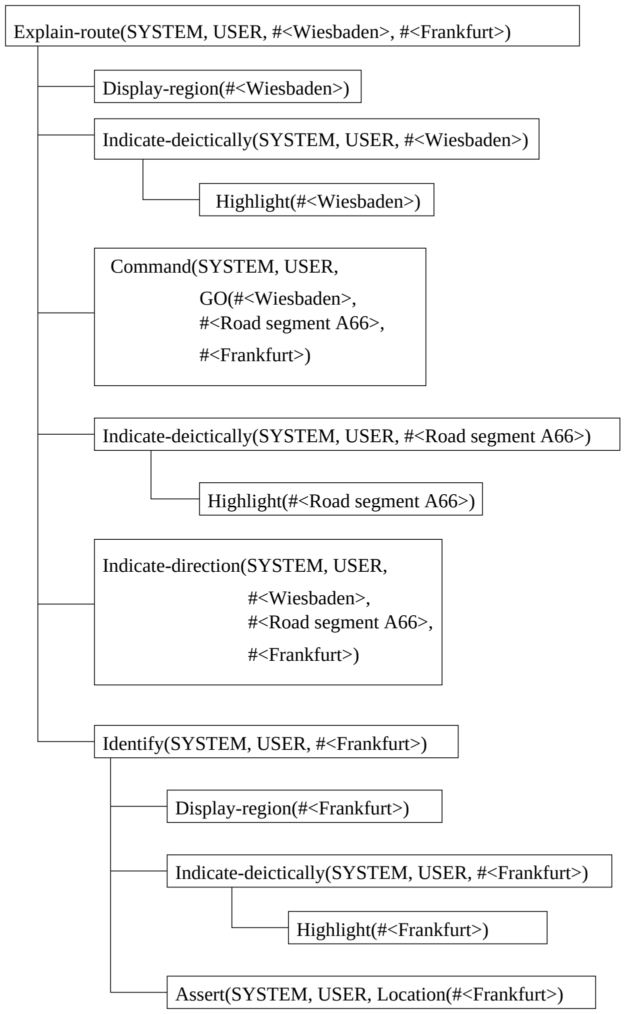


Figure 2.21: Hierarchical plan to explain a route

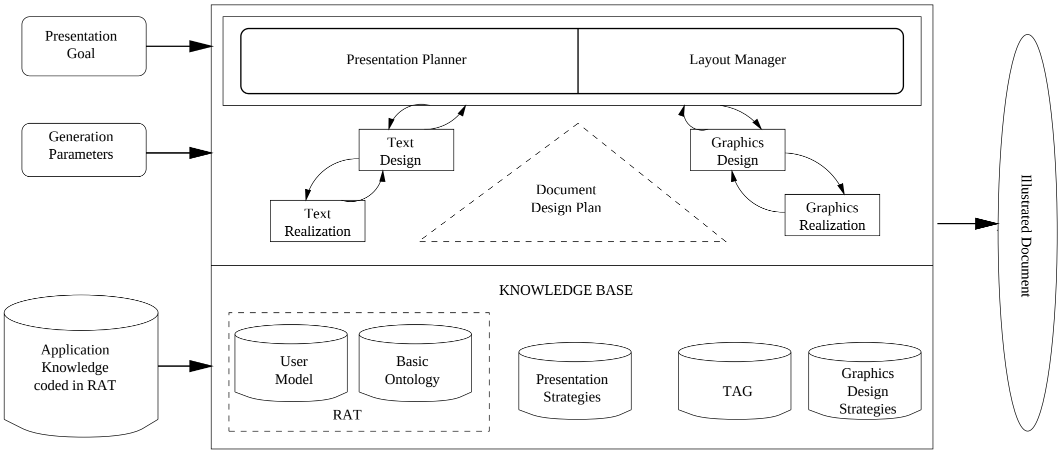


Figure 2.22: The architecture of the WIP system

intent of a planned presentation. The system responds to the stated intent with the generation of a variety of multi-modal documents. The generation process is controlled by a set of parameters such as target group, presentation objective, resource limitations and target language. If the system is commanded to give a presentation on how to fill an expresso machine with water, the response is a series of synthesised graphics and natural language texts showing how to lift the lid, how to remove the cover to fill the water container and how to fill the machine with cold tap water.

In WIP, content planning begins at the content selection phase. In particular, modality attachment occurs during content selection. This phase is performed by the presentation planner (Fig. 2.22) which takes into account that the content selection and the modality allocation might constrain each other. The system takes a fine-grained approach to media coordination in a three dimensional domain. Each piece of information to be communicated is assigned sequentially to its generators. Both, the text generator and the graphics generator are decomposed into a design component and a realization component. The evaluation of potential success is based on the knowledge of what has been generated so far. These evaluations are returned to the planner to help determine media assignments. For instance, layout considerations can influence the early stages of the planning process and constrain the media specific generators. WIP uses plan operators to design presentations [André & Rist 1993].

The main knowledge sources in WIP's architecture (Fig. 2.22) are [Wahlster et al. 1993]:

1. basic ontology and user model, both represented in RAT (**r**epresentation of **a**ctions using **t**erminological logics),

2. presentation strategies, declaratively coded as RST (**r**hetorical **s**tructure

theory) operators,

3. a natural language grammar, represented in the TAG (**t**ree **a**djoining **g**rammar) formalism, and

4. graphical design strategies.

WIP's operation is mainly defined through the interaction between the presentation strategies component and the RAT written modules. The natural language, and graphical design strategies are devoted to the surface rendering of the material already produced.

A presentation strategy has the following components:

Header. This is a complex communicative act (e.g., to enable an action, to compare two or more items, and to describe an object, etc.).

Effect. This is an intentional goal (e.g., the user knows a particular object).

Applicability conditions. These specify when the strategy may be used, and constrain the variables to be instantiated.

Main acts. A main act is a rhetorical act that is central to the goal.

Subsidiary acts. These are acts that contribute or are used to support the main acts. It seems that a presentation strategy is decomposed into just one main act and any number of subsidiary acts.

As an example, consider the following presentation operator from WIP

```
(def-presentation-strategy
  :Header (Describe P A (Orientation ?orientation) G)
  :Effect (BMB P A (Has-Orientation ?x ?orientation))
  :Applicability-Conditions
    (Bel P (Has-Orientation ?x ?orientation))
  :Main-Acts
    (S-Depict P A (Orientation ?orientation) ?p-ori ?picture)
  :Subsidiary-Acts
    (Achieve P (BMB P A (Identifiable A ?x ?px ?picture))
             ?mode))
```

Here, variable names begin with a question mark (?), `(BMB S U x)` can be read as "S and U have the mutual belief x". In this example `P` stands for the presenter, the system itself in this case, `A` stands for the addressee, the user in this case. `(Bel I p)` asserts that the individual `I` believes the proposition `p`.

The letter G at the end of the header of the operator, means that this operator will be rendered in the graphical modality.

In the example, the strategy's header is an act to describe the orientation of some object by means of the graphical medium. The effect of the strategy is that both P and A will have the mutual belief that the object ?x has the orientation ?orientation. The only condition to apply this strategy is that the presenter P, in fact, believes that the object has such an orientation. To accomplish the goal the system must rely on one main act, and one subsidiary act. Here, the main act is a surface act, i.e., an act which will be passed directly to a medium realization component, the graphics realization component in this case. The subsidiary act is, in this case, a complex goal: to establish the mutual belief that the image, ?px, of the object, ?x, is identifiable by the addressee, A, in the context of the whole ?picture. To reach this goal, the system performs a weighted search among its defined strategies and picks up the best one whose header or effect matches the intended goal[10]. In the example presented, the selected strategy could be in the form:

```
(def-presentation-strategy
  :Header (Background P A ?x ?px ?picture G)
  :Effect (BMB P A (Identifiable A ?x ?px ?picture))
  :Applicability-Conditions
    (AND (Bel P (Image-of ?x ?px ?picture))
         (Bel P (Perceptually-Accessible A ?x))
         (Bel P (Part-of ?x ?z)))
  :Main-Acts (S-Depict P A (Object ?z) ?pz ?picture)
  :Subsidiary-Acts
    (Achieve P (BMB P A (Identifiable A ?z ?pz ?picture))
             ?mode))
```

In this way, both, main and subsidiary acts can be decomposed into other main and subsidiary acts. As this branching out process is carried out, modality allocation is automatically performed as well, since this decomposition eventually leads to surface acts, prefixed with 'S-' in definitions of the strategies, which will be directly realized as text or graphics.

To succeed, a surface act requires the application knowledge, and the basic ontology, both written in RAT, to be consulted. For instance, to achieve the following surface act,

```
(S-Request P A
```

[10]The system tries first the header, and, if it does not match the goal, proceeds with the effect field.

```
(Action Put)
(Agent (The A User A))
(Object (The Cup-1 (Cup Cup-1)))
(L-Goal Region-1
  (Under (The Wateroutlet-1
             (Wateroutlet Wateroutlet-1)))))
```

the system has to consult and apply the atomic action

```
put-cup-under-water-outlet =
    ⟨(agent:person ⊓ object:cup ⊓ machine:expresso-machine),
     (object.position ≐↓ agent.has-hand.inside-region),
     (object.position ≐↓
            machine.has-water-outlet.under-region)⟩
```

which is stored in the basic ontology of the system in the RAT format. An atomic action like this is a triplet ⟨*pars, pre, post*⟩. Here *pars* is a conjunction of restrictions on feature atoms. In the example, the *pars* element is `(agent:person ⊓ object:cup ⊓ machine:expresso-machine)`. This indicates that the feature `agent` is restricted to be a `person`, the feature `object` is restricted to be a `cup`, and the feature `machine` is restricted to be an `expresso-machine`. *pre* and *post* are the precondition and the postcondition of the action respectively. These two elements are conjunctions of feature restrictions, agreements ($p \overset{\downarrow}{\doteq} q$), and disagreements ($p \overset{\downarrow}{\neq} q$). In the example, for instance, the precondition `(object.position ≐↓ agent.has-hand.inside-region)`, indicates that there must be an agreement between `object.position` and `agent.has-hand.inside-region` for this action to be applied. The final condition triggered by the successful application of this action is the postcondition `(object.position ≐↓ machine.has-water-outlet.under-region)`, which indicates that after applying this action, the object (i.e., the cup) will be located under the water-outlet of the expresso machine.

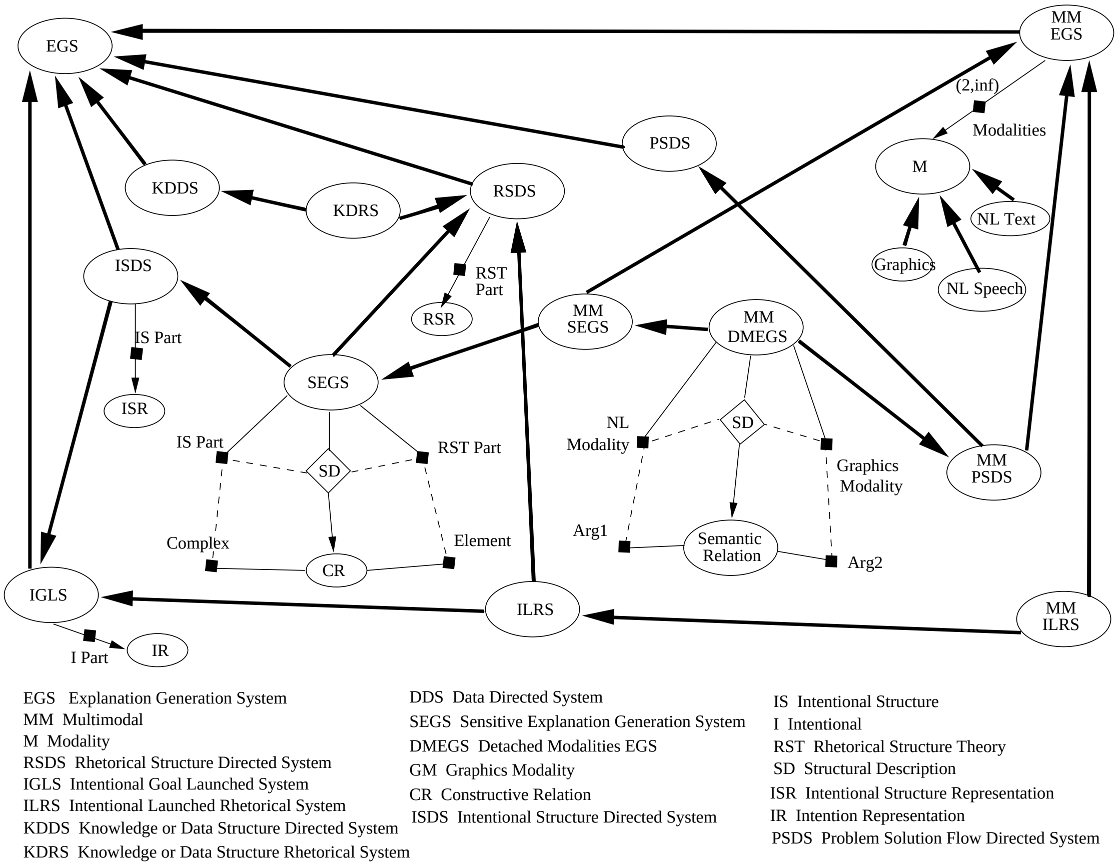


Figure 2.23: A taxonomy for explanation systems

2.6 The Hole in the Net

"Once again, the kingdom of heaven
is like a net that was let down into the lake
and caught all kinds of fish."
Matthew 13,47

In the preceding section several multimodal explanation generation systems have been reviewed. No doubt, the graphical modality introduced into these systems greatly enhances the quality of the generated explanations. Nevertheless, the immediate question that arises is whether the added graphical modality helps the systems to expand the range of interpretations of the questions they currently can answer, and even incorporate new questions. Could the surveyed systems answer a different type of question than those dealt with by the other non multimodal explanation systems, say, for instance, the Moore's PEA system? The basic questions these systems answer are *how a task is performed*, and *what* *an object is*. To be fair, it must be said that Maybury's system and the WIP system could reach the functionality exhibited by Moore's PEA system, since their plan operators took into account the same considerations as Moore's work. If the functionality of these systems was extended as mentioned, the resulting systems would fall into a new category named *multimodal sensitive explanation generation system* in Fig. 2.23. The main issue, however, still remains: is it possible for the graphical modality to contribute to the explanation generation systems in such a way that new types of questions could be answered, and those questions, dealt with by current explanation systems, could be answered from a new perspective? To answer this questions, a simple example will be considered.

Suppose that several domain objects stand in a graphical relation that could be judged as circumstantial by most multimodal explanation systems. Consider, for instance, the situation depicted in Fig. 2.24. In this figure, the black boxes represent dots that are defined in the knowledge base, while the white boxes represent dots that have emerged from the geometrical configuration of the participating objects. For instance, dot 13 is the result of the intersection of two lines. Now, in this situation, the user, who does not know what information in the depiction is not in the knowledge base, wants to ask the system: *What is the nearest dot to the black ball within a radius of 50 units? and how did you get to that conclusion?* If the system is relying only on what it has represented in its knowledge-base it would have answered that *the dot 1 is the nearest to the black ball*. The expected answer, however, would identify the searched dot with dot 15, and would explain the search process which involves the computation of the intersection between two lines. Many questions like this can be formulated, and to answer them a new sort of system needs to be introduced, namely, one that gives to the graphical modality a more active semantic role, and not just a presentational role. Some preliminary steps in this direction can be found in the works by Pineda (1989), Pineda (1992*b*), Pineda (1993), Santana & Pineda (1995), Santana et al. (1995), Pineda & Garza (1997), and Santana et al. (1998)[11]. In Fig. 2.23, this new category of system is named *multimodal*

[11] Pineda (1989) worked mainly in the solution of geometrical problems in a multimodal environment. This work, however, did not reach the stage of producing combined graphical and natural language explanations.

Pineda & Garza (1997), and more recently Pineda & Garza (1998), have tackled a specific problem of multimodal systems, namely: the resolution of multimodal references, chiefly aimed at interpretation (as opposed to generation) of natural language and graphics. For their work they presupposed some results, dealing with quantified expressions, fully developed by the author of this thesis [Santana 1995, Santana 1996], and summarised in Section 3.4 (p. 104) of this thesis. Some of the representational devices developed by Pineda & Garza (1998) following the lines of Montague (1974*d*), are used in the definition of the languages defined in Chapter 3 of this thesis, though with the differences pointed out elsewhere (see Section 3.5, p. 113). To this date, however, there is no implementation available of these Pineda & Garza's works.

The works [Santana & Pineda 1995], [Santana et al. 1995], and [Santana et al. 1998], constitute the initial proposals and setting for this thesis, and were fully developed by J. S. Santana,

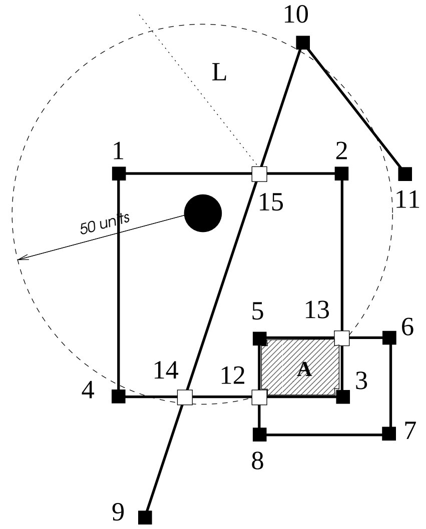


Figure 2.24: A graphical configuration

detached modalities explanation generation system. In this category, the natural language modality and the graphical modality stand in a semantic relation, such that the semantic contributions of each can be exploited by the explanation generation system. The importance of this category should not be diminished, since it opens the possibility of answering a wider range of questions. Just to mention an example, in the case of the WIP system domain, a question as simple as *"Is the switch of the expresso machine on or off?"* could not be properly answered, without the resource of explicitly stating this fact in the knowledge-base, if there is not an active semantic role of the graphical modality.

In general, many of the questions that come out from design environments cannot be answered without having a proper account of the objects emerging from the graphical configurations. Consider, for instance, the following dialogue in relation to Fig. 2.24:

User: *(Pointing at where the label A is in the figure)* Is this area suitable for a bedroom?

System: No.

User: Why?

System: Because the building norm states that bedrooms must have at least 4 units by 3, and this area has only 3 units by 2.

In order for the system to answer in this way, it must be aware that: (i) the user intends to refer to the shaded (*emergent*) rectangle and not to any of the

and reviewed by L.A. Pineda, and S. Vadera.

two other rectangles defined in the knowledge base, and (ii) the just referred graphical object could be identified with the natural language concept *bedroom*, thus connecting the graphical semantics with the natural language semantics.

Swartout (1990) and Moore (1995) have identified *extensibility* as one of the requirements for interactive explanation systems. They define the extensibility of a system as the possibility to *easily* extend the explanation facility to respond to additional question types, which may not be foreseen at design time, or to add new strategies for answering existing question types. They observed that the satisfaction of this requirement depends not only on what is represented in the knowledge base, but on making available the *design rationale*[12] behind the expert system to the explanation facility [Swartout et al. 1991]. The above dialogue shows that to make explanation systems interact in design environments, i.e., to make them able to answer the sorts of question that the design tasks require, they must additionally have a mechanism to capture the relationship between the graphical and the linguistic semantics.

The purpose of the forthcoming chapters is to propose the representational structures that exploit the semantics of the graphical modality in the same manner as that of the natural language, and show how these structures can be used to produce responses to questions of the sorts mentioned above, in the context of design environments. Put another way, the purpose and the contribution of this thesis is to create an instance of the node named *multimodal detached modalities explanation* ***generation*** *system* in Fig. 2.23, which currently does not exist in the literature. As a proof of concept, some of the key ideas in this thesis have been tried out in a prototype system developed by the author, described in Chapter 6.

[12]A *design rationale* of an expert system can be understood as a record that comprises the reasons underlying each particular decision in the expert system's design.

Chapter 3

A LOGICAL REPRESENTATION FOR GRAPHICAL AND LINGUISTIC INFORMATION

Chapter 2 presented a survey of previous work in the area of explanation systems and motivated the need for developing a logical representation for both the linguistic and graphical modalities. To enable the analysis and the development of automatic language processing procedures, a language needs to have a well defined syntax and semantics. This chapter develops the syntax and semantics for a logical language and a natural language that used to represent the modalities utilised in this work. First, the chapter begins with an example that aims to illustrate the kind of objects and processes that are to be represented by these languages. Then Sections 3.1 and 3.2 are devoted to developing the languages and their semantics. The translation between both languages is examined in Section 3.2.3. Section 3.3 addresses a significant research problem raised in Chapters 1 and 2: How can the languages be related, or coordinated, and utilised to represent a state of interpretation? Section 3.4 tackles a specific problem in the area of geometrical design problems, known as the treatment of emergent objects.

An example situation, taken from the field of the geometrical design of overhead transmission line towers, is shown in Fig. 3.1(a). This figure represents a stage in the design of a certain tower. Here, the circle represents the cable dielectric protection area, which none of the metallic elements of the tower must invade. For the sake of clarity, this example will be subjected only to this design rule, which is explicitly enunciated here for future reference as follows:

Design rule: *Every metallic bar of the tower must lie outside the dielectric area.*

Suppose that, during the design interaction, the user displaces a part of the

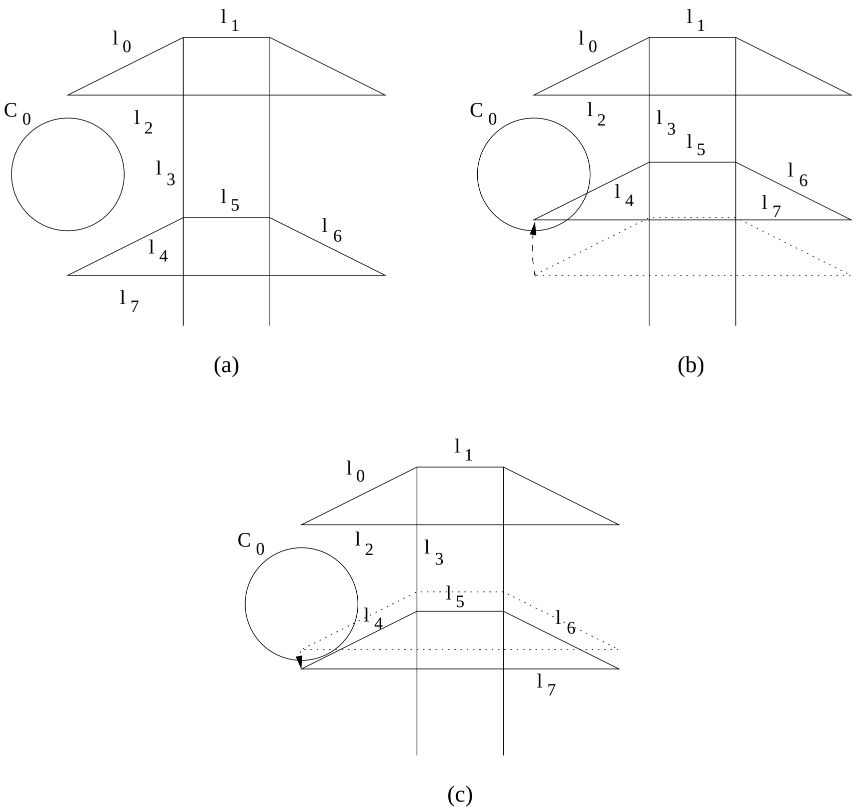


Figure 3.1: Stages in a tower design process

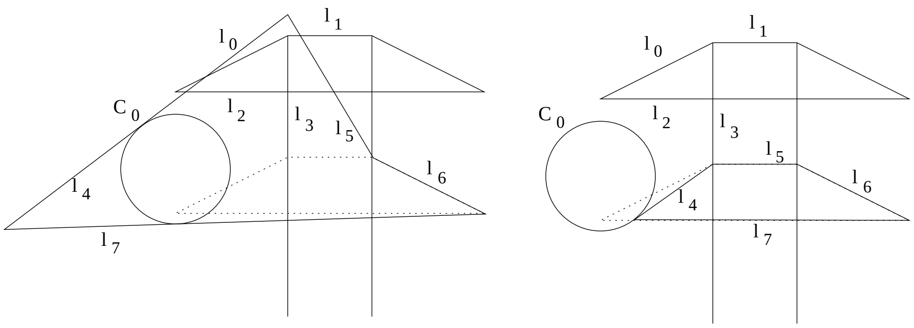
tower, specifically, a crossarm[1] as shown in Fig. 3.1(b). In this new situation, the design rule has been broken, since there are some bars of the tower invading the dielectric protection area.

The problem that both the system and the user have to solve cooperatively can roughly be described as producing a new situation in which none of the design rules is broken while the user's intentions are still satisfied. In this case, once the system has notified the user about the rule violation, either the user or the system could produce a configuration like the one shown in Fig. 3.1(c). Here, the design rule has been satisfied and the new configuration sensibly *fits the user intentions*.

The whole process represented in Fig. 3.1 can be viewed as a path through three different states. Each depiction in the figure represents a state in the tower design process, or, in other words, a state in the problem-solving process. Hence, the problem-solving process that this work deals with, is one that involves changes in discrete steps (states) in time.

This problem can be classified into a class known as geometrical constraint satisfaction problems. In the above example, the design rule about the dielectric protection area can be translated into a geometrical constraint stating that all lines must lie outside the inner region delimited by the circle. There can be several other kinds of geometrical constraints such as minimum or maximum distances between points, minimum or maximum angles, symmetry, etc. Several approaches have been tried to solve problems of this nature [Sutherland 1963, Steele 1980, Leler 1988, Cohen 1990, Borning 1981, Borning 1986, Colmerauer 1990, Puget 1994]. Most of these approaches formulate equation systems that are solved by diverse methods, when it is possible, or simplified by some sort of symbolic manipulation. Either, the solution of such an equation system or the simplified version of it is then presented to the user. Typically, this sort of problem does not have a unique solution. In fact, when dealing with real numbers, the solution space could be infinite. To select one solution from that space, some of the common practices are the use of mathematical programming techniques, the use of iterative approximation techniques, and evaluations based on local propagation techniques [Güsgen 1989, Leler 1988, Colmerauer 1990]. Though satisfying the problem constraints, the selected solution, however, may not satisfy the user's expectations. Consider, for instance, the two solutions to the problem of Fig. 3.1(b) presented in Fig. 3.2. Given the dielectric area constraint, the two configurations presented in the figure are in the solution space of the problem. From the towers design point of view, however, neither of them is a satisfactory solution. It could be argued that the problem was underspecified, since there are at least two constraints missing, namely: that the crossarm

[1] A crossarm is a trapezoidal section like the one shown in the diagram.



(a) (b)

Figure 3.2: Senseless solutions to a geometrical design problem

must be symmetric, and that the two crossarms must be identical. Nevertheless, to some degree most design problems are underspecified, since a number of the constraints are subjective, i.e., they do not obey a design norm and are subject to a particular designer's taste and common sense. Furthermore, for a user, though well trained in engineering design problems, to put constraints of this sort in the form of mathematical equations is a very hard task reserved only for those experienced in mathematical formulations. Additionally, since most of these constraints are non-linear, the complexity of the problem is greatly increased with any additional new constraint introduced.

From the explanation generation point of view, since the methods used by the mentioned systems (e.g., [Steele 1980, Leler 1988, Cohen 1990, Borning 1986, Güsgen 1989, Puget 1994]) do not explicitly take into account the designers' intentions, the only explanation generation possibility for these systems is to attach templates to their code in order to produce explanations that justify their actions. Consider, for instance, a simplified version of the problem posed in Fig. 3.1, that is shown in Fig. 3.3(a). This new version of the problem consists of just one line, L, and one circle, C, and the problem can be stated as finding a configuration in which the line is outside the circle. Using mathematical programming, under the criterion of producing the minimum displacements, the problem could be defined as minimising the objective function[2]

$$S(\vec{D}) = \sum_{i=0}^{2} \Delta^2 x_i + \Delta^2 y_i$$

subject to the condition

outside(L,C)

where

[2] This example was adapted from the thesis by Santana (1987). Formulations of this sort can be found in many of the systems devoted to solve this kind of problem: see, for instance, the ILOG system documentation [ILOG 1997, Puget 1994].

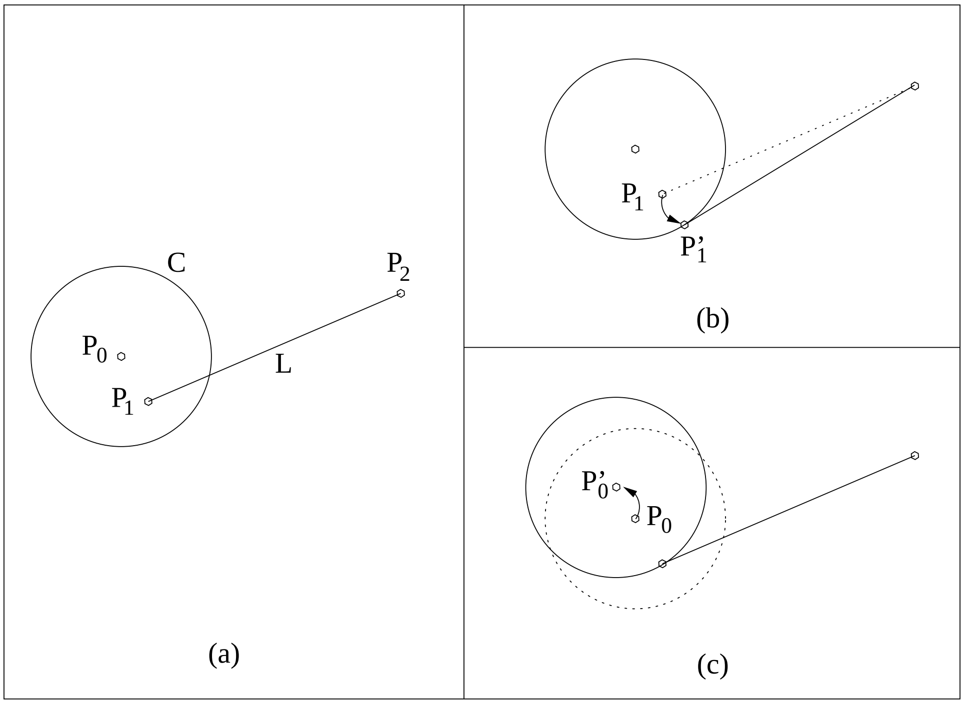


Figure 3.3: Two solutions to a simple constraint satisfaction problem

$$\vec{D} = \langle \Delta x_0, \Delta y_0, \Delta x_1, \Delta y_1, \Delta x_2, \Delta y_2 \rangle$$

$$\mathrm{P}_i = \langle x_i, y_i \rangle \ \mathrm{P}'_i = \langle x'_i, y'_i \rangle$$

$$\Delta w = w' - w$$

$$\mathrm{L} = \mathrm{line}(\mathrm{P}_1, \mathrm{P}_2)$$

and outside(*line, circle*) is a function that returns **true** if the given line segment lies outside the given circle, and **false** otherwise, and w' denotes the new value of any variable w in the final solution state. Note that the dots, P'_i, in the solution state can be computed using the individual values of the *coordinates increment vector*, $\vec{D}$.

Common mathematical programming algorithms will yield any of the two solutions depicted in Fig. 3.3(b) and (c), though there are an infinite number of solutions produced by simultaneously displacing both the extreme P_1 of the line and the centre of the circle, and a wider range of solutions is obtained if the objective function is not used[3]. Regarding the explanation generation, because

[3] In fact, when this problem is expressed and solved using a constraint logic programming language (CLP), one would finish with the intervals of real values that the variables could take

of the selected criterion, the explanation that could be attached to the problem solving strategy would say something like:

> *The dot P_0 (or P_1) was moved to its new position because moving it produced a minimum displacement of dots, and to avoid the line L being inside the circle C.*[4]

Provided that the two solutions, Fig. 3.3(b) and (c), could be made accessible to the end-user, e.g., by a back-tracking mechanism, this same explanation would be produced no matter which one of the two solutions is presented[5]. From the design point of view, however, these two solutions might be completely different: in one case, the slope of the line changed, Fig. 3.3(b), whereas in the other, the whole circle was translated to a new position, Fig. 3.3(c). These distinctions would make preferable one solution over the other. However, there are no cues on how to associate, in the code of the algorithm, any appropriate explanation to each case.

Summing up, trying to attach any explanation facility to geometrical constraint systems based on mathematical programming techniques or CLP languages, like the ones mentioned above, can be a painful task that might end in the production of explanations that are nonsense or difficult to understand. For this reason, the problem-solving approach followed in this thesis is inspired in the work developed by Pineda (1992*a*), and by Pineda et al. (1994). In this approach, constraint satisfaction problems are modelled symbolically rather than numerically. A drawing is described by a set of linguistic terms, where graphical objects are defined and manipulated as symbols of a representational language. The linguistic terms act like names or descriptions of the graphical symbols constituting the drawing. Constraints, on the other hand, are represented by sentences of such a language. To solve a problem, there is a set of plan operators that mimic the things a human user would do to find its solution. The operators are written in a meaning representational Prolog-like language specially

[Older & Vellino 1993, Older & Benhamou 1993], or with a reduced set of symbolic relations, simpler than the original set expressing the problem [Colmerauer 1990, Jaffar & Michaylov 1987]. In this way, these systems provide an insight into the problem solution space. Although very useful in many applications, these approaches leave at least two issues open, namely: (a) how to express the problem in a CLP language in a proper way to obtain the relevant answers, and (b) how to select from the problem solution space just one adequate solution.

[4]An even more tedious explanation would include details about the used method, e.g., *"to solve the problem, both the objective function and the constraints were linearised and then the simplex method was used ...."* As Moore (1995) observed, cluttering explanations with **implementation** details, like these, might obscure the underlying domain-related reasoning, and the resulting explanations might be more confusing than enlightening to the end-users (see also the article by Swartout (1983)).

[5]Furthermore, the selection of the solution does not depend on any intentional consideration, but rather on implementational grounds such as the order in which the equations are written or how the selected algorithm operates on them, etc.

designed to treat geometrical constraints. Since they are conceived to embody designer or draftsman concepts, the operators descriptions are tightly coupled to human designers or draftsmen methods of solving problems, and, since every operator includes its effects in terms of the cognitive state of the interlocutors, each step can be comprehensively explained to the end-users.

To solve problems of this kind, a system must rely on an adequate representation of the things that happen in the world. A model-theoretic approach, based on the Montague semantic theory [Dowty et al. 1981, Montague 1974*b*], has been chosen, since the proposed representation is not only aimed at solving the problem itself, but also at explaining its solution.

3.1 General Framework

A language, which is the device used to explain something, can be considered as an enumerable set of symbols and a set of rules to construct well-formed expressions. According to Montague, the interpretation of these expressions is relative to a formal model in the logical sense [Dowty et al. 1981]. In this section, the general framework, within which the formal models used in this thesis are defined, is presented.

In this approach, a model for a language, $\mathcal{L}$, is a pair $\langle A, \mathcal{F} \rangle$, where A is a non-empty set of the individuals in the universe of discourse, and $\mathcal{F}$ is a *interpretation function* that assigns to each non-logical constant of the language an object of an appropriate semantic type [6]. In general, the denotations are taken from domains constructed from the set A and the set $J = \{0, 1\}$, whose members, 0 and 1, are taken conventionally to represent the falsity or truth of propositions respectively. The semantic value of composite expressions of the language is built up from the semantic values of its constituent subexpressions following the compositionality principle of Frege [Dowty et al. 1981].

Several models are defined to represent and solve the sort of problems addressed at the beginning of this chapter. The general framework within which these models are defined is schematised in Fig. 3.4. The elements of this sketch are as follows:

1. L represents the set of natural language expressions, e.g., *the bar is painted with anticorrosive paint*, *the intersection between* bar_3 *and* bar_2, *a crossarm*, etc. Fig. 3.4 shows only some constants of the language destined to name individuals.

2. G represents the set of expressions of an intermediate logical language. Examples of expressions of this language are: c_0, $outside_*(c_1)(l_3)$, $\forall x[line(x) \rightarrow$

[6] This concept of a model will be extended later to consider several states of interpretation.

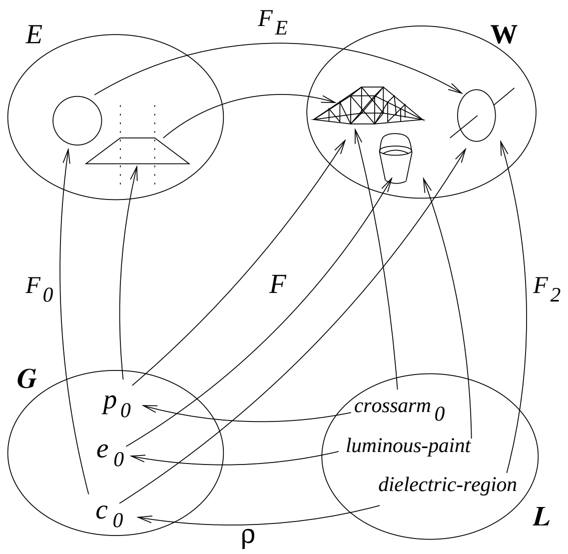


Figure 3.4: Languages and models

vertical(x)], etc. The figure, shows only some constants that will be related to individuals.

3. E represents a domain of geometrical objects with spatial extension (or a *geometrical world*), but taken in abstraction from any material realisation: they are the objects of the world of geometry. To this domain belong objects such as *dots*, *circles*, *lines*, *polygons*, etc. Objects of this domain have geometrical properties like *area*, *length*, *position*, *diameter*, etc., depending on their type[7].

4. W represents the world, with *real* objects such as *bars*, *crossarms*, *dielectric areas*, *paints*, *joints*[8], etc.

5. F_0 is an interpretation function that assigns to **a** constant of G an object of the appropriate semantic type built up from elements of E. Some constants of G map to individuals of E, but others, such as predicates and functions, map to objects of other types as stated in the formal definition of the possible denotations of expressions of G, which includes constants, given in Section 3.2.2.3 (p. 82). The figure shows that constant c_0 maps to a circle, and p_0 maps to a trapezium in E under this function. More complex mappings are not shown in the figure. For instance, one place predicates map to characteristic functions (or to subsets of E), constants destined to name functions map precisely to functions, etc. Note that F_0 is a partial function, i.e., it is undefined for some constants of G. For instance, the figure shows that constant e_0 does not have any image under F_0. Thus, there

are some constants and expressions of G that do not have any geometrical interpretation.

6. F is an interpretation function that assigns to **each** constant of G an object of the appropriate semantic type built up from elements of W. Fig. 3.4 shows that, under this function, p_0 maps to a *crossarm*, c_0 to a *dielectric area*, and e_0 to *luminous paint*, represented by the symbol  in the figure. More complex mappings, such as predicates and functions, are not shown in the figure. Again, one place predicates map to characteristic functions (or subsets of W), and constants for functions to functions, etc.

7. F_E is a representation function that assigns to each object of E, the individual object of W that it represents. Note that this function establishes the relationship between the *geometrical world* and the world: the mappings here are from objects in one world to objects in the other world.

8. ρ is the translation relation from L to G. This relation establishes a correspondence between the symbols of one language and the symbols of the other language. For instance, Fig. 3.4 shows that, under this relation, $crossarm_0$ in L corresponds to p_0 in G. This relation includes not only constants but even expressions of both languages. Since the figure is schematic, translations between expressions are not shown, though they are formally defined below. For example, the natural language sentence *bar_5 is horizontal* translates into the expression $horizontal(l_5)$ of G. For simplicity, in this thesis the relation ρ is considered as a function from L into G.

9. F_2 is an interpretation function that assigns to each constant of L an individual of W. In this work, this function is defined indirectly with the aid of the translation relation ρ and the interpretation function F, since, for all constants of L, $F_2 = F \circ \rho$.

Now, from the point of view of this thesis, there are three interesting models to consider (see Fig. 3.4):

1. the model of language G relative to the geometrical domain: $M_0 = \langle E, F_0 \rangle$,

2. the model of language G relative to the *world*: $M_G = \langle W, F \rangle$, and

3. the model of language L relative to the *world*: $M_L = \langle W, F_2 \rangle$, which is indirectly established by means of models M_0 and M_G, and the translation relation ρ between L and G.

[7] A more formal description of E can be found in Section 3.2.2.3 (p. 81).

[8] Take note that the segment of the world of interest for this work is that related to the design of electric transmission towers.

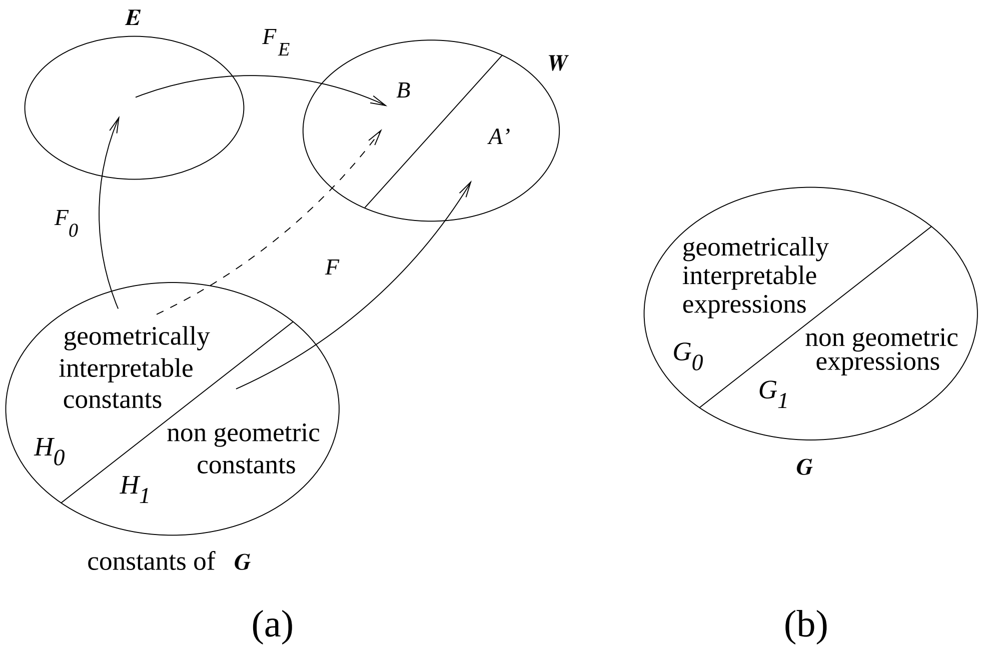


Figure 3.5: Interpretations and partitions of G

The general framework outlined above is multimodal since the interaction between the natural language (text) and the graphical modalities is taken into account according to the following considerations:

(a) The constants of G are partitioned into two subsets: those that have a geometrical interpretation, H_0, and those that do not have it, H_1, as shown in Fig. 3.5(a). Consequently, expressions of G built up exclusively from geometrically interpretable constants are geometrically interpretable as well. For instance, provided that the function F_0 is defined for the constants: *the_dot*, 3.20, -1.0, $inside_*$, c_1, and l_2, i.e., they are geometrically interpretable, the expressions $the\_dot(3.20)(-1.0)$, and $inside_*(c_1)(l_2)$ of G are geometrically interpretable. Conversely, expressions of G built up from constants of H_1 or from a combination of constants of H_1 and H_0, are not geometrically interpretable. For instance, provided that F_0 is undefined for the constants $painted\_with_*$, and e_0, i.e., they belong to H_1, the expression $painted\_with_*(e_0)(l_2)$ of G cannot be geometrically interpreted, even though l_2 has a geometrical interpretation. Accordingly, the expressions of G are partitioned into two categories, those that have a geometrical interpretation, G_0, and those that do not have it, G_1, as shown in Fig. 3.5(b). Note that $H_0 \subset G_0$ and $H_1 \subset G_1$.

(b) Since E and W are abstract, in this work the notion of *theory* in the logical sense [Boolos & Jeffrey 1989, p. 106] is utilised to specify the situation in

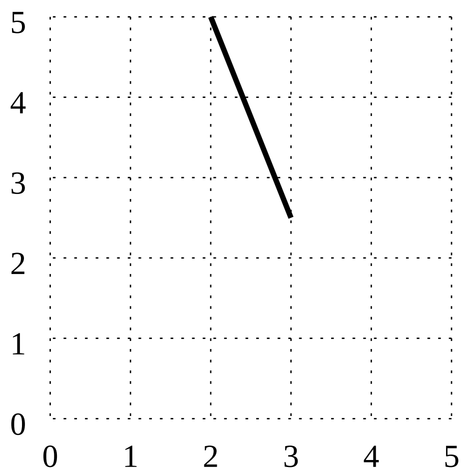


Figure 3.6: A graphical representation of l_1

both E and W. Let T_0 be a subset of *propositions* of G_0 which is selected to be the set of axioms of a theory. This theory is called the *graphical theory*, since it contains propositions that are geometrically interpretable, and they specify the situation in the geometrical domain E. For instance, a very simple graphical theory may consist of the single proposition $l_1 = the\_line(the\_dot(3)(2.50))(the\_dot(2)(5))$, which says that l_1 is equal to a segment of line that goes from the dot (3, 2.50) to the dot (2, 5). If the set of axioms of the graphical theory is consistent and true, some or all its propositions could be represented graphically. For instance, the previous proposition can be graphically represented as shown in Fig. 3.6. By depicting representations of this sort on any graphical medium, the *graphical modality* is enabled within the framework presented here. In fact, since this is the manner to specify E, the graphical representation of all axioms in T_0 is isomorphic to E.

(c) In a similar way, let T_1 be a subset of propositions of G_1 which is used to represent the set of axioms of a *non-graphical theory*. By these means, the part of the situation of the world W that is not geometrically specifiable is specified. For instance, the proposition $painted\_with(e_0)(l_1)$ might be the only axiom of the non-graphical theory. The union of the graphical theory and the non-graphical theory is called *the linguistic theory*: $T_G = T_0 \cup T_1$. Some of the propositions of the linguistic theory can be translated into the *natural language L*. For instance, provided that e_0 is translated into L as *luminous-paint*, and l_1 as bar_1, the translation of the proposition $painted\_with(e_0)(l_1)$ into L is given by the text: bar_1 *is painted with luminous paint*. The display of texts like these on a screen of a system or the output of them as a synthesised voice in a sound device of a system constitute the *linguistic modality* within the framework presented here.

(d) Within this framework, other propositions and expressions of G, although they are not part of the axioms of neither the graphical theory nor the non-graphical theory, can also be displayed, and even introduced, in the appropriate modality. Suppose, for instance, that someone wants to ask a system built according to this framework, whether bar_1 is horizontal in the context of the graphical and non-graphical theories and translations exemplified in clauses **(b)** and **(c)**. Then, she or he can utter or type in the *natural language* sentence (of L): *is* bar_1 *horizontal*?, which is in turn translated into G as the proposition $horizontal(l_1)$, and then evaluated semantically according to T_G to give 0 (false) in this example (see Fig. 3.6).

To illustrate how the multimodal interpretation process proceeds, in Section 3.3.1 (p. 100) two examples of multimodal interpretation are presented. The first example, which involves two quantified terms, illustrates how the translation of the natural language sentence into the logical language G along with the information in the graphical modality is used to compute the semantic value of the original natural language sentence. The second example illustrates how a natural language sentence is evaluated using the non-graphical information.

General remarks about the framework

Although in the above discussion it has been mentioned that language L is indirectly interpreted, it is worth stressing here that this work adopts an approach that follows the so called Montague's *indirect via* for the interpretation of natural language. This strategy was introduced by Montague in his paper *The Proper Treatment of Quantification in Ordinary English* (PTQ) [Montague 1974*d*] and contrasts with his earlier development in his paper *English as a Formal Language* [Montague 1974*a*], in which a direct model-theoretic interpretation of the natural language is given. In PTQ, each English expression is first translated into an expression of intensional logic [Montague 1974*c*, Dowty et al. 1981], and then the given model-theoretic interpretation of the intensional logic expression serves indirectly as the interpretation of the English expression.

In the work presented in this thesis, the language L is a fragment of English, whose syntax is specified by production rules conforming to the PATR formalism [Shieber 1986, Gazdar & Mellish 1989]. Following the compositionality principle embodied in the formal definition of the translation relation ρ, that takes into account the phrase structure imposed by the syntactic rules, each expression of L is translated into an expression of the logical language G. The translated expression is then interpreted, and this model-theoretic interpretation serves indirectly as the interpretation of the original expression of L.

In relation to Montague's developments in PTQ, it has to be emphasised that in this thesis the logical language G is interpreted according to either of two models, namely, $M_0 = \langle E, F_0 \rangle$, or $M_G = \langle W, F \rangle$, depending on *the nature* of the expression of G that is being interpreted. Since M_G includes M_0, it can be said that in general G is interpreted with respect to model M_G. However, to make the distinction between modalities clear, the geometrical properties of the world are separated by defining a *geometrical world E* and the corresponding geometrical model M_0.

Since the languages studied here require us to consider different states of interpretation, the concept of *model for a language* is extended as follows.

A model for a language, $\mathcal{L}$, is, an ordered tuple $\langle A, I, \mathcal{F} \rangle$, where A and $\mathcal{F}$ are as defined above, and I is a linearly ordered set of states $i_1, i_2, ..., i_n$. The function $\mathcal{F}$ must contemplate the fact that from state to state the interpretation of the constants of the language might change. Thus, the notation $[\mathcal{F}(\alpha)](i)$ is introduced to mean the value of $[\mathcal{F}(\alpha)]$ in state i.

In the forthcoming sections, the framework introduced here is formally defined, the usefulness of it is illustrated by examples of interpretation of natural language sentences in a multimodal context, and, finally, the problem of emergent objects is presented and a solution to this problem is proposed.

3.2 Formal Definition of the Multimodal System for Representation and Interpretation

In this section, the natural language L, and the logical language G are defined through the presentation of their syntax and semantic interpretation, and the translation between both languages is defined.

3.2.1 The language L

The language L will be designed to produce sentences such as *bar$_5$ is painted with anticorrosive paint, some bars are horizontal,* and *the intersection between bar$_2$ and bar$_1$ lies outside every dielectric area* in a compositional fashion. Therefore, all constants and subexpressions of the previous sentences can also be produced. This section defines this language using Montague semantics notation as defined by Dowty et al. (1981).

3.2.1.1 Lexical elements

The lexical elements of a language constitute its vocabulary. They are the basic symbols used to compose well-formed expressions of the language. To handle

Constants	Category name	Category definition
dot_1, dot_2, ... $joint_1$, $joint_2$, ... bar_1, bar_2, ... $crossarm_1$, $crossarm_2$, ... $dielectric\text{-}area_1$, $dielectric\text{-}area_2$, ... *luminous-paint*, *anticorrosive-paint*	T	t/IV
dot, intersection, bar, crossarm, dielectric-area, paint	CN	CN
horizontal, vertical, painted, parallel, perpendicular	ADJ	ADJ
lie-inside, lie-outside, be	TV	IV/(t/IV)
be	IV/ADJ	IV/ADJ
a, the, some, every	T/CN	(t/IV)/CN
	PP	CN/CN
	PP'	ADJ/ADJ
	IV	IV

Table 3.1: Constants of language *L*

these symbols within the syntactic rules, they are grouped into syntactic categories. Each category is a set which contains the symbols associated to objects of a determinate type. The set of syntactic categories of *L* is as follows:

1. The basic syntactic categories of *L* are t, IV, ADJ, and CN where t is the category of sentences, IV is the category of intransitive verbs, ADJ is the category of adjectives, and CN is the category of common nouns.

2. If A and B are syntactic categories then A/B is a category[9]

Traditional syntactic categories of natural language such as transitive verbs (TV), terms (T), prepositional phrases (PP), and determiners (T/CN) can be derived from the basic categories.

Table 3.1 shows the constants of *L*, their syntactic categories, and the definitions of these categories in terms of more basic ones.

[9]An expression of category A/B is an expression of category A from which an expression of category B is missing.

3.2.1.2 Syntax of L

The syntactic definition of a language establishes the rules for the construction of its well formed expressions. To describe these rules in this work the PATR formalism [Shieber 1986, Gazdar & Mellish 1989] has been used. A brief description of this formalism is given in the Appendix A of this thesis. The well-formed expressions of the language L are then given by the following rules[10]:

S1$_L$ Rule { Sentences }
t $\rightarrow$ T IV.

Examples:

- *joint*$_2$ *is an intersection*
- *bar*$_5$ *is painted with anticorrosive paint*
- *some bars are horizontal*
- *the intersection between bar*$_2$ *and bar*$_1$ *lies outside every dielectric area*

S2$_L$ Rule { Transitive verb phrases }
IV $\rightarrow$ TV T.

Examples:

- *be an intersection*
- *lie outside every dielectric area*
- *lie inside dielectric area*$_2$

S3$_L$ Rule { Attributive verb phrases }
IV $\rightarrow$ IV/ADJ ADJ.

Examples:

- *be horizontal*

[10] Though conforming to PATR style, the rules presented here take the appearance of simple context free phrase structure grammar rules. In this presentation, syntactic agreements (e.g., person, number, and gender) between syntactic components are omitted for simplicity. In the actual expanded grammar developed in Appendix A, these agreements are achieved by the feature system provided by the PATR formalism.

- *be painted with anticorrosive paint*
- *be perpendicular to the bar*

S4$_L$ Rule { Terms }
T $\rightarrow$ T/CN CN.

Examples:

- *an intersection*
- *some bars*
- *the intersection between* bar_2 *and* bar_1
- *every dielectric area*

S5$_L$ Rule { Common nouns }
CN $\rightarrow$ CN PP.

Examples:

- *intersection between* bar_2 *and* bar_1

S6$_L$ Rule { Compound adjectives }
ADJ $\rightarrow$ ADJ PP'.

Examples:

- *painted with anticorrosive paint*
- *perpendicular to the bar*

S7$_L$ Rule { "Between" prepositional phrases }
PP $\rightarrow$ between T and T.

Examples:

- *between* bar_2 *and* bar_1

S8$_L$ Rule { "With" prepositional phrases }
PP' $\rightarrow$ with T.

Examples:

- *with anticorrosive paint*
- *with luminous paint*

S9$_L$ Rule { “To” prepositional phrases }
PP’ $\rightarrow$ to T.

Examples:

- *to the bar*
- *to* bar_5

3.2.1.3 Semantics of L

The semantics of a language establishes the interpretation of its well-formed expressions. The first step in the interpretation of a language is to determine the semantic values of the symbols of the language. This is done by providing a *model* for the language.

The model $M_L = \langle W, F_2 \rangle$ can be indirectly established from the model $M_G = \langle W, F \rangle$, (see Fig. 3.4) using the translation relation ρ, from language L into G.

In this thesis, the translation relation ρ is assumed to be a function that assigns to each phrase-structure tree resulting from the syntactic analysis of a phrase of L, a valid expression of G. The semantic interpretation of this expression is taken to be the semantic interpretation of the original phrase of L, thus providing the basis for the building up of the model $M_L = \langle W, F_2 \rangle$. For example, consider the natural language constant *luminous-paint*. As shown in Fig. 3.4, its translation into G is given by $\rho(luminous\text{-}paint) = e_0$, and $F(e_0) =$, which is the abstract object in the world known as luminous paint. Indirectly, then it is concluded that $F_2(luminous\text{-}paint) =$ .

For examples on how the semantics of a natural language phrase is evaluated using this strategy, see Section 3.3.1 below (p. 100).

3.2.2 The language G

The logical language, G, will be designed as a multisorted high-order language with functional abstraction. This section defines this language using Montague semantics notation [Montague 1974*b*, Dowty et al. 1981].

3.2.2.1 Lexical elements

There are two basic types in the language, G, namely, the type *entity*, denoted by the symbol e, and the type *truth-value*, denoted by the symbol t. The type *entity* groups the language symbols for individuals like *dots, lines, arcs, circles*, *paints*, etc., while the type *truth-value* groups the language symbols denoting truth values.

A set of individuals of a given type, a, will be denoted by W_a. In particular, the set used to represent truth values is $W_t = \{0, 1\}$. Furthermore, the introduction of this set allows logical constants ($\wedge$, $\vee$, $\rightarrow$, etc.) to be treated categorically in the same way as the other predicates, i.e., they are assigned a (fixed) semantic value through the interpretation function.

Since the symbols of the language are conceived to denote not only objects, but even relations between them, a more elaborate construction of types is also required. Let S be the set of basic types: $S = \{e, t\}$. Then the set of the corresponding composite types, T, is constructed according to the following rules:

1. If $s \in S$ then $s \in T$.

2. If $a \in T$ and $b \in T$, then $\langle a, b\rangle \in T$.

3. Nothing else is in T.

In this notation, an expression of a type $\langle a, b\rangle$, where a and b are types, denotes a function with a domain in W_a and a range in W_b.

The symbols or lexical elements of the language are organised in two families of sets, as follows:

1. $C = \langle C_s\rangle_{s\in T}$, where C_s is the set of all constants of type s, and

2. $V = \langle V_s\rangle_{s\in T}$, where V_s is the set of all variables of type s.

For instance, C_e , is the set of all symbols associated with individuals, and $G_{\langle e,t\rangle}$ is the set which contains the names of one-place predicates. Some of the constants of G are shown in Table 3.2.

The constants $inside_*$, $outside_*$, $perpendicular\_to_*$, $parallel\_to_*$, $painted\_with_*$, and $intersection\_between_*$ are destined to denote geometrical functions between geometrical objects. For instance, the constant $intersection\_between_*$ will denote a function that takes two lines as arguments and gives the dot that is the intersection between these lines. Nevertheless, in order to handle uniformly the expressions of G that will be translated into *terms* of L preserving quantification properties, in parallel to these constants, the constants *inside, outside, perpendicular_to, parallel_to, painted_with,* and *intersection_between*, are introduced. These constants will admit the type raised arguments required to preserve quantification properties during the translation between G and L. For instance, *intersection_between* denotes a function that, given two terms, returns a predicate such that an individual satisfies it. A similar set of constants: *r_inside, r_outside, r_perpendicular_to, r_parallel_to,* and *r_painted_with*, is introduced to cope with a different quantification order when two quantifiers are used in a expression.

Constant	Type
any numeral, e.g., 3.1416 $b_1, b_2, \ldots, r_1, r_2, \ldots, d_1, d_2, \ldots,$ $l_1, l_2 \ldots, c_1, c_2, \ldots, p_1, p_2, \ldots$ *anticorrosive_paint, luminous_paint*	e
bool, real, gdot, line, circle, gpaint, polygon, ghorizontal, gvertical, gparallel, gintersection, gperpendicular, gpainted	$\langle e, t\rangle$
$\neg$	$\langle t, t\rangle$
$=$	$\langle e, \langle e, t\rangle\rangle$
$\wedge, \vee, \rightarrow, \leftrightarrow$	$\langle t, \langle t, t\rangle\rangle$
$inside_*$, $outside_*$, $perpendicular\_to_*$, $parallel\_to_*$, $painted\_with_*$	$\langle e, \langle e, t\rangle\rangle$
$intersection\_between_*$	$\langle e, \langle e, e\rangle\rangle$
the_dot, the_line, the_circle, path	$\langle e, \langle e, e\rangle\rangle$
inside, outside, perpendicular_to, parallel_to, painted_with, r_inside, r_ *outside, r_perpendicular_to, r_painted_with*	$\langle\langle\langle e, t\rangle, t\rangle, \langle e, t\rangle\rangle$
intersection_between	$\langle\langle\langle e, t\rangle, t\rangle, \langle\langle\langle e, t\rangle, t\rangle, \langle e, t\rangle\rangle\rangle$

Table 3.2: Constants of language G

3.2.2.2 Syntax of G

The well-formed expressions of the language G are organised in a family of sets:

$$E = \langle E_s\rangle_{s\in T}$$

where E_s is the set of well-formed expressions of type s defined by:

1. If $\alpha \in C_s$, then $\alpha \in E_s$.
2. If $\mu \in V_s$, then $\mu \in E_s$.
3. If $\tau \in E_a$, and $\phi \in E_{\langle a,b\rangle}$, then $\phi(\tau) \in E_b$.
4. If $\mu \in V_b$, and $\alpha \in E_a$, then $\lambda\mu[\alpha] \in E_{\langle b,a\rangle}$.
5. If $\mu \in V_s$, and $\beta \in E_t$, then $\exists\mu(\beta) \in E_t$.
6. If $\mu \in V_s$, and $\beta \in E_t$, then $\forall\mu(\beta) \in E_t$.

Although G is a very expressive language, not all its well-formed expressions are translatable into L. As an aid to define the translatable expressions of G into L in a clear and compact way, the abbreviations shown in Table 3.3 are used.

Abbreviation	Formal expression	Type
A	$\lambda P \lambda Q \exists x[P(x) \wedge Q(x)]$	$\langle\langle e,t\rangle , \langle\langle e,t\rangle , t\rangle\rangle$
SOME	$\lambda P \lambda Q \exists x[P(x) \wedge Q(x)]$	$\langle\langle e,t\rangle , \langle\langle e,t\rangle , t\rangle\rangle$
THE	$\lambda P \lambda Q \exists y[\forall x[P(x) \leftrightarrow x = y] \wedge Q(y)]$	$\langle\langle e,t\rangle , \langle\langle e,t\rangle , t\rangle\rangle$
EVERY	$\lambda P \lambda Q \forall x[P(x) \rightarrow Q(x)]$	$\langle\langle e,t\rangle , \langle\langle e,t\rangle , t\rangle\rangle$
BE_a	$\lambda \mathcal{P} \lambda x \mathcal{P}(\lambda y[x = y])$	$\langle\langle\langle e,t\rangle , t\rangle , \langle e,t\rangle\rangle$
BE_b	$\lambda P \lambda x P(x)$	$\langle\langle e,t\rangle , \langle e,t\rangle\rangle$
D_i	$\lambda P[P(d_i)]$	$\langle\langle e,t\rangle , t\rangle$
L_i	$\lambda P[P(l_i)]$	$\langle\langle e,t\rangle , t\rangle$
C_i	$\lambda P[P(c_i)]$	$\langle\langle e,t\rangle , t\rangle$
P_i	$\lambda Q[Q(p_i)]$	$\langle\langle e,t\rangle , t\rangle$
AP	$\lambda P[P(anticorrosive\_paint)]$	$\langle\langle e,t\rangle , t\rangle$
LP	$\lambda P[P(luminous\_paint)]$	$\langle\langle e,t\rangle , t\rangle$
BETWEEN	$\lambda x_{\langle\langle e,t\rangle,t\rangle} \lambda y_{\langle\langle e,t\rangle,t\rangle} \lambda w_{\langle e,t\rangle} \lambda u_e$ $[intersection\_between(x)(y)(u)]$	$\langle \langle\langle e,t\rangle , t\rangle , \langle \langle\langle e,t\rangle , t\rangle ,$ $\langle\langle e,t\rangle , \langle e,t\rangle\rangle \rangle\rangle$
TO_a	$\lambda x_{\langle\langle e,t\rangle,t\rangle} \lambda w_{\langle e,t\rangle} \lambda u_e[parallel\_to(x)(u)]$	$\langle\langle\langle e,t\rangle , t\rangle , \langle\langle e,t\rangle , \langle e,t\rangle\rangle\rangle$
TO_b	$\lambda x_{\langle\langle e,t\rangle,t\rangle} \lambda w_{\langle e,t\rangle} \lambda u_e$ $[perpendicular\_to(x)(u)]$	$\langle\langle\langle e,t\rangle , t\rangle , \langle\langle e,t\rangle , \langle e,t\rangle\rangle\rangle$
TO_c	$\lambda x_{\langle\langle e,t\rangle,t\rangle} \lambda w_{\langle e,t\rangle} \lambda u_e[r\_parallel\_to(x)(u)]$	$\langle\langle\langle e,t\rangle , t\rangle , \langle\langle e,t\rangle , \langle e,t\rangle\rangle\rangle$
TO_d	$\lambda x_{\langle\langle e,t\rangle,t\rangle} \lambda w_{\langle e,t\rangle} \lambda u_e$ $[r\_perpendicular\_to(x)(u)]$	$\langle\langle\langle e,t\rangle , t\rangle , \langle\langle e,t\rangle , \langle e,t\rangle\rangle\rangle$
WITH	$\lambda x_{\langle\langle e,t\rangle,t\rangle} \lambda w_{\langle e,t\rangle} \lambda u_e[painted\_with(x)(u)]$	$\langle\langle\langle e,t\rangle , t\rangle , \langle\langle e,t\rangle , \langle e,t\rangle\rangle\rangle$
WITH_a	$\lambda x_{\langle\langle e,t\rangle,t\rangle} \lambda w_{\langle e,t\rangle} \lambda u_e$ $[r\_painted\_with(x)(u)]$	$\langle\langle\langle e,t\rangle , t\rangle , \langle\langle e,t\rangle , \langle e,t\rangle\rangle\rangle$

Table 3.3: Abbreviations for some expressions of G

Note that the mnemonics BETWEEN, TO_a, TO_b, and WITH, abbreviate expressions whose argument denoted by w is not used in the body of the corresponding lambda expression. The predicates: *intersection*, *parallel*, *perpendicular*, and *painted*, that are destined to instantiate the variable w, in every case, are conveniently replaced by the predicates *intersection_between*, *parallel_to*, *perpendicular_to*, and *painted_with*.

The rules to define the expressions of G that can be translated into L are the following:

$S1_G$ Sentences

If $\alpha \in E_{\langle\langle e,t\rangle,t\rangle}$ and $\beta \in E_{\langle e,t\rangle}$ then $F_{G1}(\alpha, \beta) \in E_t$, where $F_{G1}(x, y) = x(y)$.

Examples:

- D_2(BE_a(A(*intersection*)))
- L_5(BE_b(WITH(AP)(*painted*)))
- SOME(*line*)(BE_b(*horizontal*))
- (THE(BETWEEN(L_2)(L_1)(*intersection*))(*outside*(EVERY(*circle*)))

$S2_G$ Transitive verb phrases

If $\alpha \in E_{\langle\langle\langle e,t\rangle,t\rangle,\langle e,t\rangle\rangle}$ and $\beta \in E_{\langle\langle e,t\rangle,t\rangle}$ then $F_{G1}(\alpha, \beta) \in E_{\langle e,t\rangle}$.

Examples:

- BE_a(A(*intersection*))
- *outside*(EVERY(*circle*))
- *inside*(C_2)

$S3_G$ Attributive verb phrases

If $\alpha \in E_{\langle\langle e,t\rangle,\langle e,t\rangle\rangle}$ and $\beta \in E_{\langle e,t\rangle}$ then $F_{G1}(\alpha, \beta) \in E_{\langle e,t\rangle}$.

Examples:

- BE_b(*horizontal*)
- BE_b(WITH(AP)(*painted*))
- BE_b(TO(THE(*line*))(*perpendicular*))

$S4_G$ Terms

If $\alpha \in E_{\langle\langle e,t\rangle,\langle\langle e,t\rangle,t\rangle\rangle}$ and $\beta \in E_{\langle e,t\rangle}$ then $F_{G1}(\alpha, \beta) \in E_{\langle\langle e,t\rangle,t\rangle}$.

Examples:

- A(*intersection*)
- SOME(*line*)
- THE(BETWEEN(L_2)(L_1)(*intersection*))
- EVERY(*circle*)

S5$_G$ Common nouns
If $\alpha \in E_{\langle\langle e,t\rangle,\langle e,t\rangle\rangle}$ and $\beta \in E_{\langle e,t\rangle}$ then $F_{G1}(\alpha, \beta) \in E_{\langle e,t\rangle}$.

Examples:
- BETWEEN(L_2)(L_1)(*intersection*)
- BETWEEN(A(*line*))(L_1)(*intersection*)

S6$_G$ Compound adjectives
If $\alpha \in E_{\langle\langle e,t\rangle,\langle e,t\rangle\rangle}$ and $\beta \in E_{\langle e,t\rangle}$ then $F_{G1}(\alpha, \beta) \in E_{\langle e,t\rangle}$.

Examples:
- WITH(AP)(*painted*)
- TO_b(THE(*line*))(*perpendicular*)

S7$_G$ *between* prepositional phrases
If $\alpha, \beta \in E_{\langle\langle e,t\rangle,t\rangle}$ and α, β are either L_i or $Q(line)$,[11] then $F_{G2}(\alpha, \beta) \in E_{\langle\langle e,t\rangle,\langle e,t\rangle\rangle}$, where $F_{G2}(x, y)$ =BETWEEN$(x)(y)$.

Examples:
- BETWEEN(L_2)(L_1)
- BETWEEN(A(*line*))(L_1)

S8$_G$ *with* prepositional phrases
If $\alpha \in E_{\langle\langle e,t\rangle,t\rangle}$ and α is either AP, LP or $Q(paint)$, then $F_{G3}(\alpha) \in E_{\langle\langle e,t\rangle,\langle e,t\rangle\rangle}$, where $F_{G3}(x)$ =WITH(x).

Examples:
- WITH(AP)
- WITH(LP)
- WITH(SOME(*paint*))

S9$_G$ *to* prepositional phrases
If $\alpha \in E_{\langle\langle e,t\rangle,t\rangle}$ and α is either L_i or $Q(line)$, then $F_{G4}(\alpha) \in E_{\langle\langle e,t\rangle,\langle e,t\rangle\rangle}$, and $F_{G5}(\alpha) \in E_{\langle\langle e,t\rangle,\langle e,t\rangle\rangle}$, where $F_{G4}(x) = \mathrm{TO}_a(x)$ and $F_{G5}(x) = \mathrm{TO}_b(x)$.

Examples:
- TO_a(THE(*line*))
- TO_b(L_5)

[11] Where Q stands for any of the quantifiers: A, SOME, THE, or EVERY, defined in Table 3.3, and similarly in rules S8$_G$ and S9$_G$.

3.2.2.3 Semantics of G

The semantics of a language establishes the interpretation of its well-formed expressions. This is done by providing a *model* for the language, i.e., stating how the expressions of the language are related to a *domain of interpretation*.

Interpretation domains of G

The language G will be interpreted according to two different, though related, domains of interpretation, namely, a geometrical domain, E, and W, which represents the world (see Figs. 3.4 and 3.5). The domain E is constituted by geometrical objects thought of as an abstraction from any kind of material realisation. From this standpoint, it is expected that any individual with a spatial extension of W has its corresponding geometrical object in E, and that this object is distinctively unique in E.

Formally, E can be thought of as a set of geometrical objects in the bidimensional space and the real numbers, i.e., $E \subseteq P(\Re^2) \cup \Re$, where $\Re$ is the set of real numbers, and $P(S)$ represents the power set of S. For the purposes of this thesis, E can contain dots, circles, line segments, and polylines, plus real numbers. A dot $\langle x, y\rangle$ is introduced into E as the set $\{\langle x, y\rangle\}$. A circle with center in $\langle h, k\rangle$ and radius r, is introduced as the set $\{\langle x, y\rangle \,|(x-h)^2 + (y-k)^2 \leq r^2\}$. A line segment starting in $\langle x_0, y_0\rangle$ and ending in $\langle x_1, y_1\rangle$, is introduced as the set $\{\langle x, y\rangle \,|\, \langle x, y\rangle = \langle x_0, y_0\rangle + t * (\langle x_1, y_1\rangle - \langle x_0, y_0\rangle), 0 \leq t \leq 1\}$. Finally, a polyline is introduced as the union of the sets corresponding to the line segments composing the polyline. As a result, for instance, a *circle* of E is a set of dots, but also a *dot* of E is a set of dots containing only one dot. Furthermore, not all dots, circles, line segments, and polylines belong to E, but only those introduced by the explicit declarations in a state of interpretation as it is described below.

The segment of W of interest for this work is that related to the design of electric transmission towers. This segment of the world can only contain metallic *bars*, which are the structural elements to build up a whole tower, *crossarms*, which are trapezoidal structural elements, composed of metallic bars, for the support of the cables, *dielectric areas*, which are protection areas around the conductive cables, *paints*, which are used to cover or protect the metallic elements of the tower, and *joints,* which are the ends of the bars of the tower or intersections between bars.

E and W are related through the representation function F_E, which assigns to each object of E the individual object of W that it represents. Table 3.4 shows the relevant correspondences established by function F_E. It should be noted here that the *geometry* of a metallic bar is abstracted as the projection of its axis to a bidimensional plane appropriate for the design task, and the *geometry* of a dielectric area, as the projection of the dielectric cylinder around the cable to the same plane, that happens to be orthogonal to the cable.

Object of E	Individual of W
A line	A metallic bar with the same *geometry* as the line
A circle	A dielectric area with the same *geometry* as the circle
A polyline	A crossarm with the same *geometry* as the polyline
A dot	A joint with the same *geometry* as the dot

Table 3.4: Correspondences between E and W, established by F_E

To interpret G, its symbols are partitioned into two subsets: those symbols that have a geometrical interpretation, H_0, and those that do not have it, H, as shown in Fig. 3.5. Thus, there will be two interpretation functions: F_0, for the geometrically interpretable symbols, and F, for the other symbols. To characterise these functions, the concept of the *possible denotation* for an expression of a type is introduced below.

The notation D_x stands for the *set of possible denotations for expressions of type x*. The sets of this nature are defined as follows:

1. For the basic types t and e:

1.1 D_t is $\{0, 1\}$.

1.2 When G is interpreted with respect to E, D_e is E.

1.3 When G is interpreted with respect to W, D_e is W.

2. For a composite type $\langle a, b\rangle$, $D_{\langle a,b\rangle} = D_b^{D_a}$, which is the set of all functions from D_a into D_b.

According to these definitions, F_0 is a function that assigns to each constant of G of type a a member of D_a, with D_e as specified in **1.2**, and, similarly, F is a function that assigns to each constant of G of type a a member of D_a, with D_e as specified in **1.3**.

Furthermore, for geometrically interpretable symbols, an interpretation with respect to W can be built up, i.e., the function F can be indirectly computed according to the structural relations between F and F_0 given below.

State of interpretation

A state of interpretation of G is a description of a relevant state-of-affairs of both the geometrical world E, and the world W, in a given time. This state is defined by means of a set of declarations specifying the objects in E and W and their properties. Such declarations are propositions of language G, typically, equalities whose left hand side (LHS) is a constant destined to *name* an object and whose right hand side (RHS) is a description denoting a property or a set of

Constant of G	Expression of G	Interpretation in E (under F_0)
Any numeral	e.g., 3.14	Conventional meaning, e.g., 3.14
the_dot	$the\_dot(x)(y)$	for any reals, x and y, is mapped to the set $\{\langle x, y\rangle\}$
the_line	$the\_line(the\_dot\ (x_0)(y_0))(the\_dot\ (x_1)(y_1))$	for any reals x_0, y_0, x_1, and y_1, is mapped to the set $\{\langle x, y\rangle \mid \langle x, y\rangle = \langle x_0, y_0\rangle + t * (\langle x_1, y_1\rangle - \langle x_0, y_0\rangle), 0 \leq t \leq 1\}$
the_circle	$the\_circle(\ the\_dot\ (x_0)(y_0))(r)$	for any reals x_0, y_0, and r, is mapped to the set $\{\langle x, y\rangle \mid (x - x_0)^2 + (y - y_0)^2 \leq r^2\}$
path	$path(s_1)(s_2)$	for the two sets S_1 and S_2, corresponding to any two *contiguous* polylines denoted by s_1 and s_2 in G, is mapped to the set $S_1 \cup S_2$

Table 3.5: Interpretation of some constants of G under F_0

properties that pick the intended object (the bearer of the name) in the interpretation state. The constants *the_dot*, *the_line*, *the_circle*, and *path*, are used to introduce these definitions[12] as shown in Table 3.5.
Example.-

The following is a simple knowledge base representing a possible state of interpretation, whose graphical depiction is shown in Fig. 3.7.

$$
\begin{array}{l}
\underline{\text{A possible state of interpretation}} \\
d_4 = the\_dot(0.0)(2.0) \\
d_5 = the\_dot(4.0)(4.0) \\
d_7 = the\_dot(7.0)(4.0) \\
d_8 = the\_dot(11.0)(2.0) \\
l_4 = the\_line(d_4)(d_5) \\
l_5 = the\_line(d_5)(d_7) \\
l_6 = the\_line(d_7)(d_8) \\
l_7 = the\_line(d_4)(d_8) \\
c_0 = the\_circle(the\_dot(0.0)(5.0))(2.0) \\
p_2 = path(l_4)(path(l_5)(path(l_6)(l_7)))
\end{array}
\tag{3.1}
$$

In the description of this state, for instance, the symbol l_4 denotes the line segment that corresponds to the set $\{\langle x, y\rangle \mid \langle x, y\rangle = \langle 0.0, 2.0\rangle + t * (\langle 4.0, 4.0\rangle - \langle 0.0, 2.0\rangle), 0 \leq t \leq 1\}$.

[12]Other constants, such as *intersection_between*$_*$, can be used to introduce *emergent* objects as it is explained below in Section 3.4 (p. 104).

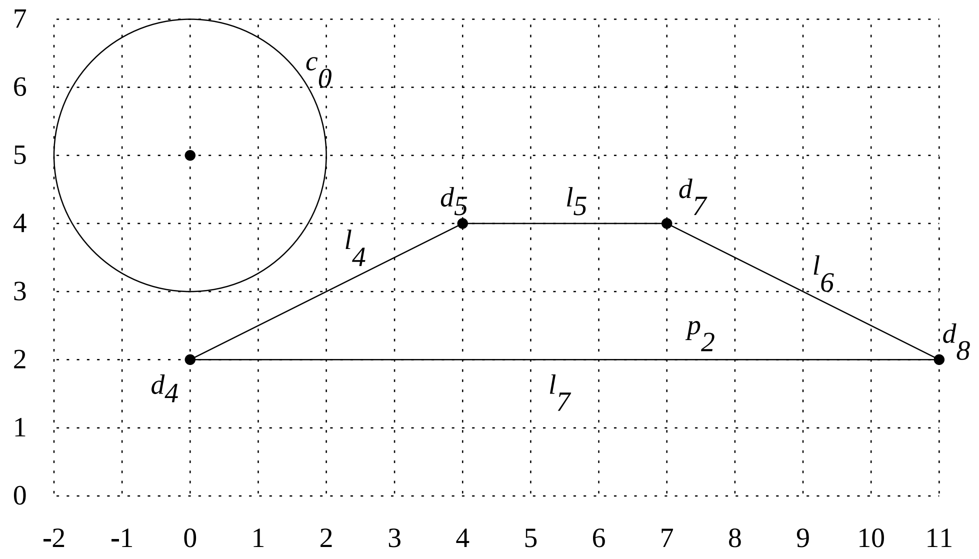


Figure 3.7: The graphical depiction of a possible state of interpretation

Due to the representation function F_E from E into W, the introduction of geometrical objects into E implies the introduction of the corresponding individuals into W (see Table 3.4). Thus, the previous example declarations, given in expression (3.1), have implicitly specified the introduction of four joints, four metallic bars, one dielectric area, and one crossarm into W.

Other non-geometrical properties that hold in this particular state of interpretation can be specified by means of other propositions of G involving symbols that are not geometrically interpretable. For instance, to state that two bars are painted with some kind of paint, the following propositions have to be added to the description of the state of interpretation:

<u>A possible state of interpretation (cont.)</u> (3.2)
$painted\_with_*(anticorrosive\_paint)(l_4)$
$painted\_with_*(anticorrosive\_paint)(l_5)$

A set of propositions such as the given in expressions (3.1) and (3.2), can constitute the initial database of a design problem, and they correspond to the so called *list of graphical segments* or *central structure storage* in traditional computer graphics systems [Foley et al. 1994, pp. 12,247], though here this specification is made in the context of a logical language (G).

Structural relations between F and F_0:

By virtue of the relationship between W and E, the following structural relations between the interpretation functions F and F_0 hold:

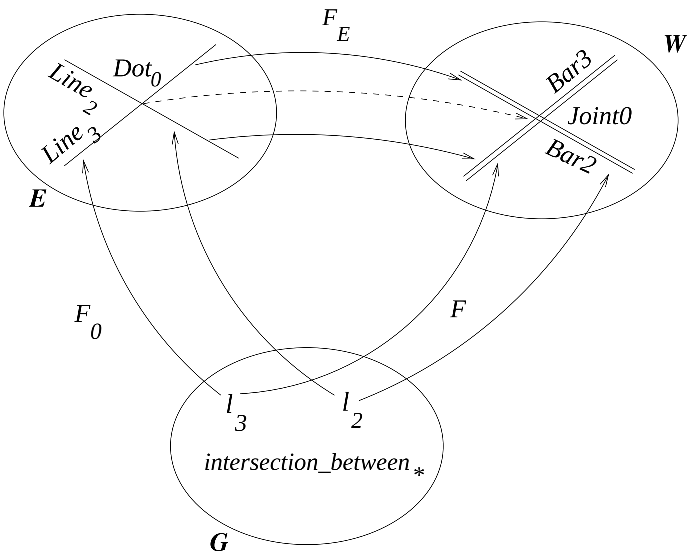


Figure 3.8: Structural relations between F and F_0

1. If k is a constant of G of a simple type, s, then

   1.1 if the type s is t, k denotes a truth value, and $F(k) = F_0(k)$,

   1.2 if the type s is e, k denotes a geometrical object when interpreted with respect to E, and a world individual when interpreted with respect to W, and $F(k) = F_E(F_0(k))$,

2. If k is a constant of G of a composite type $\langle a_1, \langle a_2, \ldots \langle a_n, b\rangle \ldots\rangle\rangle$, where $a_1, a_2, \ldots, a_n$ are either simple or composite types, and b is a simple type, either t or e, and $k_1, k_2, \ldots, k_n$ are arbitrary constants of types $a_1, a_2, \ldots, a_n$ respectively, then $F_0(k)$ and $F(k)$ are functions such that

   2.1 if the type b is t, then
   $F(k)(F(k_1))(F(k_2))\ldots(F(k_n)) = F_0(k)(F_0(k_1))(F_0(k_2))\ldots(F_0(k_n))$,

   2.2 if the type b is e, then
   $F(k)(F(k_1))(F(k_2))\ldots(F(k_n)) = F_E(F_0(k)(F_0(k_1))(F_0(k_2))\ldots(F_0(k_n)))$.

Example.-
Suppose fragments of G, E, and W as shown in Fig. 3.8. Here, $F_0(l_3)$ is the line labelled with $Line_3$ in the figure, $F_E(Line_3)$ is the bar (an object of the world) labelled with *Bar3* in the figure, etc., and similarly for the other objects. To simplify the explanation, it will be said that $F_0(l_3) = Line_3$, and then, similarly, $F_E(Line_3)$ =*Bar3*, $F_0(l_2) = Line_2$, and $F_E(Line_2)$ =*Bar2*. By the structural relation **1.2**, it can be concluded that $F(l_3)$ =*Bar3*, and $F(l_2)$ =*Bar2*.

In order to review what happens with the constant $intersection\_between_*$ of G, it should be noted that this constant, when interpreted with respect to E, maps to a geometrical function that takes a line as argument, say *Line-A*, and gives as a result another function that taking as argument another line, *Line-B*, gives as a result the dot that is the intersection between *Line-A* and *Line-B*.

According to Table 3.2 (p. 77), the constant $intersection\_between_*$ is of a type $\langle e, \langle e, e\rangle\rangle$, then to apply the structural relations given above, let us take $k = intersection\_between_*$, $\langle a_1, \langle a_2, b\rangle\rangle = \langle e, \langle e, e\rangle\rangle$, and $k_1 = l_3, k_2 = l_2$, then, since b is e, by structural relation **2.2** it can be concluded that

$$F(intersection\_between_*)(F(l_3))(F(l_2)) =$$
$$F_E(F_0(intersection\_between_*)(F_0(l_3))(F_0(l_2))),$$
$$F(intersection\_between_*)(Bar_3)(Bar_2) =$$
$$F_E(F_0(intersection\_between_*)(Line_3)(Line_2)) = F_E(Dot_0) = Joint0.$$

Semantic rules of G

Having two possible interpretation domains, E and W, two models are introduced for language G, namely: $M_0 = \langle E, I, F_0\rangle$ and $M_G = \langle W, I, F\rangle$, where I is a linearly ordered set of states of interpretation: $i_1, i_2, \ldots, i_n$, and the functions F_0 and F are as defined above.

For the interpretation of the variables, a *value assignment function* g is defined. This function assigns to each variable of type a a member of D_a.

In order to provide the semantics of the language G, the following notational conventions are introduced.

Notational convention 1 *For any expression α, $[\![\alpha]\!]^{M,i,g}$ is used to denote the semantic value of α with respect to a model M and a value assignment g in a state $i \in I$.*

Notational convention 2 *The notation $g^{u/d}$ is used to indicate the value assignment exactly like g except that it assigns the individual d to the variable u.*

Due to the structural relations between F and F_0, given in Section 3.2.2.3 (p. 84), it suffices to consider model M_0 in most of the cases, and make a reference to model M_G only in those cases where it is not possible to build up a geometrical interpretation of an expression of G. Thus, when an expression of G is purely geometric, it will be interpreted according to model M_0, and when an expression of G is purely *linguistic*[13] or contains a mixture of geometrically interpretable sub-expressions with non geometrical sub-expressions, it will be interpreted according to model M_G.

[13] By *purely linguistic expression* here we mean an expression of language G that is built up only with elements of the subset H_1 shown in Fig. 3.5 (p. 68).

Then, considering the model M_0, the semantic values for language G are defined as follows:

1. If $\alpha \in C_s$, then $[\![\alpha]\!]^{M_0,i,g} = [F_0(\alpha)](i)$ (that is, the value of $F_0(\alpha)$ in state i).
2. If $\mu \in V_s$, then $[\![\mu]\!]^{M_0,i,g} = [g(\mu)](i)$.
3. If $\tau \in E_a$, and $\phi \in E_{\langle a,b \rangle}$, then $[\![\phi(\tau)]\!]^{M_0,i,g} = [\![\phi]\!]^{M_0,i,g}([\![\tau]\!]^{M_0,i,g})$.
4. If $\mu \in V_b$, and $\alpha \in E_a$, then $[\![\lambda\mu[\alpha]]\!]^{M_0,i,g}$ is that function $h : D_b \to D_a$ such that for all individuals $k \in D_b$, $h(k)$ is equal to $[\![\alpha]\!]^{M_0,i,g^{\mu/k}}$.
5. If $\mu \in V_s$, and $\beta \in E_t$ then $[\![\exists\mu(\beta)]\!]^{M_0,i,g} = 1$ iff for some $d \in D_s$, $[\![\beta]\!]^{M_0,i,g^{\mu/d}} = 1$.
6. If $\mu \in V_s$, and $\beta \in E_t$ then $[\![\forall\mu(\beta)]\!]^{M_0,i,g} = 1$ iff for every $d \in D_s$, $[\![\beta]\!]^{M_0,i,g^{\mu/d}} = 1$.

For those cases where there is not geometrical interpretation, the definition of the semantic values of G according to model M_G is as follows:

1. If $\alpha \in C_s$, then $[\![\alpha]\!]^{M_G,i,g} = [F(\alpha)](i)$ (that is, the value of $F(\alpha)$ in state i).
2. If $\mu \in V_s$, then $[\![\mu]\!]^{M_G,i,g} = [g(\mu)](i)$.
3. If $\tau \in E_a$, and $\phi \in E_{\langle a,b \rangle}$, then $[\![\phi(\tau)]\!]^{M_G,i,g} = [\![\phi]\!]^{M_G,i,g}([\![\tau]\!]^{M_G,i,g})$.
4. If $\mu \in V_b$, and $\alpha \in E_a$, then $[\![\lambda\mu[\alpha]]\!]^{M_G,i,g}$ is that function $h : D_b \to D_a$ such that for all individuals $k \in D_b$, $h(k)$ is equal to $[\![\alpha]\!]^{M_G,i,g^{\mu/k}}$.
5. If $\mu \in V_s$, and $\beta \in E_t$ then $[\![\exists\mu(\beta)]\!]^{M_G,i,g} = 1$ iff for some $d \in D_s$, $[\![\beta]\!]^{M_G,i,g^{\mu/d}} = 1$.
6. If $\mu \in V_s$, and $\beta \in E_t$ then $[\![\forall\mu(\beta)]\!]^{M_G,i,g} = 1$ iff for every $d \in D_s$, $[\![\beta]\!]^{M_G,i,g^{\mu/d}} = 1$.

As mentioned above, the constants for the predicates: *inside, outside, perpendicular_to, parallel_to, painted_with*, and *intersection_between*, admit type-raised arguments that are used to preserve quantification properties during the translations between G and L. However, these predicates cannot be directly interpreted. Instead, they have to be converted to their equivalent non-type-raised version. For instance, the expression $painted\_with(\lambda P[P(anticorrosive\_paint)])$ (l_3), would be true if the world individual corresponding to l_3, a specific bar, is painted with anticorrosive paint. However, the first argument of this predicate refers not to the actual anticorrosive paint, but to the set of properties that this particular paint has. Then, to have a proper translation, this expression is to be translated to $painted\_with_*(anticorrosive\_paint)(l_3)$. To formalise this sort of syntactic transformations, the equivalences established by the following meaning postulates are introduced:

Meaning postulates.-

MP1. $\forall x \forall P[\delta(P)(x) \leftrightarrow P(\lambda y[\delta_*(y)(x)])]$, where $\delta \in \{inside, outside, perpendicular\_to, parallel\_to, painted\_with\}$.

MP2. $\forall x \forall P_1 \forall P_2[\delta(P_1)(P_2)(x) \leftrightarrow P_2(\lambda u[P_1(\lambda v[\delta_*(v)(u) = x])])]$, where $\delta = intersection\_between$.

MP3. $\forall x \forall P[\delta(P)(x) \leftrightarrow P(\lambda y[\delta_*(x)(y)])]$, where $\delta \in \{r\_inside, r\_outside, r\_perpendicular\_to, r\_parallel\_to, r\_painted\_with\}$.

Here, P, P_1, and P_2 range over type raised arguments (of type $\langle\langle e, t\rangle, t\rangle$), and x, y, u, and v, over individuals.

Examples to illustrate the semantic evaluation defined here are given below in Section 3.3.1 (p. 100).

3.2.3 Translations between L and G

In this section, the translation functions ρ, from L into G, and ρ^{-1}, from G into L, are defined in such a way that a correspondence between syntactic categories of L and types of the logical language G, is preserved. Following Montague [Montague 1974*d*, p. 260], this correspondence is defined in terms of function f, as follows:

1. $f(\mathsf{t}) = t$
2. $f(\mathsf{IV}) = f(\mathsf{ADJ}) = f(\mathsf{CN}) = \langle e, t\rangle$
3. For any categories A and B, $f(\mathrm{A/B}) = \langle f(\mathrm{B}), f(\mathrm{A})\rangle$.

The translation rules presented below will preserve this correspondence.

3.2.3.1 Translation from L into G

There will be a translation rule corresponding to each phrase structure rule in the syntax of L. A phrase structure tree (ps-tree) embodies the structure imposed to a particular phrase by the syntactic rules, i.e., each syntactic rule introduces a ps-subtree. Therefore, each semantic rule corresponding to a syntactic rule, will be defined having as reference the subtree that the syntactic rule has introduced.

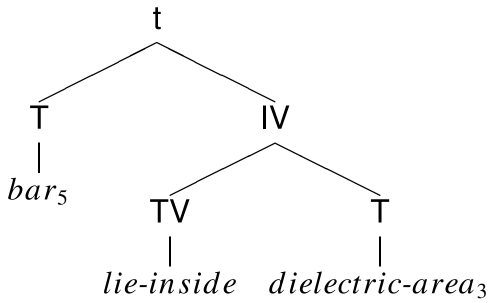
The following convention will be adopted to denote phrase structure trees in the translation rules.

Notational convention 3

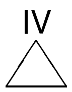
1. *The empty tree is represented by* $\lceil\rceil$.
2. *A tree with root* α*, and branches* $\beta_1, \beta_2, \ldots, \beta_n$ *is represented by* $\alpha\lceil\beta_1, \beta_2, \ldots, \beta_n\rceil$.
3. *A tree with root* α*, and whose branches are not necessary to be mentioned, is represented by* $\alpha\lceil\ldots\rceil$.
4. *The linearisation of a tree,* T*, is denoted by* $\ell(T)$*, and consists of the concatenation of all the leaves found during a depth-first order traversal of the tree, with the morphological transformations required by the applied syntactic rules.*
5. *The phrase structure tree given by the syntactic analysis of a phrase* p*, is denoted by* $\Lambda(p)$.

Examples.-

- The tree

  $\Lambda(bar_5\ \textit{lies-inside dielectric-area}_3) =$ 

  is represented by $\mathsf{t}\lceil\mathsf{T}\lceil bar_5\rceil, \mathsf{IV}\lceil\mathsf{TV}\lceil \textit{lie-inside}\rceil, \mathsf{T}\lceil \textit{dielectric-area}_3\rceil\rceil\rceil$, and its linearisation is given by $\ell(\mathsf{t}\lceil\mathsf{T}\lceil bar_5\rceil, \mathsf{IV}\lceil\mathsf{TV}\lceil \textit{lie-inside}\rceil, \mathsf{T}\lceil \textit{dielectric-area}_3\rceil\rceil\rceil) = bar_5\ \textit{lies-inside dielectric-area}_3$.

- The tree 

  , whose branches are not shown, is represented by $\mathsf{IV}\lceil\ldots\rceil$, in this case, conventionally, $\ell(\mathsf{IV}\lceil\ldots\rceil) = \ldots$.

The definition of the translation function ρ, from L into G, is given in two parts, namely, the translation of the lexical elements of L, that is shown in Table 3.6, and the following set of rules.

Translation rules.-

Constant of L (α)	Category definition	Translation into G $\rho(\alpha)$	Type in G
$joint_i$ bar_i $crossarm_i$ $dielectric\text{-}area_i$ $luminous\text{-}paint$ $anticorrosive\text{-}paint$	t/IV	$\lambda P[P(d_i)]$ $\lambda P[P(l_i)]$ $\lambda P[P(p_i)]$ $\lambda P[P(c_i)]$ $\lambda P[P(luminous\_paint)]$ $\lambda P[P(anticorrosive\_paint)]$	$\langle\langle e,t\rangle, t\rangle$
dot $intersection$ bar $crossarm$ $dielectric\text{-}area$ $paint$	CN	$gdot$ $gintersection$ $line$ $polygon$ $circle$ $gpaint$	$\langle e,t\rangle$
$horizontal$ $vertical$ $parallel$ $perpendicular$ $painted$	ADJ	$ghorizontal$ $gvertical$ $gparallel$ $gperpendicular$ $gpainted$	$\langle e,t\rangle$
$lie\text{-}inside$ $lie\text{-}outside$ be	IV/(t/IV)	$inside$ $outside$ $\lambda P\lambda x[P(\lambda y[x=y])]$	$\langle\langle\langle e,t\rangle, t\rangle, \langle e,t\rangle\rangle$
be	IV/ADJ	$\lambda P\lambda x[P(x)]$	$\langle\langle e,t\rangle, \langle e,t\rangle\rangle$
a $some$ the $every$	(t/IV)/CN	$\lambda P\lambda Q\exists x[P(x)\wedge Q(x)]$ $\lambda P\lambda Q\exists x[P(x)\wedge Q(x)]$ $\lambda P\lambda Q\exists y[\forall x[P(x)\leftrightarrow x=y]\wedge Q(y)]$ $\lambda P\lambda Q\forall x[P(x)\rightarrow Q(x)]$	$\langle\langle e,t\rangle, \langle\langle e,t\rangle, t\rangle\rangle$

Table 3.6: Translations of constants of L to G

T0$_{L\text{-}G}$ Lexical elements

If a subtree $\alpha = \gamma\lceil\beta\rceil$ is introduced by a lexical category γ, and β is a lexical item (a constant of L of category γ), the translation of α is given by $\rho(\alpha) = \rho(\beta)$. The translation of the lexical items is given in Table 3.6.

Examples:

- The analysis of the lexical item $joint_2$, gives rise to the tree $\tau = \mathsf{T}\lceil joint_2\rceil$, whose translation is $\rho(\tau) = \lambda P[P(d_2)]$ or, equivalently, $\rho(\tau) = \mathrm{D}_2$, using the abbreviations of Table 3.3 (p. 78). With the notation introduced above (p. 89), this can be written as $\rho(\Lambda(joint_2)) = \rho(\mathsf{T}\lceil joint_2\rceil) = \lambda P[P(d_2)] = \mathrm{D}_2$.

T1$_{L\text{-}G}$ Sentences

The syntactic rule $\mathsf{t} \rightarrow \mathsf{T}\ \mathsf{IV}$, introduces the subtree $\alpha = \mathsf{t}\lceil\beta,\gamma\rceil$, where $\beta = \mathsf{T}\lceil\ldots\rceil$, and $\gamma = \mathsf{IV}\lceil\ldots\rceil$, the translation of α is given by $\rho(\alpha) = \rho(\beta)(\rho(\gamma))$.

Examples:

- The analysis of sentence *joint*$_2$ *is an intersection*, gives rise to a tree $\tau = \mathsf{t}\lceil\mathsf{T}\lceil joint_2\rceil, \mathsf{IV}\lceil\mathsf{TV}\lceil be\rceil, \mathsf{T}\lceil\mathsf{T/CN}\lceil a\rceil, \mathsf{CN}\lceil intersection\rceil\rceil\rceil\rceil$, whose translation is $\rho(\tau) = \mathrm{D}_2(\mathrm{BE}_a(\mathrm{A}(intersection)))$, that is, $\rho(\Lambda(joint_2\ is\ an\ intersection)) = \rho(\mathsf{t}\lceil\mathsf{T}\lceil joint_2\rceil, \mathsf{IV}\lceil\mathsf{TV}\lceil be\rceil, \mathsf{T}\lceil\mathsf{T/CN}\lceil a\rceil, \mathsf{CN}\lceil intersection\rceil\rceil\rceil\rceil) = \mathrm{D}_2(\mathrm{BE}_a(\mathrm{A}(gintersection)))$.
- $\rho(\Lambda(bar_5\ is\ painted\ with\ anticorrosive\text{-}paint)) = \rho(\mathsf{t}\lceil\mathsf{T}\lceil bar_5\rceil, \mathsf{IV}\lceil\mathsf{IV/ADJ}\lceil be\rceil, \mathsf{ADJ}\lceil\mathsf{ADJ}\lceil painted\rceil, \mathsf{PP'}\lceil\mathsf{with}, \mathsf{T}\lceil anticorrosive\text{-}paint\rceil\rceil\rceil\rceil\rceil) = \mathrm{L}_5(\mathrm{BE}_b\ (\mathrm{WITH}(\mathrm{AP})(gpainted)))$.

T2$_{L\text{-}G}$ Transitive verb phrases

The syntactic rule $\mathsf{IV} \rightarrow \mathsf{TV}\ \mathsf{T}$, introduces the subtree $\alpha = \mathsf{IV}\lceil\beta,\gamma\rceil$, where $\beta = \mathsf{TV}\lceil\ldots\rceil$, and $\gamma = \mathsf{T}\lceil\ldots\rceil$, the translation of α is given by $\rho(\alpha) = \rho(\beta)(\rho(\gamma))$.

Examples:

- $\rho(\Lambda(\ is\ an\ intersection)) = \rho(\mathsf{IV}\lceil\mathsf{TV}\lceil be\rceil, \mathsf{T}\lceil\mathsf{T/CN}\lceil a\rceil, \mathsf{CN}\lceil intersection\rceil\rceil\rceil) = \mathrm{BE}_a(\mathrm{A}(intersection))$
- $\rho(\Lambda(lie\text{-}outside\ every\ dielectric\text{-}area)) = \rho(\mathsf{IV}\lceil\mathsf{TV}\lceil lie\text{-}outside\rceil, \mathsf{T}\lceil\mathsf{T/CN}\lceil every\rceil, \mathsf{CN}\lceil dielectric\text{-}area\rceil\rceil\rceil) = outside(\mathrm{EVERY}(circle))$

T3$_{L\text{-}G}$ Attributive verb phrases

The syntactic rule $\mathsf{IV} \rightarrow \mathsf{IV/ADJ}\ \mathsf{ADJ}$, introduces the subtree $\alpha = \mathsf{IV}\lceil\beta,\gamma\rceil$, where $\beta = \mathsf{IV/ADJ}\lceil\ldots\rceil$, and $\gamma = \mathsf{ADJ}\lceil\ldots\rceil$, the translation of α is given by $\rho(\alpha) = \rho(\beta)(\rho(\gamma))$.

Examples:

- $\rho(\Lambda(are\ horizontal)) = \rho(\mathsf{IV}\lceil\mathsf{IV/ADJ}\lceil be\rceil, \mathsf{ADJ}\lceil horizontal\rceil\rceil) = \mathrm{BE}_b($

ghorizontal).
- $\rho(\Lambda(\textit{is painted with anticorrosive-paint}))$ = $\rho($ IV$\lceil$ IV/ADJ$\lceil$ *be*$\rceil$, ADJ$\lceil$ ADJ$\lceil$*painted*$\rceil$, PP'$\lceil$with, T$\lceil$*anticorrosive-paint* $\rceil\rceil\rceil\rceil)$ = BE$_b$(WITH(AP) (*gpainted*)).

T4$_{L\text{-}G}$ Terms

The syntactic rule T $\rightarrow$ T/CN CN, introduces the subtree α = T$\lceil\beta,\gamma\rceil$, where β =T/CN$\lceil\ldots\rceil$, and γ =CN$\lceil\ldots\rceil$, the translation of α is given by $\rho(\alpha)$ = $\rho(\beta)(\rho(\gamma))$.

Examples:
- $\rho(\Lambda(\textit{an intersection}))$ = $\rho($ T$\lceil$ T/CN$\lceil$ *a*$\rceil$, CN$\lceil$ *intersection* $\rceil\rceil)$ = A(*gintersection*)
- $\rho(\Lambda(\textit{some bars}))$ = $\rho($ T$\lceil$ T/CN$\lceil$ *some*$\rceil$, CN$\lceil$ *bar*$\rceil\rceil)$ = SOME(*line*)
- $\rho(\Lambda(\textit{the intersection between } bar_2 \textit{ and } bar_1))$ = $\rho($T$\lceil$T/CN$\lceil$*the*$\rceil$, CN$\lceil$CN$\lceil$ *intersection*$\rceil$, PP$\lceil$between, T$\lceil bar_2\rceil$, T$\lceil bar_1$ $\rceil\rceil\rceil\rceil)$ =THE(BETWEEN (L$_2$)(L$_1$)(*gintersection*)).

T5$_{L\text{-}G}$ Common nouns

The syntactic rule CN $\rightarrow$ CN PP, introduces the subtree α = CN$\lceil\beta,\gamma\rceil$, where β =CN$\lceil\ldots\rceil$, and γ =PP$\lceil\ldots\rceil$, the translation of α is given by $\rho(\alpha)$ = $\rho(\gamma)(\rho(\beta))$.

Examples:
- $\rho(\Lambda(\textit{intersection between } bar_2 \textit{ and } bar_1))$ = $\rho($CN$\lceil$ CN$\lceil$*intersection*$\rceil$, PP $\lceil$between, T$\lceil bar_2\rceil$, T$\lceil bar_1$ $\rceil\rceil\rceil)$ =BETWEEN(L$_2$)(L$_1$)(*gintersection*).

T6$_{L\text{-}G}$ Compound adjectives

The syntactic rule ADJ $\rightarrow$ ADJ PP', introduces the subtree α = ADJ$\lceil\beta,\gamma\rceil$, where β =ADJ$\lceil\ldots\rceil$, and γ =PP'$\lceil\ldots\rceil$, the translation of α is given by $\rho(\alpha)$ = $\rho(\gamma)(\rho(\beta))$.

Examples:
- $\rho(\Lambda(\textit{painted with anticorrosive-paint}))$ = $\rho($ ADJ$\lceil$ ADJ$\lceil$*painted*$\rceil$, PP'$\lceil$ with, T$\lceil$*anticorrosive-paint* $\rceil\rceil\rceil)$ = WITH(AP)(*gpainted*).

T7$_{L\text{-}G}$ *between* prepositional phrases

The syntactic rule PP $\rightarrow$ between T and T, introduces the subtree α = PP$\lceil$between, β,and,$\gamma\rceil$, where β =T$\lceil\ldots\rceil$, and γ =T$\lceil\ldots\rceil$, the translation of α is given by $\rho(\alpha)$ = BETWEEN$(\rho(\gamma))(\rho(\beta))$.

Examples:
- $\rho(\Lambda(\textit{between } bar_2 \textit{ and } bar_1))$ = $\rho($PP$\lceil$between, T$\lceil bar_2\rceil$, T$\lceil bar_1\rceil\rceil)$ = BETWEEN(L$_2$)(L$_1$).

T8$_{L\text{-}G}$ *with* prepositional phrases

The syntactic rule PP' $\rightarrow$ with T, introduces the subtree α = PP'$\lceil$with, $\beta\rceil$, where β =T$\lceil\ldots\rceil$, the translation of α is given by $\rho(\alpha)$ =WITH$(\rho(\beta))$.

Examples:

- $\rho(\Lambda(\textit{with anticorrosive-paint}))$ = ρ(PP'$\lceil$ with, T$\lceil$*anticorrosive-paint*$\rceil\rceil$) = WITH(AP).

T9$_{L\text{-}G}$ *to* prepositional phrases

The syntactic rule PP' $\rightarrow$ to T, introduces the subtree α = PP'$\lceil$to, $\beta\rceil$, where β =T$\lceil\ldots\rceil$, the translation of α is given either by $\rho(\alpha)$ =TO$_a(\rho(\beta))$ or by $\rho(\alpha)$ =TO$_b(\rho(\beta))$.

Examples:

- $\rho(\Lambda(\textit{to the bar}))$ = ρ(PP'$\lceil$to, T$\lceil$T/CN$\lceil$*the*$\rceil$, CN$\lceil$*bar*$\rceil\rceil\rceil$) = TO$_a$(THE(*line*))
- $\rho(\Lambda(\textit{to } bar_5))$ = ρ(PP'$\lceil$ to, T$\lceil bar_5\rceil\rceil$) = TO$_b$(L$_5$)

3.2.3.2 Translation from *G* into *L*

In this section, the translation function ρ^{-1}, from G into L is presented. This function takes a valid expression of G and gives a phrase-structure tree as result. To obtain the natural language phrase corresponding to the given expression of G, this phrase-structure tree is linearised.

The basic translations from G into L are shown in Tables 3.7 and 3.8. Note that some composite expressions of G translate into constants of L, as shown in Table 3.8.

The rules for the translation of expressions of G into L are the following:

T1$_{G\text{-}L}$ Sentences

If $\alpha \in E_{\langle\langle e,t\rangle,t\rangle}$ and $\beta \in E_{\langle e,t\rangle}$, then the translation of $F_{G1}(\alpha,\beta) = \alpha(\beta)$ is $\rho^{-1}(F_{G1}(\alpha,\beta))$ =t$\lceil$T$\left\lceil\rho^{-1}(\alpha)\right\rceil$, IV$\left\lceil\rho^{-1}(\beta)\right\rceil\rceil$.

Examples:

- $\ell(\rho^{-1}$(D$_2$(BE$_a$(A(*gintersection*))))) = *joint*$_2$ *is an intersection*
- $\ell(\rho^{-1}$(L$_5$(BE$_b$(WITH(AP)(*gpainted*))))) = *bar*$_5$ *is painted with anticorrosive paint*
- $\ell(\rho^{-1}$(SOME(*line*)(BE$_b$(*ghorizontal*)))) = *some bars are horizontal*
- $\ell(\rho^{-1}$((THE(BETWEEN(L$_2$)(L$_1$)(*gintersection*))(*outside*(EVERY(*circle*)))) = *the intersection between bar*$_2$ *and bar*$_1$ *lies-outside every dielectric-area*

T2$_{G\text{-}L}$ Transitive verb phrases

If $\alpha \in E_{\langle\langle\langle e,t\rangle,t\rangle,\langle e,t\rangle\rangle}$ and $\beta \in E_{\langle\langle e,t\rangle,t\rangle}$, then the translation of $F_{G1}(\alpha,\beta) = \alpha(\beta)$ is $\rho^{-1}(F_{G1}(\alpha,\beta))$ =IV$\lceil$TV$\left\lceil\rho^{-1}(\alpha)\right\rceil$, T$\left\lceil\rho^{-1}(\beta)\right\rceil\rceil$.

Constant of G (α)	Translation into L $\rho^{-1}(\alpha)$
$gdot$	dot
$gintersection$	$intersection$
$line$	bar
$polygon$	$crossarm$
$circle$	$dielectric\text{-}area$
$gpaint$	$paint$
$ghorizontal$	$horizontal$
$gvertical$	$vertical$
$gparallel$	$parallel$
$gperpendicular$	$perpendicular$
$gpainted$	$painted$
$inside$	$lie\text{-}inside$
$outside$	$lie\text{-}outside$

Table 3.7: Translations of constants of G to L

Expression of G (α)	Translation into L $\rho^{-1}(\alpha)$
$\lambda P[P(d_i)]$	$joint_i$
$\lambda P[P(l_i)]$	bar_i
$\lambda P[P(p_i)]$	$crossarm_i$
$\lambda P[P(c_i)]$	$dielectric\text{-}area_i$
$\lambda P[P(luminous\_paint)]$	$luminous\text{-}paint$
$\lambda P[P(anticorrosive\_paint)]$	$anticorrosive\text{-}paint$
$\lambda P \lambda x[P(\lambda y[x = y])]$	be
$\lambda P \lambda x[P(x)]$	be
$\lambda P \lambda Q \exists x[P(x) \wedge Q(x)]$	a
$\lambda P \lambda Q \exists x[P(x) \wedge Q(x)]$	$some$
$\lambda P \lambda Q \exists y[\forall x[P(x) \leftrightarrow x = y] \wedge Q(y)]$	the
$\lambda P \lambda Q \forall x[P(x) \rightarrow Q(x)]$	$every$

Table 3.8: Translations of some expressions of G to constants of L

Examples:

- $\ell(\rho^{-1}(\mathrm{BE}_a(\mathrm{A}(gintersection))))$ = *be an intersection*
- $\ell(\rho^{-1}(outside(\mathrm{EVERY}(circle))))$ = *lie-outside every dielectric-area*
- $\ell(\rho^{-1}(inside(\mathrm{C}_2)))$ = *lie-inside dielectric-area*$_2$

T3$_{G\text{-}L}$ Attributive verb phrases

If $\alpha \in E_{\langle\langle e,t\rangle,\langle e,t\rangle\rangle}$ and $\beta \in E_{\langle e,t\rangle}$, then the translation of $F_{G1}(\alpha,\beta) = \alpha(\beta)$ is $\rho^{-1}(F_{G1}(\alpha,\beta))$ =IV⌈IV/ADJ$\left\lceil\rho^{-1}(\alpha)\right\rceil$, ADJ$\left\lceil\rho^{-1}(\beta)\right\rceil$⌉.

Examples:

- $\ell(\rho^{-1}(\mathrm{BE}_b(ghorizontal)))$ = *be horizontal*
- $\ell(\rho^{-1}(\mathrm{BE}_b(\mathrm{WITH}(\mathrm{AP})(gpainted))))$ = *be painted with anticorrosive-paint*
- $\ell(\rho^{-1}(\mathrm{BE}_b(\mathrm{TO}_b(\mathrm{THE}(line))(gperpendicular))))$ = *be perpendicular to the line*

T4$_{G\text{-}L}$ Terms

If $\alpha \in E_{\langle\langle e,t\rangle,\langle\langle e,t\rangle,t\rangle\rangle}$ and $\beta \in E_{\langle e,t\rangle}$, then the translation of $F_{G1}(\alpha,\beta) = \alpha(\beta)$ is $\rho^{-1}(F_{G1}(\alpha,\beta))$ =T⌈T/CN$\left\lceil\rho^{-1}(\alpha)\right\rceil$, CN$\left\lceil\rho^{-1}(\beta)\right\rceil$⌉.

Examples:

- $\ell(\rho^{-1}(\mathrm{A}(gintersection)))$ = *an intersection*
- $\ell(\rho^{-1}(\mathrm{SOME}(line)))$ = *some bars*
- $\ell(\rho^{-1}(\mathrm{THE}(\mathrm{BETWEEN}(\mathrm{L}_2)(\mathrm{L}_1)(gintersection))))$ = *the intersection between bar*$_2$ *and bar*$_1$
- $\ell(\rho^{-1}(\mathrm{EVERY}(circle)))$ = *every dielectric-area*

T5$_{G\text{-}L}$ Common nouns

If $\alpha \in E_{\langle\langle e,t\rangle,\langle e,t\rangle\rangle}$ and $\beta \in E_{\langle e,t\rangle}$, then the translation of $F_{G1}(\alpha,\beta) = \alpha(\beta)$ is $\rho^{-1}(F_{G1}(\alpha,\beta))$ =CN⌈CN$\left\lceil\rho^{-1}(\beta)\right\rceil$, PP$\left\lceil\rho^{-1}(\alpha)\right\rceil$⌉.

Examples:

- $\ell(\rho^{-1}(\mathrm{BETWEEN}(\mathrm{L}_2)(\mathrm{L}_1)(gintersection)))$ = *intersection between bar*$_2$ *and bar*$_1$
- $\ell(\rho^{-1}(\mathrm{BETWEEN}(\mathrm{A}(line))(\mathrm{L}_1)(gintersection)))$ = *intersection between a bar and bar*$_1$

T6$_{G\text{-}L}$ Compound adjectives

If $\alpha \in E_{\langle\langle e,t\rangle,\langle e,t\rangle\rangle}$ and $\beta \in E_{\langle e,t\rangle}$, then the translation of $F_{G1}(\alpha,\beta) = \alpha(\beta)$ is $\rho^{-1}(F_{G1}(\alpha,\beta))$ =ADJ⌈ADJ$\left\lceil\rho^{-1}(\beta)\right\rceil$, PP'$\left\lceil\rho^{-1}(\alpha)\right\rceil$⌉.

Examples:

- $\ell(\rho^{-1}(\mathrm{WITH}(\mathrm{AP})(gpainted)))$ = *painted with anticorrosive-paint*
- $\ell(\rho^{-1}(\mathrm{TO}(\mathrm{THE}(line))(gperpendicular)))$ = *perpendicular to the bar*

T7$_{G\text{-}L}$ *between* prepositional phrases

If $\alpha, \beta \in E_{\langle\langle e,t\rangle,t\rangle}$ and α, β are either L_i or $Q(line)$,[14]then the translation of $F_{G2}(\alpha, \beta)$ =BETWEEN$(\alpha)(\beta)$ is $\rho^{-1}(F_{G2}(\alpha, \beta))$ =PP$\lceil$between, T$\left\lceil\rho^{-1}(\alpha)\right\rceil$, and, T$\left\lceil\rho^{-1}(\beta)\right\rceil\rceil$.

Examples:

- $\ell(\rho^{-1}(\mathrm{BETWEEN}(\mathrm{L}_2)(\mathrm{L}_1)))$ = *between bar*$_2$ *and bar*$_1$
- $\ell(\rho^{-1}(\mathrm{BETWEEN}(\mathrm{A}(line))(\mathrm{L}_1)))$ = *between a bar and bar*$_1$

T8$_{G\text{-}L}$ *with* prepositional phrases

If $\alpha \in E_{\langle\langle e,t\rangle,t\rangle}$ and α is either AP, LP or $Q(paint)$, then the translation of $F_{G3}(\alpha)$ =WITH(α) is $\rho^{-1}(F_{G3}(\alpha))$ =PP'$\lceil$with, T$\left\lceil\rho^{-1}(\alpha)\right\rceil\rceil$.

Examples:

- $\ell(\rho^{-1}(\mathrm{WITH}(\mathrm{AP})))$ = *with anticorrosive-paint*
- $\ell(\rho^{-1}(\mathrm{WITH}(\mathrm{LP})))$ = *with luminous-paint*
- $\ell(\rho^{-1}(\mathrm{WITH}(\mathrm{SOME}(gpaint))))$ = *with some paint*

T9$_{G\text{-}L}$ *to* prepositional phrases

If $\alpha \in E_{\langle\langle e,t\rangle,t\rangle}$ and α is either L_i or $Q(line)$, then the translations of $F_{G4}(\alpha)$ = $\mathrm{TO}_a(\alpha)$ and $F_{G5}(\alpha)$ =$\mathrm{TO}_b(\alpha)$ are $\rho^{-1}(F_{G4}(\alpha))$ =PP'$\lceil$to, T$\left\lceil\rho^{-1}(\alpha)\right\rceil\rceil$, and $\rho^{-1}(F_{G5}(\alpha))$ = PP'$\lceil$to, T$\left\lceil\rho^{-1}(\alpha)\right\rceil\rceil$.

Examples:

- $\ell(\rho^{-1}(\mathrm{TO}_a(\mathrm{THE}(line))))$ = *to the bar*
- $\ell(\rho^{-1}(\mathrm{TO}_b(\mathrm{L}_5)))$ = *to bar*$_5$

3.3 States and the Coordinated Linguistic and Graphical Semantic Units

To solve problems like the ones posed at the beginning of this chapter and depicted in Figs. 3.1 and 3.3, an adequate representation of each state in the problem solution process is required. The logical language G, described above, serves this purpose.

How can a situation like the one depicted in Fig. 3.1(a), and repeated here in Fig. 3.9 for convenience, be represented? We use a *theory*, in the logical sense [Boolos & Jeffrey 1989], to represent the graphical and the non-graphical

[14]Where Q stands for any of the quantifiers: A, SOME, THE, or EVERY, defined in Table 3.3, and similarly in rules T8$_{G\text{-}L}$ and T9$_{G\text{-}L}$.

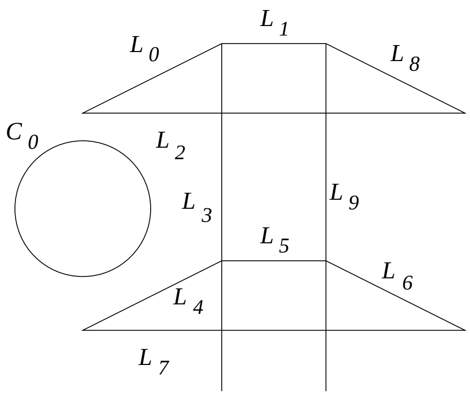


Figure 3.9: A state of interpretation

parts of the states. A state of interpretation will be represented by a *coordinated linguistic and graphical semantic unit* (COLGSU), which is a structure defined as follows:

Definition 1 *A* Coordinated Linguistic and Graphical Semantic Unit *(COLGSU) is a triplet,* $\langle T_0, T_1, \rho\rangle$*, consisting a graphical theory,* T_0*, a non-graphical theory,* T_1*, and a translation function,* ρ*, between the symbols of the logical language* G*, and the* natural *language* L*. A COLGSU represents a state of interpretation that can be given in a knowledge base.*

Thus, the graphical part of the initial state, 0, depicted in Fig. 3.9, can be represented by the following:

$$
\begin{array}{l}
\underline{\text{Graphical Theory } (T_0)} \\
d_0 = the\_dot(0.0)(8.0) \\
d_1 = the\_dot(4.0)(10.0) \\
d_2 = the\_dot(11.0)(8.0) \\
d_3 = the\_dot(7.0)(10.0) \\
d_4 = the\_dot(0.0)(2.0) \\
d_5 = the\_dot(4.0)(4.0) \\
d_6 = the\_dot(0.0)(5.0) \\
d_7 = the\_dot(7.0)(4.0) \\
d_8 = the\_dot(11.0)(2.0) \\
d_9 = the\_dot(4.0)(0.0) \\
d_{10} = the\_dot(7.0)(0.0) \\
l_0 = the\_line(d_0)(d_1) \\
l_1 = the\_line(d_1)(d_3) \\
l_2 = the\_line(d_0)(d_2) \\
l_3 = the\_line(d_1)(d_9)
\end{array}
\tag{3.3}
$$

$l_4 = the\_line(d_4)(d_5)$
$l_5 = the\_line(d_5)(d_7)$
$l_6 = the\_line(d_7)(d_8)$
$l_7 = the\_line(d_4)(d_8)$
$l_8 = the\_line(d_2)(d_3)$
$l_9 = the\_line(d_3)(d_{10})$
$c_0 = the\_circle(d_6)(2.0)$
$p_2 = path(l_4)(path(l_5)(path(l_6)(l_7)))$

The constants *the_dot*, *the_line*, *the_circle*, and *path* of G, denote geometrical functions that construct geometrical objects out from other simpler geometrical objects. For instance, the constant *the_line*, denotes a geometrical function that takes a dot as argument, say d_i, and and gives as result another function that taking another dot, d_j, as argument gives as result the line that starts in d_i and ends in d_j.[15]

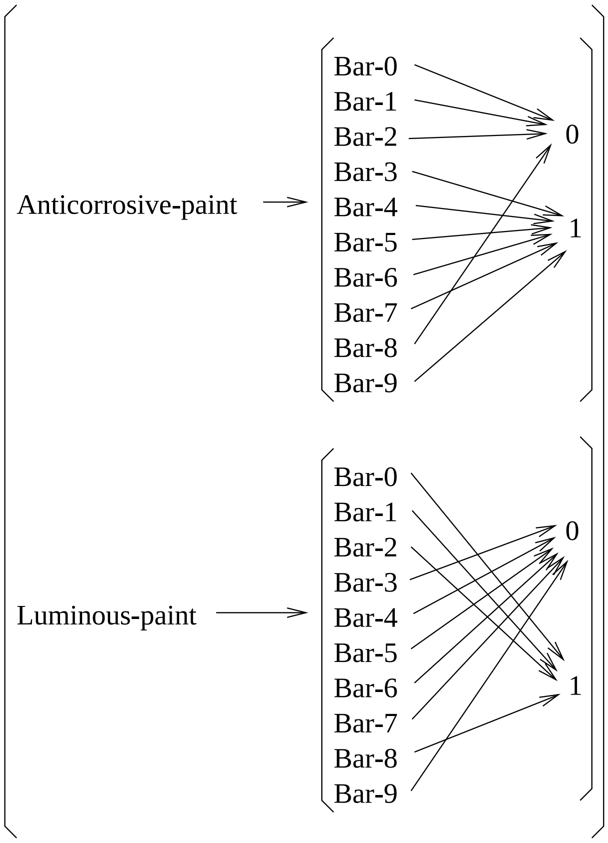
There is a large amount of information condensed in the graphical theory. A great number of facts corresponding to this representation can be asserted. In particular, facts such as those stated by the functions corresponding to the natural language phrases *"be horizontal"* and *"lie outside"* could be extracted from the geometrical information implicit in the graphical theory[16]. Therefore, there is no need to explicitly represent facts like these in the corresponding *non-graphical theory*. Nevertheless, there can be some information that is not expressible only graphically. For the example being examined, suppose that there are some bars that will be painted with anticorrosive paint, and some others with luminous paint[17], as stated by the following function which will be associated with the two-place predicate $painted\_with_*$ of language G:

[15] In this work the term *line* will always be used to refer to a segment of a straight line starting in a dot and ending in another dot, except when indicated differently.

[16] This operation, termed *predicate extraction* by Chandrasekaran (1998), consists of computing a given set of predicates from a spatial representation. In this thesis, the mechanism used to perform this operation is a scheme for the translation between the two modalities involved, i.e., the definition of the translation functions ρ and ρ^{-1} (see Fig. 3.4), and the definition of a geometrical model M_0.

[17] It could be argued that this information could be represented graphically, with a code of colours for instance. The geometrical model M_0 as defined above, however, does not include this attributive information, so in language G, the information about the class of paint used on each bar should be referred to model M_G.

$$[\![painted\_with_*]\!]^{M_G,i,g} = \quad (3.4)$$



It should be stressed that in previous expression, (3.4), such labels as 'Anticorrosive-paint', 'Bar-0', 'Bar-1', etc., actually mean the corresponding individuals in the world (W). To make this function part of the non-graphical theory, it can be specified in terms of the language G, as follows:

<u>Non-graphical Theory (T_G)</u> (3.5)

$painted\_with_*(luminous\_paint)(l_0)$
$painted\_with_*(luminous\_paint)(l_1)$
$painted\_with_*(luminous\_paint)(l_2)$
$painted\_with_*(anticorrosive\_paint)(l_3)$
$painted\_with_*(anticorrosive\_paint)(l_4)$
$painted\_with_*(anticorrosive\_paint)(l_5)$
$painted\_with_*(anticorrosive\_paint)(l_6)$
$painted\_with_*(anticorrosive\_paint)(l_7)$
$painted\_with_*(luminous\_paint)(l_8)$
$painted\_with_*(anticorrosive\_paint)(l_9)$
$\vdots$

3.3.1 Examples of semantic interpretation of natural language sentences

Example 1.-

Turning back to the design rule mentioned at the beginning of this chapter (p. 59), it can be translated into a constraint that can be expressed as the sentence

$$\textit{every bar lies outside the dielectric area} \tag{3.6}$$

The syntactic analysis of this sentence gives the following phrase structure tree:

$$\tau = \mathsf{t}\lceil \mathsf{T}\lceil \mathsf{T/CN}\lceil every\rceil, \mathsf{CN}\lceil bar\rceil\rceil, \\ \mathsf{IV}\lceil \mathsf{TV}\lceil lie\text{-}outside\rceil, \mathsf{T}\lceil \mathsf{T/CN}\lceil the\rceil, \mathsf{CN}\lceil dielectric\text{-}area\rceil\rceil\rceil\rceil \tag{3.7}$$

By translation rule $\mathbf{T1}_{L\text{-}G}$,

$$\rho(\tau) = \rho(\mathsf{T}\lceil \mathsf{T/CN}\lceil every\rceil, \mathsf{CN}\lceil bar\rceil\rceil) \\ (\rho(\mathsf{IV}\lceil \mathsf{TV}\lceil lie\text{-}outside\rceil, \mathsf{T}\lceil \mathsf{T/CN}\lceil the\rceil, \mathsf{CN}\lceil dielectric\text{-}area\rceil\rceil\rceil)) \tag{3.8}$$

By translation rule $\mathbf{T4}_{L\text{-}G}$,

$$\rho(\tau) = \rho(\mathsf{T/CN}\lceil every\rceil)(\rho(\mathsf{CN}\lceil bar\rceil)) \\ (\rho(\mathsf{IV}\lceil \mathsf{TV}\lceil lie\text{-}outside\rceil, \mathsf{T}\lceil \mathsf{T/CN}\lceil the\rceil, \mathsf{CN}\lceil dielectric\text{-}area\rceil\rceil\rceil)) \tag{3.9}$$

By translation rule $\mathbf{T0}_{L\text{-}G}$,

$$\rho(\tau) = \mathrm{EVERY}(line) \\ (\rho(\mathsf{IV}\lceil \mathsf{TV}\lceil lie\text{-}outside\rceil, \mathsf{T}\lceil \mathsf{T/CN}\lceil the\rceil, \mathsf{CN}\lceil dielectric\text{-}area\rceil\rceil\rceil)) \tag{3.10}$$

By translation rule $\mathbf{T2}_{L\text{-}G}$,

$$\rho(\tau) = \mathrm{EVERY}(line) \\ (\rho(\mathsf{TV}\lceil lie\text{-}outside\rceil)\ (\rho(\mathsf{T}\lceil \mathsf{T/CN}\lceil the\rceil, \mathsf{CN}\lceil dielectric\text{-}area\rceil\rceil))) \tag{3.11}$$

By translation rule $\mathbf{T0}_{L\text{-}G}$,

$$\rho(\tau) = \mathrm{EVERY}(line)(outside \\ (\rho(\mathsf{T}\lceil \mathsf{T/CN}\lceil the\rceil, \mathsf{CN}\lceil dielectric\text{-}area\rceil\rceil))) \tag{3.12}$$

By translation rule $\mathbf{T4}_{L\text{-}G}$,

$$\rho(\tau) = \mathrm{EVERY}(line)(outside \\ (\rho(\mathsf{T/CN}\lceil the\rceil)(\rho(\mathsf{CN}\lceil dielectric\text{-}area\rceil)))) \tag{3.13}$$

By translation rule $\mathbf{T0}_{L\text{-}G}$,

$$\rho(\tau) = \mathrm{EVERY}(line)(outside(\mathrm{THE}(circle))) \tag{3.14}$$

Developing the abbreviation EVERY (Table 3.3, p.78),

$$\rho(\tau) = \forall x[line(x) \to outside(\mathrm{THE}(circle))(x)] \tag{3.15}$$

By meaning postulate MP1 (p. 88),

$$\rho(\tau) = \forall x[line(x) \to (\mathrm{THE}(circle)(\lambda y[outside_*(y)(x)]))] \tag{3.16}$$

Developing the abbreviation THE (Table 3.3, p.78),

$$\rho(\tau) = \forall x[line(x) \to \exists u[\forall v[circle(v) \leftrightarrow u = v] \wedge \\ \lambda y[outside_*(y)(x)](u)]] \tag{3.17}$$

Finally,

$$\rho(\tau) = \forall x[line(x) \to \exists u[\forall v[circle(v) \leftrightarrow u = v] \wedge outside_*(u)(x)]] \tag{3.18}$$

The semantic interpretation of this expression is performed according to the rules given in Section 3.2.2.3 (p. 86). Since the expression is purely graphical, it suffices to consider model M_0 to find its semantic value: $[\![\rho(\tau)]\!]^{M_0,0,g}$. In order to explain adequately the semantic interpretation of expression (3.18), $\rho(\tau)$ can be rewritten in terms of its subexpressions as follows:

$$\left.\begin{array}{ll} (3.19.a) & \gamma = (circle(v) \leftrightarrow u = v) \\ (3.19.b) & \beta = \forall v[\gamma] \wedge outside_*(u)(x) \\ (3.19.c) & \alpha = line(x) \rightarrow \exists u[\beta] \\ (3.19.d) & \rho(\tau) = \forall x[\alpha] \end{array}\right\} \quad (3.19)$$

Note that all the language variables that partake in expressions (3.18) and (3.19), x, u, and v, are of type e. This means that each one ranges over the individuals of E, i.e., the geometrical objects, when interpreted with respect to the geometrical domain E.

Table 3.9 sumarises the semantic interpretation of expression (3.18) or its equivalent form given in expression (3.19). To visualise how this table is related to the semantic interpretation rules, the universally quantified expression (3.19.d) is to be reviewed. The interpretation of this expression starts in column 1 and ends in column 13 of Table 3.9. According to semantic rule 6, $[\![\rho(\tau)]\!]^{M_0,0,g} = [\![\forall x[\alpha]]\!]^{M_0,0,g} = 1$, iff for every individual $d \in D_e$ ($D_e = E$ in this case), $[\![[\alpha]]\!]^{M_0,0,g^{x/d}} = 1$. This interpretation involves the assignment function $g^{x/d}$ which is exactly like g except that it assigns the individual d to the variable x. The labels, with reference to Fig. 3.9, of the possible values for d that are to be assigned to variable x are set down in column 1. Each one of these assignments determines a different assignment function, namely, $g^{x/L_0}, g^{x/L_1}, g^{x/L_2}, \ldots$, etc. The interpretation of α, expression (3.19.c), starts in column 2 and ends in column 12. This expression is an implication whose antecedent is evaluated in column 2, and whose consequent is evaluated from columns 3 to 11. The consequent is an existential expression which is to be interpreted according to semantic rule 5. In this case, $[\![\exists u[\beta]]\!]^{M_0,0,g'} = 1$, iff for some individual $d \in D_e$, $[\![[\beta]]\!]^{M_0,0,g'^{u/d}} = 1$. Here, g' represents the current assignment function coming from the interpretation of the enclosing universally quantified expression, i.e., g' ranges over $g^{x/L_0}, g^{x/L_1}, g^{x/L_2}, \ldots$, etc. The labels for the possible values of d that are to be assigned to variable u are written down in column 3. In this column, an entry like $\neq C_0$, is used to symbolise any individual except the one labelled with C_0 in Fig. 3.9. Thus, each assignment of a value for d, determines a different assignment function such as $g^{x/L_0,u/L_0}, g^{x/L_0,u/L_1}, \ldots, g^{x/L_0,u/C_0}, \ldots g^{x/L_1,u/L_0}, \ldots$, etc. A similar procedure is observed to interpret the remaining subexpressions, with the following considerations:

1. The headings of columns are just mnemonic. They stand for the semantic value of the expression on them. For instance, the heading of column 2, $line(x)$, simbolises that the cells in this column are destined to hold the value $[\![line(x)]\!]^{M_0,0,g} = [\![line]\!]^{M_0,0,g}([\![x]\!]^{M_0,0,g})$.

1	2	3	4	5	6	7	8	9	10	11	12	13
$[\forall$												$\forall]$
x	*line* (x)	$[\exists$								$\exists]$	$\rightarrow$ α	
		u	$[\forall$				$\forall]$	$outside_*$ $(u)(x)$	$\wedge$ β			
			v	*circle* (v)	$u = v$	$\leftrightarrow$ γ						
L_0	1	$\neq C_0$	$\neq' C_0$	0	0	1	$\downarrow$	-	-	-	-	-
			C_0	1	0	0	0	-	-	-	-	-
		C_0	$\neq C_0$	0	0	1	$\downarrow$	-	-	-	-	-
			C_0	1	1	1	1	1	1	1	1	$\downarrow$
L_1	1	$\neq C_0$	$\neq' C_0$	0	0	1	$\downarrow$	-	-	-	-	-
			C_0	1	0	0	0	-	-	-	-	-
		C_0	$\neq C_0$	0	0	1	$\downarrow$	-	-	-	-	-
			C_0	1	1	1	1	1	1	1	1	$\downarrow$
			$\vdots$			$\vdots$			$\vdots$			$\vdots$
L_9	1	$\neq C_0$	$\neq' C_0$	0	0	1	$\downarrow$	-	-	-	-	-
			C_0	1	0	0	0	-	-	-	-	-
		C_0	$\neq C_0$	0	0	1	$\downarrow$	-	-	-	-	-
			C_0	1	1	1	1	1	1	1	1	$\downarrow$
any other	0	-	-	-	-	-	-	-	-	-	1	1

Table 3.9: Semantic interpretation of expression (3.18)

2. An entry like $\neq' C_0$ in column 4, is used to symbolise any individual different from any of the one labelled with C_0 or the one selected for variable u (column 3) in the same row.

3. The result for the double-implication, γ, enclosed in a universally quantified expression is given in column 7, the result for the *and* expression, β, enclosed in an existentially quantified expression is given in column 10, and the result for the implication, α, enclosed in a universally quantified expression is given in column 12.

4. A down-arrow, $\downarrow$, in a cell is used to indicate that further interpretation of an enclosed subexpression is required to determine the semantic value of the enclosing expression.

The semantic value of the whole expression is found in the last cell of column 13, and it is 1. This signals the initial state, 0, as *consistent*, from the tower design point of view, i.e., the design rule is not violated in this state.

Summarising, in the examined example the *coordinated linguistic and graphical semantic unit* for the initial state (0) is defined by the conjoined representational structures corresponding to the graphical theory T_0, described in expression (3.3), the non-graphical (or *linguistic*) theory T_G, described in expression (3.5), and the translation function ρ, described in Section 3.2.3.

Example 2.-

Consider now the sentence,

every bar is painted with luminous paint

The analysis of this sentence gives the following phrase-structure tree:

$$\tau = \mathsf{t}\lceil \mathsf{T}\lceil \mathsf{T/CN}\lceil every \rceil, \mathsf{CN}\lceil bar \rceil\rceil, \mathsf{IV}\lceil \mathsf{IV/ADJ}\lceil be \rceil, \mathsf{ADJ}\lceil \mathsf{ADJ}\lceil painted \rceil, \mathsf{PP'}\lceil \mathsf{with}, \mathsf{T}\lceil luminous\_paint \rceil\rceil\rceil\rceil\rceil \tag{3.20}$$

Applying the translation rules in a similar way than in the previous example, the translation of this tree into G is given by:

$$\rho(\tau) = \mathrm{EVERY}(line)(\mathrm{BE}_b(\mathrm{WITH}(\mathrm{LP})(painted))) \tag{3.21}$$

which is an abbreviated form of

$$\rho(\tau) = \forall x[line(x) \to painted\_with(\lambda P[P(luminous\_paint)])(x)] \tag{3.22}$$

In order to interpret this expression, it have to be transformed using meaning postulate MP1 (p. 88). The transformed expression is:

$$\rho(\tau) = \forall x[line(x) \to painted\_with_*(luminous\_paint)(x)] \tag{3.23}$$

Since neither $painted\_with_*$ nor $luminous\_paint$ have a geometrical interpretation, expression (3.23) have to be interpreted with respect to model M_G. The semantic interpretation of this expression according to model M_G is summarised on Table X. The following considerations apply to this table:

1. '≠Bar-*i*' in column 1 means any value different than an individual bar.

2. 'Bar-*i*' in column 1, where *i* is an integer number, means the individual of the world known as the bar *i*.

3. The heading '$line(x)$' in column 3 signals that each cell in that column is destined to hold the semantic value of that expression, i.e., $[\![line(x)]\!]^{M_G,0,h} = Bar([\![x]\!]^{M_G,0,h})$, where Bar is a function that takes an individual as argument and returns 1 in case such an individual is a bar, and 0 otherwise.

4. The semantic value of $painted\_with_*$ is the function defined in the non-graphical theory, and given in expression (3.5).

Expression (3.23), then, evaluates to 0. This means that the original sentence *every bar is painted with luminous paint* is false according to model M_G.

1	2	3	4	5
[$\forall$				$\forall$]
x	$line(x)$	$painted\_with_*$ $(luminous\_paint)(x)$	$\rightarrow$ α	
$\neq$Bar-i	0	any value	1	$\downarrow$
Bar-0	1	1	1	$\downarrow$
Bar-1	1	1	1	$\downarrow$
Bar-2	1	1	1	$\downarrow$
Bar-3	1	0	0	0

Table 3.10: Semantic interpretation of expression (3.23)

3.4 Treatment of Emergent Objects

Part of the motivation of the material developed in this chapter was the problem of *emergent objects* addressed in Section 2.6, starting on p. 56. An emergent object will be defined as a geometrical object that emerges or appears from the geometrical configuration in a given graphical depiction, e.g., the dot that emerges from the intersection between two lines, or the rectangle that emerges from the overlapping of two rectangles, etc. The graphical language semantics defined in this chapter provides a mechanism to deal with these objects, specially in the context of quantified expressions. The following subsections describe the problem and its solution with the aid of illustrative examples. Sections 3.4.1 and 3.4.2 describe the problem, and section 3.4.3 presents a solution with the aid of an example.

3.4.1 Quantification domain

In the previous chapter, the following problem was posed: considering the situation depicted in Fig. 3.10, where the black boxes represent dots that are actually defined in the knowledge base, and the white boxes represent dots that have emerged from the geometrical configuration, a user asks the system *"What is the nearest dot to the black ball within a distance of 50 units?"*.

This question can be reformulated as the quantified expression[18]:

$$\exists v_0(\epsilon(v_0) \wedge \forall v_1(\epsilon(v_1) \rightarrow (\delta(v_0) \leq \delta(v_1)))) \tag{3.24}$$

where v_0 and v_1 are variables that range over geometrical objects, $\delta(dot)$ is a function whose value is the distance from the given argument to the black ball, and $\epsilon(dot)$ is a predicate which is 1 if the given argument is a dot which is inside the circle whose center is the black ball and with a radius of 50 units, and it is 0 otherwise.

[18]This formulation will be justified later in the text.

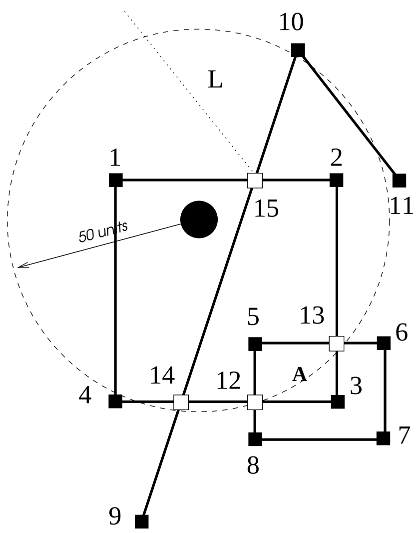


Figure 3.10: What is the nearest dot to the black ball?

Given such an expression, the problem is to determine the values of the quantified variables, v_0 and v_1, that should be considered. This problem will be termed here as the *quantification domain problem*. To understand this problem, suppose the following knowledge base, representing a state of interpretation, whose graphical representation is depicted in Fig. 3.11:

$$
\begin{aligned}
&d_1 = the\_dot(5.75)(6.15)\\
&d_2 = the\_dot(7.80)(7.20)\\
&d_3 = the\_dot(8.0)(5.5)\\
&d_4 = the\_dot(9.00)(8.20)\\
&d_5 = the\_dot(10.60)(4.10)\\
&d_6 = the\_dot(9.95)(7.95)\\
&d_7 = the\_dot(11.85)(7.20)\\
&d_8 = the\_dot(8.80)(6.15)\\
&l_1 = the\_line(d_1)(d_2)\\
&l_2 = the\_line(d_1)(d_8)\\
&l_3 = the\_line(d_2)(d_5)\\
&l_4 = the\_line(d_3)(d_4)\\
&l_5 = the\_line(d_2)(d_7)\\
&l_6 = the\_line(intersection\_between_*(l_4)(l_5))(d_6)\\
&l_7 = the\_line(d_7)(d_8)\\
&l_8 = the\_line(intersection\_between_*(l_4)(l_5))(the\_dot(10.15)(6.80))
\end{aligned}
$$

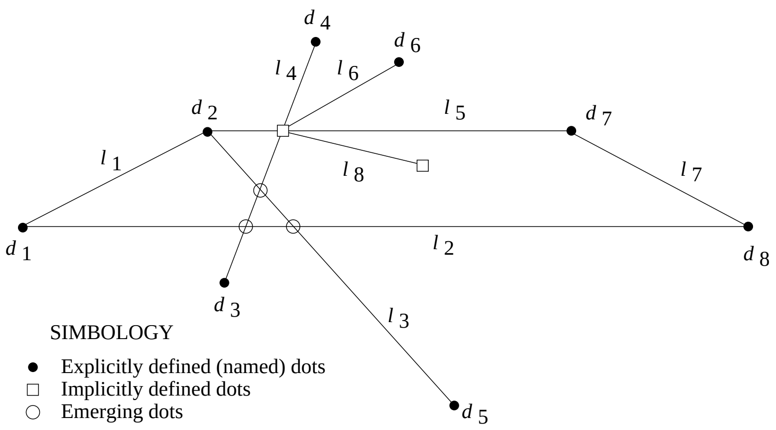


Figure 3.11: Quantifying over dots of different kinds

What are the dots with a semantic meaning for the quantified expression (3.24) given above? There is an infinite number of dots in the Euclidian space where the graphic representing the knowledge base lies. Obviously, these are not the object of the quantification. Every line segment consists of an infinite number of dots. It could be possible, in certain circumstances, to regard all these dots as the object of the quantification. However, it is better, in this case, to transform the quantified formula into another in terms of the finite set of line segments, than to deal directly with the uncountable set of the dots composing the line segments. Thus, this work will restrict the determination of the quantification domain to finite sets. Then, what are, in this case, the dots that appear to be *interesting* from the quantification point of view? The given knowledge base and its graphical representation, shown in Fig. 3.11, exhibit dots of three different kinds, as follows:

1. **Explicitly defined (named) dots**. These are the dots that have an explicit name (symbol) on the left hand side of the facts asserted in the knowledge base, e.g., d_3. In the given example, there are eight of these dots, depicted in Fig. 3.11 as black balls.

2. **Implicitly defined dots**. These are dots, that are not named, in the left hand side of the knowledge base facts, but are mentioned somewhere in the right hand side of the knowledge base facts, e.g., $the\_dot(10.15)(6.80)$ in the last row of the knowledge base. In the example, there are two of these dots, shown in Fig. 3.11 as white squares.

3. **Emergent dots**. These dots are not mentioned anywhere in the knowledge base. However, they are perceived as individual dots which *emerge* from the

mode of the geometrical combination of the line segments in the graphical representation of the knowledge base. In the case that is being examined, the intersections of the line segments are the typical emerging dots. In the example, there are three of these dots, represented in Fig. 3.11 as white balls (the intersection of the segments l_4 and l_5 could also be included here, though it was in the kind of *implicitly defined dots*).

3.4.2 Quantifiable and emergent objects

In a quantification process, dots of the three kinds are candidates to be taken into account. Yet, the application or an end-user, by means of the semantic value attached to every kind of dot, is the sole responsible for deciding whether that kind has to be taken into account in the quantification process. The system, however, has to provide mechanisms to facilitate this task. Here, some principles in this direction are established.

A quantification scheme that takes into account only those dots named on the left-hand side of the knowledge base facts, can be summarised as the following constrained quantified expressions:

$$\forall v_1(\mathsf{named}(v_1) \rightarrow p(v_1)))$$
$$\exists v_1(\mathsf{named}(v_1) \wedge p(v_1))) \qquad (3.25)$$

where named is a predicate that takes a geometrical object and returns 1 if the object has an explicit name in the knowledge base, and 0 otherwise.

The emergent objects are then defined more precisely as those that, from the user's point of view, are there in the graphical representation of the knowledge base, even though they were not explicitly defined nor mentioned in the knowledge base. To capture these objects, the language must be able to express them, i.e., there must be at least one operator that, taking the combining objects as arguments, produces an emerging object. For instance, in the case of the emerging dots cited above, the operator $intersection\_between_*$ plays this role, since it takes any two line segments and produces, if possible, a dot. Let operators that have this ability be called *emerger operators*.

3.4.3 Examples

Where does the necessity of taking into account emerging objects arise? Are there any situations in which these objects need to be quantifiable? In this section, an example based on the classical technique of graphical *gravity* fields [Newman & Sproull 1979] is developed.

Suppose that, using the same knowledge base specified above and graphically depicted in Fig. 3.11, a user positions a graphical cursor in a certain location and he or she wants to attach some graphical object, say a new line, to the nearest dot to the cursor within a sensible square area around it, as shown in Fig. 3.12.

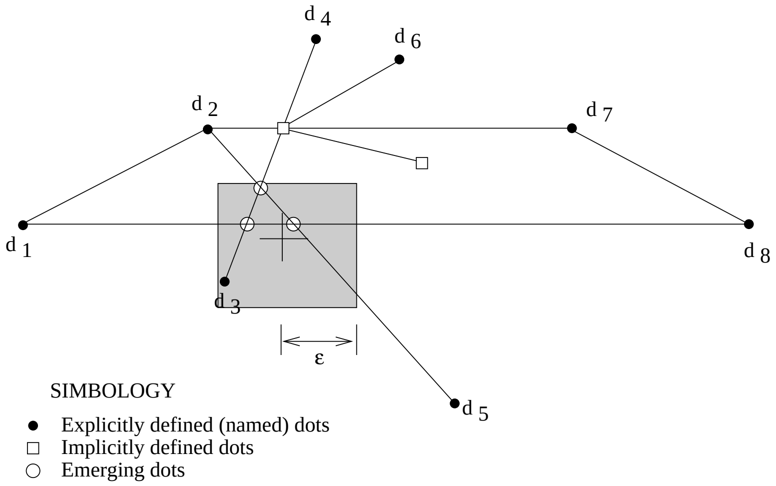


Figure 3.12: Graphical *gravity* fields

In this case, the user is interested not only in line segment ending dots but also in those emerging as the result of intersection between line segments.

A first-order expression for the problem can be obtained in two parts: (1) to find a formula that picks the nearest dot, irrespective of whether it is inside or outside the sensible area, and (2) to constrain the formula to consider only those dots inside the sensible area:

$$\exists v_0(\epsilon(v_0) \wedge \forall v_1(\epsilon(v_1) \rightarrow (\delta(v_0) \leq \delta(v_1)))) \tag{3.26}$$

Where δ is a function that takes as argument a *dot*, d, and gives as its result the distance from the cursor to the *dot*, d, and ϵ is a predicate that takes any geometrical object, d, as argument, and $\epsilon(d)$ is true iff d is a dot that lies in the sensible area. In this example, provided there are dots inside the sensible area, the result is not affected if the sensible area is not considered.

Consider now a more interesting example which consists of investigating which dot is the furthest away from the cursor within the sensible area. The expression in this case is:

$$\exists v_0(\epsilon(v_0) \wedge \forall v_1(\epsilon(v_1) \rightarrow (\delta(v_0) \geq \delta(v_1)))) \tag{3.27}$$

The following section shows how such expressions are semantically evaluated.

3.4.3.1 Semantic interpretation

To interpret the formula given in expression (3.27), the semantic interpretation rules given in Section 3.2.2.3 (p. 87) have to be considered. These rules are summarised in the following scheme where $\mathcal{Q}$ is either $\forall$ or $\exists$:

> If $\mu \in V_s$ and $\beta \in E_t$, then $[\![\mathcal{Q}\mu(\beta)]\!]^{M_0,i,g}$ is 1 iff for some $d \in D_s$, in the case $\mathcal{Q}$ is $\exists$, or for every $d \in D_s$, in the case $\mathcal{Q}$ is $\forall$, $[\![\beta]\!]^{M_0,i,g^{\mu/d}} = 1$.

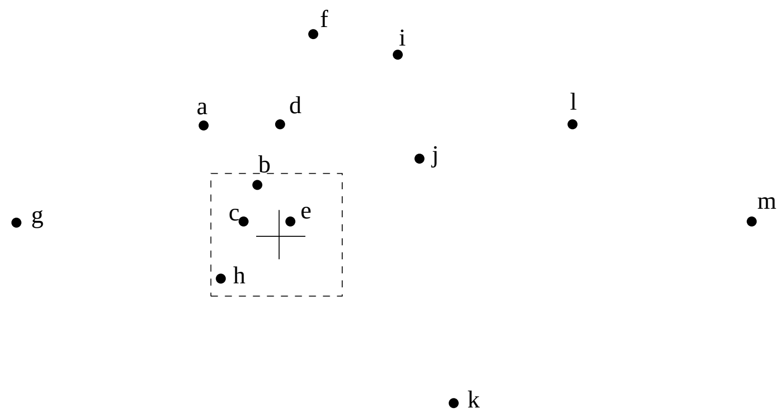


Figure 3.13: A complex of dots

Considering all the dots of the example, and supposing that the actual dots are named as shown in Fig. 3.13, the semantic interpretation, according to the rules given above, can be summarised in Table 3.11, where $\alpha = \epsilon(v_1) \rightarrow (\delta(v_0) \geq \delta(v_1))$ and $\beta = \epsilon(v_0) \wedge \forall v_1(\alpha)$, which are the subexpressions that require to be evaluated in order to find the semantic value of expression (3.27).

From this table, it can be seen that the unique dot for which all expressions α are evaluated to 1 is the dot labelled with 'h'. Under these circumstances, the whole quantified expression (3.27) is evaluated to 1, and the variable v_0 finally holds the value of 'h', which effectively names the furthest dot to the cursor within the sensible area.

3.4.3.2 Discovering the quantification domain

To discover the quantification domain, the application (or the user) has to define a set of *emerger operators*. In the case that is being examined, the quantification domain holds elements of type *dot*, and the set of *emerger operators* contains only one element: the operator $intersection\_between_*$.

The starting set of dots contains ten dots: the eight explicitly defined dots, black balls in Fig. 3.11, and the two implicitly defined dots, white squares in the figure.

To discover the additional emergent dots, a cross product between the sets containing the objects of types required as arguments of the *emerger operator* $intersection\_between_*$, has to be performed. In this case, both arguments share the same set: the set of line segments. This set is initially composed of all line segments explicitly and implicitly defined in the knowledge base. In this example, the set is limited to the eight named line segments. The cross product for the $intrsection\_between_*$ operator is the set of segment lines by itself.

$[\exists$								$\exists]$
v_0	$\epsilon(v_0)$	$[\forall$				$\forall]$	β	
		v_1	$\epsilon(v_1)$	$\delta(v_0) \geq \delta(v_1)$	α			
a	0	any	-	-	-	-	0	$\downarrow$
b	0	any	-	-	-	-	0	$\downarrow$
c	1	a	0	-	1	$\downarrow$		
		b	1	0	0	0	0	$\downarrow$
d	0	any	-	-	-	-	0	$\downarrow$
e	1	a	0	-	1	$\downarrow$		
		b	1	0	0	0	0	$\downarrow$
f	0	any	-	-	-	-	0	$\downarrow$
g	0	any	-	-	-	-	0	$\downarrow$
h	1	a	0	-	1	$\downarrow$		
		b	1	1	1	$\downarrow$		
		c	1	1	1	$\downarrow$		
		d	0	-	1	$\downarrow$		
		e	1	1	1	$\downarrow$		
		f	0	-	1	$\downarrow$		
		g	0	-	1	$\downarrow$		
		h	1	1	1	$\downarrow$		
		i	0	-	1	$\downarrow$		
		j	0	-	1	$\downarrow$		
		k	0	-	1	$\downarrow$		
		l	0	-	1	$\downarrow$		
		m	0	-	1	1	1	1

Table 3.11: Semantic interpretation of expression (3.27)

	l_1	l_2	l_3	l_4	l_5	l_6	l_7	l_8
l_1	e	e	e	e	e	e	e	e
l_2		e	○	○	e	e	e	e
l_3			e	○	e	e	e	e
l_4				e	○	e	e	e
l_5					e	e	e	e
l_6						e	e	e
l_7							e	e
l_8								e

Table 3.12: Discovering the quantification domain

Each element of the cross product has to be passed as argument to the *intersection_between*$_*$ operator to see whether it produces an emergent dot. Table 3.12 summarises this task. Here, the letter e stands for an unsuccessful application of the operator where the argument line segments do not intersect. The symbol ○ stands for an intersection between the argument line segments. Note that the intersection between l_4 and l_5 has also been detected here, though this dot was already recorded in the starting set of ten dots. Since the *intersection_between*$_*$ operator is commutative the resulting matrix is symmetric.

The operation performed does not alter the set of lines. Therefore it is not necessary to perform additional operations, and so the final quantification domain is the union of the initial set of the ten dots and the set of four dots found in Table 3.12.

When more than one *emerger operator* is involved, the process has to be repeated for each operator, and for each new emergent object that results. This leads to the following algorithm for computing the set of emergent objects.

STEP 1. The initial sets of objects must include both explicitly and implicitly defined objects.

STEP 2. Make a list containing all *emerger operators*.

STEP 3. From the list of *emerger operators* take one, and call it **the operator**.

> **STEP 4.** Make the cross product of the sets of objects of the types required by the arguments of **the operator**, i.e., every element of this cross product can be passed as argument to **the operator**. From now on this cross product will be called **the table**, and each element of it will be called **an entry**.
>
> **STEP 5.** Take in order an **entry** of **the table** and call it **the entry**.
>
> > **STEP 6.** Pass **the entry** to **the operator** as its argument.

STEP 7. If this operation results in a new object add it to its corresponding set according to its type, i.e., update the set with this new element. Otherwise do nothing.

STEP 8. Return to STEP 5 (i.e., take another **entry**).

STEP 9. Return to STEP 3 (i.e., take another **operator**)

STEP 10. Make a new list of *emerger operators* consisting of those operators whose arguments belong to a set that has been updated in STEP 7.

STEP 11. If the list is not empty go back to STEP 3 with this new list. Otherwise, the task is finished.

This algorithm computes the inductive closure [Gallier 1988] of the basic objects and the set of *emerger operators* of the graphical language. In order to avoid an explosive proliferation of new objects the set of *emerger operators* has to be defined very carefully.

To conclude this section, it should be said that common multimodal explanation systems cannot refer to emergent objects. Having provided the possibility of introducing these objects into *quantification domains*, so providing a straightforward way to interpret them, the work proposed here enables emergent objects as referable objects. Since each of these objects is not named by a single symbol of the language but by a valid composed expression of the language instead, the introduction of deictic expressions seems more natural to refer to them. For instance, to identify the intersecting dot of the two continuous lines in Fig. 3.14, this approach would have generated a phrase like *"this dot"*, or *"this intersection"*, while highlighting the dot.

Having at hand a set of emergent objects produced by the algorithm developed in this thesis, another approach to single out one such objects could be to generate an appropriate referring expression using an algorithm like the one developed by Dale & Reiter (1995). With the adequate translations of some geometrical properties into attribute-value pairs, their algorithm would render something like: *"the dot that is the intersection between two lines"*, or the shorter phrase: *"the intersection between two lines"*, or simply: *"the intersection"*, to refer to the intersecting dot between the continuous lines in Fig. 3.14, provided they were the only intersecting lines on the display. However, if there were more than one intersection, a more detailed description would have been generated. Considering the dashed lines in the situation depicted in Fig. 3.14, and given the emergent objects, Dale & Reiter's algorithm could be used to generate an expression like: *"the intersection on the left"*.

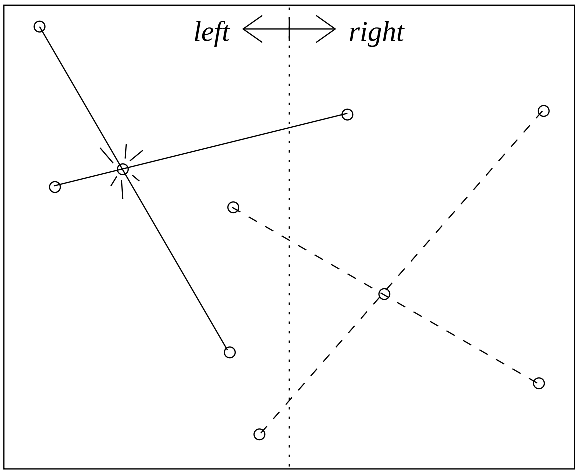


Figure 3.14: Referring to an emergent dot

3.5 Summary

Two languages G and L have been defined in this chapter. Language G is a logical language related to the graphical modality through model M_0, and to the *linguistic* modality through model M_G. A translation relation, ρ, allows for translations from natural language expressions (of language L) into G, and its inverse, ρ^{-1}, allows to translate some expressions of G into L. Both translation relations have also been described in this chapter. It has been shown that this framework, based on the languages L and G, the models M_G and M_0, and the translation relation ρ, provides an adequate ground for representing and interpreting multimodal information.

Part of the formal presentation of languages L and G was inspired in the work by Pineda & Garza (1998). However, in this work the definition of language L uses phrase structure grammar rules instead of Montague's style syntactic formation rules. The geometrical domain, as a representational object, is treated here just as a set of geometrical objects, while in [Pineda & Garza 1998], this domain is regarded as a more structured language. In this thesis, all interpretation of the natural language L is achieved through the semantic interpretation of its translated expressions into G, while in [Pineda & Garza 1998] the interpretation of some basic constants of L is directly achieved in L. In this approach, G is interpreted according to any of two models, one for each modality, while in [Pineda & Garza 1998] there is one model for each language.

A collateral problem that it is worth mentioning here is that of the occlusion of geometrical objects. This problem occurs when two objects, say two lines, coincide in the geometrical space[19]. To avoid ambiguities coming from this phenomenon, in this work these objects are treated as separate geometrical objects that incidentally happen to have the same depiction. Thus, a function F_E is defined as an one-to-one function from E into W, and more precisely, F_E is an one-to-one function from E onto B (see Fig. 3.5 p. 68), $B \subseteq W$ that relates every geometrical object in E with the unique individual it represents in B[20].

The two languages L and G along with the translation relation ρ provide the propositional contents for the speech acts and the rhetorical structures used in the generation of multimodal explanations. The two forthcoming chapters are dedicated to expose how multimodal explanations are generated from speech acts and rhetorical and intentional structures.

[19]This is a contingent problem that arises from the fact that E abstracts the world space as a projection to a 2D space, since two different lines in a 3D space might have the same projection in a 2D space. Had we chosen E as a 3D space, this problem would not have appeared at all.

[20]Formally, to cope with this problem, E is redefined as $E \subseteq \Im \times P(\Re^2) \cup \Re$, where $\Im$ is the set of integer numbers, so that, now a member of E can be either a real number or a pair $\langle i, s \rangle$, where the integer i is used as a unique identifier of the object, and s is a set of dots used to describe its geometry.

Chapter 4

SPEECH ACTS, AND A SCHEME FOR TRANSLATION FROM GRAPHICAL TO LINGUISTIC OBJECTS

All explanations generated in design oriented dialogues, are ultimately rendered as speech acts. In the context of this thesis, the explanations include both textual and graphical output. Fig. 4.1 shows a typical explanation. The generation of such explanations requires to operationalise speech acts to cope with both modalities and be capable of handling deictic references. This chapter presents the operationalisation developed in this thesis, which provides the speech act operators utilised in the next chapter for the generation of explanations.

4.1 Speech Acts

In Chapter 3, the translation relation ρ, between languages L and G was specified. This relation is embodied in a feature, named lform, of the PATR grammar rules for language L specified in Section A.3.2 of Appendix A. Here it is shown how such feature is utilised to generate natural language text.

The basic units in G that are required to be translated into L are propositions. Having in mind that the purpose of these translations is the generation of explanations, they make sense only in the context of *speech acts*. A speech act is an act that a speaker performs in uttering a sequence of words. For instance, in uttering the sequence "*The bar is painted*" the speaker is *asserting* that a certain bar is painted, while if he utters "*Is the bar painted?*", he is *asking* if that bar is painted, and if he says "*the bar must be painted*", he is *specifying* that such a bar must be painted. Here, *asserting*, *asking*, and *specifying* are speech acts. Searle

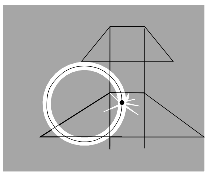
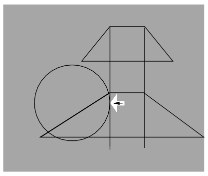
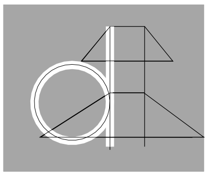
TEXT	GRAPHICS
System: (• *Highlighting the circle and the starting point of the displacement*) I moved this dielectric-area from here ...	
System: (• *Highlighting the ending point of the displacement and showing the new postion of the circle*) ... to here, so that ...	
System: (• *Highlighting line l_1 and the circle*) ... now this bar lies outside this dielectric-area.	

Figure 4.1: A typical explanation in a design dialogue

Implicit IA	Explicit IA	F	R	$p = P(R)$	$F(p)$
The bar is painted.	*I assert that the bar is painted.*	assert			assert(p)
Is the bar painted?	*I ask if the bar is painted.*	ask	*The bar* $\downarrow$ bar_5	$painted(bar_5)$	ask(p)
The bar must be painted.	*I specify that the bar must be painted.*	specify			specify(p)

Table 4.1: Decomposition of some simple illocutionary acts (IA)

(1969) distinguished three distinct kinds of acts that occur when someone utters a sentence, namely:

1. Performing *utterance acts*: uttering of words (morphemes, sentences).
2. Performing *propositional acts*: referring and predicating.
3. Performing *illocutionary acts*: stating, questioning, asserting, commanding, requesting, promising, etc.

By making the distinction between propositional acts and illocutionary acts, Searle isolated the illocutionary act (IA) from its propositional content. This allows a decomposition of the illocutionary acts as shown in Table 4.1. In this table, F stands for the illocutionary force of the acts, R stands for referential act, P stands for a predication, p, which is a functional composition of R and P, stands for the propositional acts embedded in the illocutionary acts. When uttered by a speaker within an illocutionary act, a referring expression has the purpose of identifying an object or objects of the world. In the examples presented here, the referring expression "*The bar*" is associated with an individual of the world, which in the table is simply symbolised with "bar_5". Note that, for the three examples shown, the referential act is the same, and the propositional act is the same as well, while it is the illocutionary force indicator which makes them different.

When elaborating an explanation about model for design, the propositions of language G provide the main source for the propositional content of the speech acts constituting the explanation. Some of these propositions can be graphical in nature, others can be purely linguistic, and still others can be a combination of both. In what follows, the translation from G into L is examined in the context of the production of speech acts.

4.1.1 One-argument predicates

Let $p(a)$ be a proposition of G. For the assert speech act, this proposition will be translated into language L as follows:

$$\text{assert}(p(a)) = \mathsf{t}[p(a)] \tag{4.1}$$

where, $\mathsf{t}[lform]$ stands for the generation of a sentence according to the given logical form *lform*, $p(a)$ in this case. The grammar rules and the procedure by which this generation is carried out is described in Section A.3 of Appendix A. For instance, given the translation relation defined in Section 3.2.3 (p. 88), the graphical predicate $ghorizontal(l_5)$ will be directly translated as:

$$\mathsf{t}[ghorizontal(l_5)] \tag{4.2}$$

which renders the following text:

$$\text{“}bar_5 \textit{ is horizontal}\text{”} \tag{4.3}$$

For the specify speech act, the proposition $p(a)$ will be translated into L as follows:

$$\text{specify}(p(a)) = \mathsf{t}[p(a), \mathsf{mdflist}(gmust)] \tag{4.4}$$

where $\mathsf{mdflist}(gmust)$ indicates that the verb of the produced sentence will be modified by the auxiliary verb *must*. For instance, $\text{specify}(ghorizontal(l_5))$ will be resolved as follows:

$$\mathsf{t}[ghorizontal(l_5), \mathsf{mdflist}(gmust)] \tag{4.5}$$

which renders the following text:

$$\text{“}bar_5 \textit{ must be horizontal}\text{”} \tag{4.6}$$

While *assertions* enable direct translations from logical language G to L, other speech acts provided here, e.g. *specify*, allow for a wider spectrum in the sentences that can be generated.

4.1.2 Two-argument predicates

Propositions of the form $p(a_2)(a_1)$, are translated into subject-object sentences of the language L. For the assert speech act, this propostion is translated as follows:

$$\text{assert}(p(a_2)(a_1)) = \mathsf{t}[p(a_2)(a_1)] \tag{4.7}$$

and for the specify speech act:

$$\text{specify}(p(a_2)(a_1)) = \mathsf{t}[p(a_2)(a_1), \mathsf{mdflist}(gmust)] \tag{4.8}$$

Then, for instance, the speech act $\text{specify}(outside(c_2)(p_1))$ is resolved as

$$\mathsf{t}[outside(c_2)(p_1), \mathsf{mdflist}(gmust)] \tag{4.9}$$

which renders the following text:

$$\text{“}crossarm_1 \ must \ lie \ outside \ dielectric\text{-}area_2\text{”} \tag{4.10}$$

4.1.3 Quantified expressions

The translation scheme presented in Section 3.2.3 (p. 88) of the previous chapter, yields room for managing quantified expressions in a natural way. Consider that for proposition $\mathcal{P}$ of G, provided that it is translatable into L, the assert and specify speech acts can be assembled as follows

$$\mathrm{assert}(\mathcal{P}) = \mathsf{t}[\mathcal{P}] \tag{4.11}$$

$$\mathrm{specify}(\mathcal{P}) = \mathsf{t}[\mathcal{P}, \mathsf{mdflist}(gmust)] \tag{4.12}$$

There is no constraint for proposition $\mathcal{P}$ to include quantified expressions. For instance, the expression EVERY(*line*)(BE_b(*ghorizontal*)) can be asserted as follows.

$\mathrm{assert}(\mathrm{EVERY}(line)(\mathrm{BE}_b(ghorizontal))) =$
$\mathsf{t}[\mathrm{EVERY}(line)(\mathrm{BE}_b(ghorizontal))]$

where EVERY and BE_b abbreviate the expressions stated in Table 3.3 (p. 78). Now, the expression to be translated, which is a valid language G expression[1], is handed over to the generator of language L to produce the sentence:

every bar is horizontal

Following this line, the scheme for the translation of more complex quantified expressions is possible. For instance, the speech act: specify(EVERY(*line*)(*outside*(EVERY(*circle*)))), is translated to:

$$\mathsf{t}[\mathrm{EVERY}(line)(outside(\mathrm{EVERY}(circle))), \mathsf{mdflist}(gmust)] \tag{4.13}$$

which renders the following text:

every bar must lie outside every dielectric area (4.14)

The speech act specify(EVERY(*circle*)(*r_outside*(EVERY(*line*)))), which is similar but with different quantifiers' order, is translated to:

$$\mathsf{t}[\mathrm{EVERY}(circle)(r\_outside(\mathrm{EVERY}(line))), \mathsf{mdflist}(gmust)] \tag{4.15}$$

which, when handed over to the NL generator, renders a different sentence:

for each dielectric area every bar must lie outside it[2] (4.16)

Likewise, the NL grammar (Appendix A, p.200) includes rules to generate the following:

---

[1]This holds for assertions but not for most other speech acts.

[2]In fact, this text was produced by grammar rule A.94 (Appendix A, p.214), while the text in expression (4.14) was generated by grammar rule A.1 (Appendix A, p.200).

1. From assert(EVERY(*line*)(*outside*(SOME(*circle*)))),
   every bar lies outside some dielectric areas (grammar rule A.1, Appendix A, p.200).

2. From assert(SOME(*circle*)(*r_outside*(EVERY(*line*)))),
   there is some dielectric area such that every bar lies outside it (grammar rule A.93, p.214).

4.1.4 Reference and deictic acts

The above discussion provides a method for the generation of speech acts from graphical propositions. Not a word has been said about the referring expressions embedded in such speech acts so far. Take, for instance, expression (4.6), which is repeated here for convenience

$$\text{“}\mathit{bar}_5 \textit{ must be horizontal}\text{”} \tag{4.17}$$

The referring expression this contains is the name “bar_5”, and by uttering it, it is intended that the hearer identifies the world object (in the graphical representation in this case) that is referred to by that expression. Unless the graphical model consists only of a few objects or all of them are identified in the graphical representation by some mechanism, say by labelling them, it is unlikely that the hearer could identify the object referred to by such an expression. Furthermore, the labelling of all graphical objects is not practical for graphical models of a medium size. A strategy could be to label only those objects mentioned in the issued speech act, expect that the hearer could infer the link between the linguistic and the graphical symbols, and, once this effect is achieved, delete the labels. However, this approach is disadvantageous when dealing with quantified expressions and with plural noun phrases.

In this thesis, a different approach is explored, namely: the inclusion of *deictic expressions* as the referring expressions of the speech acts. A *deictic expression*, of the kind developed here, involves the combination of two acts: the performer utters a demonstrative pronoun (this, that, those, etc.), or a demonstrative adjective followed by a common name (this table, those spies, etc.), while he or she points (or physically indicates towards) the object that he or she intends to refer to, allowing a direct introduction of referents into the discourse. To simulate this kind of act in a multimodal system of the sort described here, the pointing action is implemented by a graphical device such as highlighting, encircling, or indicating with an arrow, etc., the graphical object or objects that the act is intending to identify.

To manage deictic actions that combine the languages G and L, the concept $deictic(a)$ will be introduced as follows:

$$deictic(a) = \left\{ \begin{array}{c} highlight(a) \\ \mathsf{T}[gthis(GtypeOf(a))] \end{array} \right\} \tag{4.18}$$

where a is an expression of G, and $highlight(g)$ is a expression of the language G which denotes an object similar to the one denoted by the graphical expression g except that its graphical depiction changes in an attribute which can be the colour[3], and $GtypeOf(g)$ is a function that returns the geometrical type of its argument, e.g., *dot*, *line*, *circle*, etc.

The introduced function, *deictic*, extend the linguistic acts to what Maybury (1993*a*) has called communicative and rhetorical acts (see Fig. 2.19). Following Searle (1969), who conceptualised his propositional acts (referring and predicating) as inseparable things from his speech (illocutionary) acts, here, deictic expressions are treated as an integral part of the combined speech-deictic act. Let this special sort of act be implemented for natural language and graphics through a device called a *coordinated linguistic and graphical act* (COLGRA). COLGRAs will be similar to simple speech acts, except that they will accept deictic expressions as its referring expressions. For instance, there will be COLGRAs called *d-assert, d-ask,* and *d-specify* corresponding to the pure linguistic speech acts *assert, ask,* and *specify* respectively. To illustrate the mechanics of these acts, suppose that it is required to generate such an act to achieve the same effects as expression (4.17). Since, to generate that expression a *specify* speech act was used, here the corresponding *d-specify* COLGRA is used as follows:

$$\text{d-specify}(ghorizontal(\text{deictic}(l_5))) \tag{4.19}$$

which is realised as

$$\left\{ \begin{array}{c} highlight(l_5) \\ \text{“}this\ bar\text{”} \end{array} \right\} \text{“}must\ be\ horizontal\text{”} \tag{4.20}$$

and the combined visual effect of this COLGRA is shown in Fig. 4.2[4].

4.1.5 Negation

There are several syntactic devices available to produce sentences negating a proposition, P. One simple approach is that of juxtaposing the phrase *It is not*

[3]To define formally the function *deictic* consider the subset $G_e \subset G$ which contains only the geometrically interpretable symbols and expressions of type e, and the subset $L_T \subset L$ which contains the symbols and expressions of category T. Then, $deictic : G_e \to G \times L$ is defined so that $deictic(a) = \langle highlight(a), thisg(a)\rangle$, where $thisg : G_e \to L_T$ is a function that maps its argument into a term which consists of the demonstrative adjective “this” followed by the name of the geometrical type of its argument.

[4]Note in this Fig. 4.2, that, although the highlighted bar has been specified to be horizontal, it is not horizontal. It is precisely one of the goals of the design process to satisfy constraints like these.

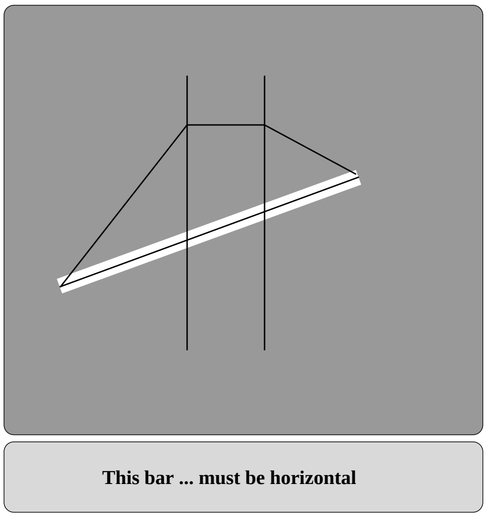


Figure 4.2: Realisation of d-specify$(ghorizontal(deictic(l_5)))$

the case that to the assertion of the proposition P, as it is shown in the following expressions:

$$\text{deny}(P) = [\textit{It is not the case that}]\ \text{assert}(P) \tag{4.21}$$
$$\text{d-deny}(P) = [\textit{It is not the case that}]\ \text{d-assert}(P) \tag{4.22}$$

Using this mechanism, the communicative act specified by d-deny$(ghorizontal(\text{deictic}(l_5)))$ will render the following display:

$$\text{``}\textit{It is not the case that}\text{''} \left\{ \begin{array}{c} highlight(l_5) \\ \text{``}\textit{this bar}\text{''} \end{array} \right\} \text{``}\textit{is horizontal}\text{''} \tag{4.23}$$

Another approach is to modify the verbal phrase to include the negation. For instance, for the proposition $p(a_2)(a_1)$ with two arguments the following formulae can be stated:

$$\text{negate}(p(a_2)(a_1)) = \text{t}[p(a_2)(a_1), \text{mdflist}(gnot)] \tag{4.24}$$
$$\text{d-negate}(p(a_2)(deictic(a_1))) = \text{t}[p(a_2)(deictic(a_1)), \text{mdflist}(gnot)] \tag{4.25}$$

where mdflist$(gnot)$ renders the sentence in its negative form. For instance "negate $(outside_*(c_1)(l_3))$" will result in the text "bar_3 *does not lie outside dielectric-area*$_1$".

Note that, prior to the NL generation phase, complex logical negation formulas are simplified using the following equivalences:

$$\neg(\forall x P) \equiv \exists x \neg P$$
$$\neg(\exists x P) \equiv \forall x \neg P$$
$$\neg(P \wedge Q) \equiv P \rightarrow \neg Q$$
$$\neg(P \vee Q) \equiv \neg P \wedge \neg Q$$

$$\neg(P \rightarrow Q) \equiv P \wedge \neg Q$$
$$\neg(P \leftrightarrow Q) \equiv \neg P \leftrightarrow Q$$

For instance, $\neg(\forall x[line(x) \rightarrow \exists y[circle(y) \wedge outside_*(y)(x)]])$ is translated into $\exists x[line(x) \wedge \forall y[circle(y) \rightarrow \neg outside_*(y)(x)]]$, which is used to produce the NL text:

some bars do not lie outside every dielectric-area[5]

4.2 Speech Acts and COLGRAs as Planning Operators

To be used in an explanation generation system, the speech acts and the COLGRAs are implemented as planning operators. These operators will constitute the *surface level* building blocks of composite explanations, whose structure is determined according to the planning operators that will be discussed below in Chapter 5. The structure of these explanations is developed according to some rhetorical relations inspired by those defined by Mann & Thompson (1988). Following the same design principles of these operators, the newly introduced basic operators in this work, designed to operationalise speech acts, rely on their preconditions fields to post new goals that will be satisfied by other operators whose effects fields match these goals. These operators are satisfied directly by speech acts or by other operators in this same category.

The speech acts and COLGRAs planning operators are defined according to the following fields:

Operator: contains the name of the operator and defines the form of its arguments.

Restrictions: the conditions stated in this field delimit the use of the operator, i.e., if one of these conditions is not satisfied, the operator will be ruled out.

Preconditions: the conditions established here have to be satisfied before the operator is applied.

Effects: this field states the results that are obtained by the application of the operator.

Nucleus: this field contains either a new goal to be satisfied, or a speech act expression that is directly rendered as text or a combination of text and a graphical action.

[5] The logical form provided by the given formula leads the NL generator to select grammar rule A.1 (Appendix A, p.200) to produce this text.

The *inform* type operators that end in *assert* speech acts are as follows:

Frame 4.1.

Operator: inform(S, H, P)
Restrictions:
<none>
Preconditions:
bel(S, not bel(H, P))
intend(S, bel(H, P))
Effects:
bel(H, P)
Nucleus:
informed(H, P)

Frame 4.2.

Operator: inform1(S, H, P)
Restrictions:
<none>
Preconditions:
bel(S, not informed(H, P))
intend(S, informed(H, P))
Effects:
informed(H, P)
Nucleus:
assert(S, H, P)

Frame 4.3.

Operator: informg1(S, H, p(a))
Restrictions:
graphical(a)
Preconditions:
bel(S, not informed(H, p(a)))
intend(S, informed(H, p(a)))
Effects:
informed(H, p(a))
Nucleus:
d-assert(S, H, p(deictic(a)))

Frame 4.4.

Operator: informg2(S, H, p(a_2)(a_1))
Restrictions:
graphical(a_1)
Preconditions:
bel(S, not informed(H, p(a_2)(a_1)))
intend(S, informed(H, p(a_2)(a_1)))
Effects:
informed(H, p)
Nucleus:
d-assert(S, H, p(a_2)(deictic(a_1)))

Frame 4.5.	*Frame 4.6.*
Operator: informg3(S, H, p$(a_2)(a_1)$) **Restrictions:** graphical(a_2) **Preconditions:** bel(S, not informed(H, p$(a_2)(a_1)$)) intend(S, informed(H, p$(a_2)(a_1)$)) **Effects:** informed(H, p) **Nucleus:** d-assert(S, H, p(deictic(a_2))(a_1))	**Operator:** informg4(S, H, p$(a_2)(a_1)$) **Restrictions:** graphical(a_1), graphical(a_2) **Preconditions:** bel(S, not informed(H, p$(a_2)(a_1)$)) intend(S, informed(H, p$(a_2)(a_1)$)) **Effects:** informed(H, p) **Nucleus:** d-assert(S, H, p(deictic(a_2))(deictic(a_1)))

To illustrate how the operators, given in *frames 4.1* to *4.6*, interact, suppose that the speaker (S) wants the hearer (H) to believe a certain proposition, P. This intention is translated into the goal "bel(H, P)". Since the **Effects** field of the operator "inform" (*frame 4.1*) matches this goal, it could be selected to satisfy it. If this is the case, the operator will post a new goal "informed(H, P)". This means that for the hearer to believe a certain proposition, he or she has to be informed of such a proposition. To satisfy this new goal the speaker could select among many other operators whose **Effects** field matches that goal. As a sample, the operators defined in *frames 4.2* to *4.6* could match the goal depending on the form of the proposition P.

Now, the operators that allow the description of rules are the following:

Frame 4.7.

Operator: state-rule(S, H, P)
Restrictions:
<none>
Preconditions:
bel(S, rule(P))
bel(S, not bel(H, rule(P)))
intend(S, bel(H, rule(P)))
Effects:
bel(H, rule(P))
Nucleus:
informed(H, rule(P))

Frame 4.8.

Operator:
state-rulel(S, H, P)
Restrictions:
<none>
Preconditions:
bel(S, not informed(H, rule(P)))
intend(S, informed(H, rule(P)))
Effects:
informed(H, rule(P))
Nucleus:
specify(S, H, P)

Frame 4.9.

Operator:
state-ruleg1(S, H, p(a))
Restrictions:
graphical(a)
Preconditions:
bel(S, not informed(H, rule(p(a))))
intend(S, informed(H, rule(p(a))))
Effects:
informed(H, rule(p(a)))
Nucleus:
d-specify(S, H, p(deictic(a)))

Frame 4.10.

Operator:
state-ruleg2(S, H, p(a_2)(a_1))
Restrictions:
graphical(a_1)
Preconditions:
bel(S, not informed(H, rule(p(a_2)(a_1))))
intend(S, informed(H, rule(p(a_2)(a_1))))
Effects:
informed(H, rule(p(a_2)(a_1)))
Nucleus:
d-specify(S, H, p(a_2)(deictic(a_1)))

Frame 4.11.

Operator:
state-ruleg3(S, H, p(a_2)(a_1))
Restrictions:
graphical(a_2)
Preconditions:
bel(S, not informed(H, rule(p(a_2)(a_1))))
intend(S, informed(H, rule(p(a_2)(a_1))))
Effects:
informed(H, rule(p(a_2)(a_1)))
Nucleus:
d-specify(S, H, p(deictic(a_2))(a_1))

Frame 4.12.

Operator:
state-ruleg4(S, H, p(a_2)(a_1))
Restrictions:
graphical(a_1)
graphical(a_2)
Preconditions:
bel(S, not informed(H, rule(p(a_2)(a_1))))
intend(S, informed(H, rule(p(a_2)(a_1))))
Effects:
informed(H, rule(p(a_2)(a_1)))
Nucleus:
d-specify(S, H, p(deictic(a_2))(deictic(a_1)))

A similar set of operators of a type *stated-rule* can easily be produced using the operators from *frame 4.7* to *frame 4.12* as models. To this aim, instead of “state” in the name of the operators write “stated”, e.g., “state-ruleg3” will produce “stated-ruleg3”. In the body of the operator, then, substitute the word “rule” by “was-rule”, and “specify” by “expect”. With this mechanism, for instance, taking the *state-ruleg4* operator in frame 4.12 as a model, the operator *stated-ruleg4* is produced as follows:

Frame 4.13.

Operator:
stated-ruleg4(S, H, p(a_2)(a_1))
Restrictions:
graphical(a_1)
graphical(a_2)
Preconditions:
bel(S, not informed(H, was-rule(p(a_2)(a_1))))
intend(S, informed(H, was-rule(p(a_2)(a_1))))
Effects:
informed(H, was-rule(p(a_2)(a_1)))
Nucleus:
d-expect(S, H, p(deictic(a_2))(deictic(a_1)))

"expect(S, H, P)" and "d-expect(S, H, P)" are speech acts similar to those speech acts "specify(S, H, P)" and "d-specify(S, H, P)" described above, except that the former ones are rendered using the auxiliary verb *should* instead of *must*.

Chapter 5

INTENTIONAL AND RHETORICAL STRUCTURES FOR MULTIMODAL EXPLANATIONS

In Chapter 1, a desirable multimodal dialogue was introduced. Chapter 2 placed systems for producing such dialogues in the context of natural language explanation generation systems and multimodal explanation generation systems. Chapter 3 was devoted to the definition of the logical language G, the natural language L, and the coordinated linguistic and graphical semantic units (COL-GSUs), to provide the semantic basis for generating multimodal explanations. This chapter develops an operationalisation of rhetorical structures to enable the generation of explanations, that utilise linguistic and graphical modalities, in design systems. To begin with, Section 5.1 presents an example dialogue to illustrate the kind of requirements necessary for generating explanations. Section 5.2 introduces the concept of rhetorical structures and, motivates the approach taken in this thesis for the generation of explanations. Section 5.3 shows how the rhetorical operators are used to generate explanations for describing a particular state of interpretation. Finally, Section 5.4 shows how explanations about transitions between states can be elaborated by interleaving communicative goals with problem-solving planning operators.

5.1 Requirements for the Generation of Explanations in Design Oriented Dialogues

Two kinds of explanation can be distinguished in design oriented dialogues, namely:

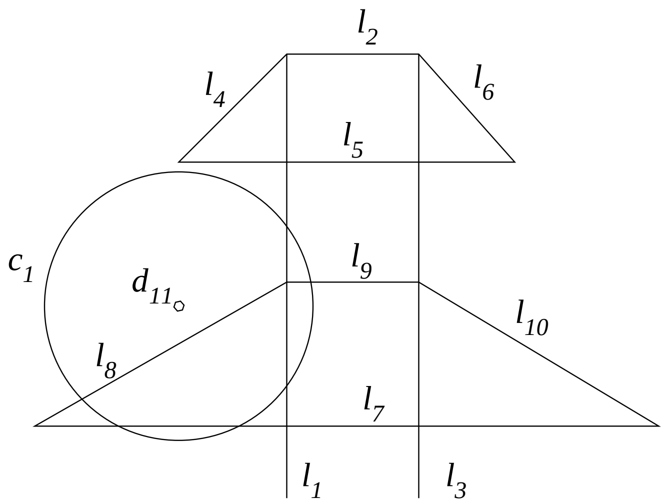


Figure 5.1: A simple configuration for a tower

1. Explanations about the current state of interpretation, in which the state of affairs of the state currently described in the knowledge base is explained. This sort of explanation will be called *state analysis explanations*.

2. Explanations about how a state was obtained from another state. This sort of explanation will be named *state synthesis explanations*.

To illustrate and develop the features of these explanations, consider the geometrical design situation depicted in Fig. 5.1. The corresponding coordinated linguistic and semantic unit (COLGSU) of this interpretation state is given by the *graphical theory* of expression (5.1), the *non-graphical theory* of expression (5.2), and the translation relation described in Section 3.2.3 (p.88).

$$
\begin{array}{l}
\underline{\text{Graphical Theory } (T_0)} \\
d_1 = the\_dot(-1.5)(0.0) \\
d_2 = the\_dot(-1.5)(12.0) \\
d_3 = the\_dot(1.5)(12.0) \\
d_4 = the\_dot(1.5)(0.0) \\
d_5 = the\_dot(-4.5)(10.0) \\
d_6 = the\_dot(4.5)(10.0) \\
d_7 = the\_dot(-7.5)(3.0) \\
d_8 = the\_dot(7.5)(3.0) \\
d_9 = the\_dot(-1.5)(6.0) \\
d_{10} = the\_dot(1.5)(6.0) \\
d_{11} = the\_dot(-4.5)(6.0) \\
l_1 = the\_line(d_1)(d_2) \\
l_2 = the\_line(d_2)(d_3)
\end{array}
\tag{5.1}
$$

$l_3 = the\_line(d_3)(d_4)$
$l_4 = the\_line(d_5)(d_2)$
$l_5 = the\_line(d_6)(d_5)$
$l_6 = the\_line(d_3)(d_6)$
$l_7 = the\_line(d_8)(d_7)$
$l_8 = the\_line(d_7)(d_9)$
$l_9 = the\_line(d_9)(d_{10})$
$l_{10} = the\_line(d_{10})(d_8)$
$c_1 = the\_circle(d_{11})(3.5)$
$p_1 = path(l_4)(path(l_2)(path(l_6)(l_5)))$
$p_2 = path(l_7)(path(l_8)(path(l_9)(l_{10})))$

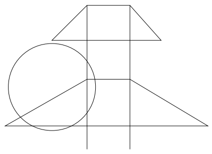
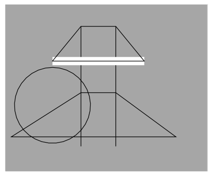
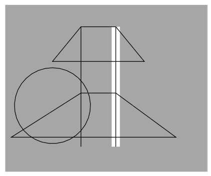
Non-graphical Theory (T_G) (5.2)
$painted\_with_*(anticorrosive\_paint)(l_1)$
$painted\_with_*(luminous\_paint)(l_2)$
$painted\_with_*(anticorrosive\_paint)(l_3)$
$painted\_with_*(luminous\_paint)(l_4)$
$painted\_with_*(luminous\_paint)(l_5)$
$painted\_with_*(luminous\_paint)(l_6)$
$painted\_with_*(luminous\_paint)(l_7)$
$painted\_with_*(luminous\_paint)(l_8)$
$painted\_with_*(luminous\_paint)(l_9)$
$painted\_with_*(luminous\_paint)(l_{10})$

In design environments, an explanation of the current state of interpretation in terms of the stated design constraints allows the designer to evaluate the status of his or her design goals. Design constraints can be modelled in the languages L and G developed in Chapter 3 as propositions (see the design rule (3.6) and its corresponding translation to a logical formula (3.18) in Chapter 3, p.100). These propositions will be grouped in a set called *a constraints set*. When tested against the world model, represented by the current COLGSU, some of these constraints will evaluate to **true** and some of them to **false**. The constraints that evaluated to **true** will be called *satisfied* constraints, while those evaluated to **false** will be called *unsatisfied* constraints.

Let the example being considered here be subject to the *design constraints* given in the following set:

$$\text{constraints-set} = \{\forall x[line(x) \rightarrow \forall y[circle(y) \rightarrow outside_*(y)(x)]], \\ ghorizontal(l_5), painted\_with(anticorrosive\_paint)(l_3)\} \quad (5.3)$$

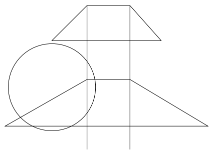
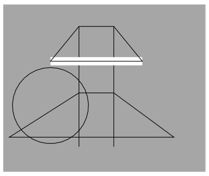
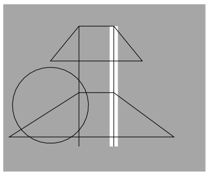
A possible multimodal dialogue between a designer and a system, in the context of the proposed example, could proceed as follows.

TEXT	GRAPHICS
[5.1]: Designer: Describe design-constraints[a]. **[5.2]: System:** The following are the design constraints: **[5.3]: System:** Every bar must lie outside every dielectric area [a] The current version of the prototype system only accepts inputs in a special command language. The expressions of this command language are translated into communicative goals which are used to trigger the explanation planning process.	
[5.4]: System: (• *Highlighting* l_5) This bar must be horizontal	
[5.5]: System: (• *Highlighting* l_3) This bar must be painted with anticorrosive-paint	

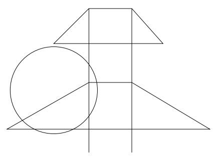
At the start of this dialogue, the user (a designer) asks for a description of the design constraints. The required information, in this case, consists of presenting the current *active* constraints. To satisfy this requirement, the system has to build up a discourse consisting of an *introduction* (the utterance in **[5.2]**) followed by a series of three sentences (as shown in **[5.3]**, **[5.4]**, and **[5.5]**): one corresponding to each of the design constraints given in expression (5.3). According to Mann & Thompson's rhetorical structure theory (RST)[1], the *introduction* and the series of sentences stand in a rhetorical relation, such that the *introduction* provides a background to the material exposed by the series of sentences. The rhetorical relations defined by Mann & Thompson (1988) categorise the two or more involved spans of text as the nucleus, which contains the essential material of the discourse, and the satellite or satellites, which contain the supporting material of the discourse. Since, what is essential to the explanation, in this case, is the series of sentences explaining the design constraints, such a series plays the role of the *nucleus*, while the *introduction* plays the role of a *satellite* in the RST relation. Furthermore, the series of sentences embodies by itself a rhetorical relation among its components. This relation is multinuclear, since all its components have the same *importance*, and it is similar to Mann & Thompson's *sequence* relation, though here, no succession relation is required among the components. When examined more closely, it is noticeable that every constraint is described by a similar syntactic device, namely, a sentence construed with the auxiliary verb *must*. Here, it can be said that the system has issued a speech act called *specify* a rule or a constraint[2]. It should be noted too that inside the sentences, some of the objects are introduced into the discourse by explicit deictic references. In sentence **[5.4]**, for instance, a particular line of the graphical depiction is highlighted while the system utters the phrase *"this bar"*, enabling the designer to identify the graphical representation of the object referred to by that phrase.

[1] See the article by Mann & Thompson (1988), and the introductory material to this theory presented in section 2.4 (p. 31).

[2] It can be argued that it is not the system which specifies the design constraints that must be satisfied by a user, but the other way around. In a design cycle, however, a frequent situation is that one designer specifies the design constraints/rules and another designer uses them to design some artifact. Common examples of this are the design manuals or handbooks, and the design companies procedures, which, in general, are written by other designers than those who are achieving the particular design tasks. When the system performs a *specify* speech act, it is playing the role of a design manual in the sense that it is communicating to one designer what another designer specified.

TEXT	GRAPHICS
[5.6]: Designer: Describe satisfied-design-constraints.	
[5.7]: System: The following are the satisfied constraints: **[5.8]: System:** (• *Highlighting* l_5) This bar is horizontal	
[5.9]: System: (• *Highlighting* l_3) This bar is painted with anticorrosive-paint	

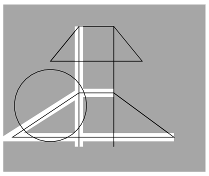
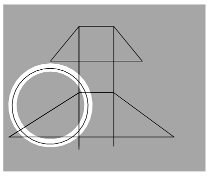
Continuing with the proposed example, the designer now asks for those design constraints that are satisfied in the current COLGSU. To respond to the user, the system constructs a discourse in a very similar fashion to the previous one, i.e., an introductory background followed by a series of sentences, each of them corresponding to each satisfied constraint (from [5.7] to [5.9]). In this case, however, the surface form of each sentence is the simple present tense, i.e., the speech act the system performs here is a simple assertion of a fact.

TEXT	GRAPHICS
[5.10]: Designer: Describe unsatisfied-design-constraints. **[5.11]: System:** The following are the unsatisfied constraints: **[5.12]: System:** Every bar should lie outside every dielectric-area, but some bars do not lie outside some dielectric-areas.	

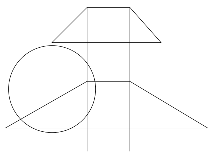
Now the designer asks for the unsatisfied design constraints. The discourse elaboration, in this case, starts with the same pattern: an introductory background **[5.11]** followed by the central topic of the discourse **[5.12]**. Here, however, to explain a single unsatisfied rule, instead of a simple sentence, a compound of two sentences is used. In terms of the RST, this compound consists of two nuclei, the two sentences separated by the conjunction *but*, which stand in a *contrast* relation: the fact that the current configuration does not satisfy the design constraint is contrasted with the contents of the constraint itself. To produce the surface text of the involved sentences, two new speech acts are performed: *expect*, which is introduced by the modal verb should, and *negate*, which is exhibited by the simple negative present tense form of the sentence. In this particular case, the only unsatisfied constraint is a quantified expression, namely: $\forall x[line(x) \rightarrow \forall y[circle(y) \rightarrow outside_*(y)(x)]]$, whose negation is given by $\neg(\forall x[line(x) \rightarrow \forall y[circle(y) \rightarrow outside_*(y)(x)]]) = \exists x[line(x) \wedge \exists y[circle(y) \wedge \neg outside_*(y)(x)]]$, which is the expression used to produce the sentence in the negative form.

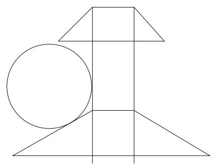
TEXT	GRAPHICS
[5.13]: Designer: Detail quantified-expression.	

Since the quantified expression is not specific about the objects referred to by the expressions *"some bars"* and *"some dielectric-areas"*, the designer asks the system for detailed information about these objects.

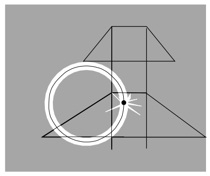
TEXT	GRAPHICS
[5.14]: System: (• *Highlighting a set of lines*) These bars do not lie outside ...	
[5.15]: System: ... (• *Highlighting the circle*) this dielectric-area.	

During the evaluation of quantified expressions, the information about the participating objects is recorded. This allows the system to elaborate on the previously quantified expression, giving more detailed information. In this example, the set of objects (i.e., their graphical representation as lines) referred to by the plural indefinite expression *"some bars"* of utterance **[5.12]**, is associated deictically to the plural definite noun phrase *"these bars"* of utterance **[5.14]**.

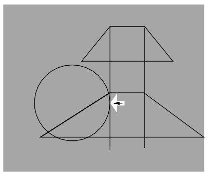
TEXT	GRAPHICS
[5.16]: Designer: Solve-problem.	

[5.17]: System: (• *Applies a problem-solving algorithm and shows the final solution*) **[5.18]: Designer:** Explain problem-solving procedure.	

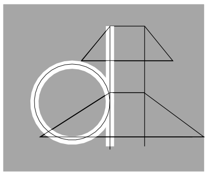
Now, the designer calls upon the system to solve the problem, i.e., to reach a state in which all design constraints are satisfied **[5.16]**. With this aim, the system applies a problem-solving strategy and presents the final result in a graphical fashion **[5.17]**. Then, the designer asks for a detailed explanation about the problem-solving procedure carried out **[5.18]**. Since a sequence of planning operators were applied to solve the problem, to produce the discourse corresponding to this explanation, the system uses, as the upper level structure, a *sequence* rhetorical relation, which is a multinuclear RST relation. Two high level planning operators were used to solve the problem in this case. Therefore, two nuclei constitute the *sequence* rhetorical relation. The first nucleus, is shown below from **[5.19]** to **[5.21]**, and the second, from **[5.22]** to **[5.24]**.

TEXT	GRAPHICS
[5.19]: System: (• *Highlighting the circle and the starting point of the displacement*) I moved this dielectric-area from here ...	

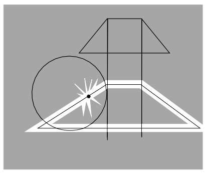
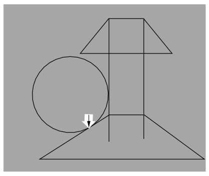
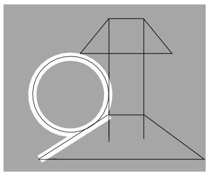
[5.20]: System: (• *Highlighting the ending point of the displacement and showing the new postion of the circle*) ... to here, so that ...



[5.21]: System: (• *Highlighting line l_1 and the circle*) ... this bar lies outside this dielectric-area.



The explanation of each individual step in the problem-solving process, is carried out by a discourse whose constituting elements stand in a *solutionhood* RST relation. In this relation, the nucleus presents the solution to a problem, and the satellite shows that the problem has been solved. In the example, the nucleus, i.e., the solution to the problem, is presented first from **[5.19]** to **[5.20]**, and then the solved problem itself is introduced and presented from **[5.20]** to **[5.21]**. To build up this explanation, the system has to keep track of the sequence of states produced during the problem-solving process. The system actions that refer back to a previous state are explained using a sentence in the simple past tense **[5.19]**. The constraint satisfied through this particular step is explained using an assertion in simple present tense **[5.21]**. The solution to the other constraint is explained in a similar fashion, resulting in the dialogue shown below.

TEXT	GRAPHICS
[5.22]: System: (• *Highlighting polyline p_2 and the starting point of the displacement*) Then, I moved this crossarm from here ...	
[5.23]: System: (• *Highlighting the ending point of the displacement and showing the new position of the polyline*) ... to here, so that ...	
[5.24]: System: (• *Highlighting line l_8 and the circle*) ... this bar lies outside this dielectric-area.	

To consider the requirements for generating the sort of responses illustrated by this example, first let us look at some of the characteristics of the dialogue in the above example. From the functional point of view, the two types of explanation that occur in design environments, are exhibited in this example, namely: state analysis explanations, which are illustrated from dialogue portion **[5.2]** to dialogue portion **[5.15]**, and state synthesis explanations, which are illustrated from dialogue portion **[5.19]** to dialogue portion **[5.24]**. From the syntactic and structural point of view, several levels can be distinguished:

1. **Pure textual sentences**. For instance, dialogue portion **[5.3]** is one such simple sentence.
2. **Combinations of a graphical depiction with a sentence**. For instance, dialogue portion **[5.4]** shows a sentence that explicitly refers to a graphical depiction.
3. **Structures of two or more sentences**. For instance, the text generated by the system in dialogue portion **[5.12]** shows such a structure.
4. **Structures of two or more combinations of graphical depictions with sentences**. For instance, the displays produced by the system in dialogue portions **[5.19]**, **[5.20]**, and **[5.21]** show a structure of this kind.

From the communicative standpoint, it can be said that the syntactic devices as well as the structures exhibited are the product of intentional goals of the *system* to change the mental state of the designer. Put another way, at the sentence level, there are syntactic devices that convey the *system* intentions, and at the *discourse* level, i.e., the level where two or more sentences are involved, there are, additionally, rhetorical relations that help to convey these intentions.

From the multimodality standpoint, in many cases, the intentional goals are supported by the inclusion of explicit deictic references. See, for instance, how the deictic association of a set of lines in dialogue portion **[5.14]** helps the designer to identify the bars referred to by the expression "these bars".

In view of this, to produce the sort of explanation shown here, it is required to have an account of :

a) the generation of sentences, with deictic references included, from intentional goals, and

b) the generation of discourses from rhetorical relations that help to reach some intentional goals.

Thus, the aim of this chapter is to present the design foundations of a planning system capable of generating the sort of responses in the example shown above, using, for the generation of the sentence level, the operationalisation of speech acts and, for the discourse level, an operationalisation of the rhetorical structure theory. The operationalisation of speech acts was presented in Chapter 4, and Section 5.2, which follows below, presents the approach taken in this thesis to the operationalisation of rhetorical structure theory, and uses as an example the dialogue, from displays **[5.19]** to **[5.24]**. Section 5.3 develops the main operators for generating state analysis explanations based on the kind of structure presented in Section 5.2. Section 5.4 explains how state synthesis explanations, of the kind exemplified in Section 5.2, are obtained during the problem solving process, by interleaving communicative goals with the planning operators devoted to problem solving.

5.2 Intentional Goals and Rhetorical Structures

There are several approaches for building up a rhetorical structure and presenting explanations based on that structure. Chapter 2 summarised two main approaches that have been used successfully for realistic prototype systems, namely: the use of schemata, which are augmented transition networks (ATN) designed to produce structured discourses [McKeown 1982, Feiner & McKeown 1991, Lester & Porter 1997], and the use of a planning technique whose operators embody the structure of the discourse [Moore 1995, Maybury 1993*a*, Wahlster et al. 1993]. The kind of generation task addressed in this thesis can potentially utilise either of these approaches as a basis.

To select the approach actually used, the following research strategy and considerations were made:

1. The approach adopted has to be as generic as possible so as not to exclude future directions of research.

2. The approach has to be transparent and modular so that it is easier develop and evolve.

3. The task being solved is dynamic and requires a scheme for dynamically generating explanations.

4. The approach should enable deictic references to be easily incorporated.

Taking these considerations into account, both approaches were viable when considering criteria 3 and 4. That is, it looked possible to adapt both approaches so that they would work in a dynamic fashion and deictic references could be introduced. However, at the time of deciding, based on what was published on McKeown's work, the plan based approach appeared to be more promising when considering the first two criteria. This view was based on the following observation of McKeown's work. Schemata in McKeown's work are used to select the content of an explanation and to organise it according to a definite discourse purpose. Although in her PhD thesis [McKeown 1982], she enlists several schemata which were derived from analysed texts, in the COMET system, she and her colleagues implemented only two. Furthermore, they mentioned [McKeown et al. 1990] that the development of such schemata is a *time-intensive* process. This indicates that schemata are very much dependent on the target task for which they are intended to generate explanations, and that modifying or adapting them is not an easy job since it requires dealing not only with the structure to be imposed on the discourse, but also with details of *how* this structure is being imposed on the discourse. From the standpoint of the quality of the discourse produced, schemata are like precompiled plans, while the planning approach generates them *on the fly*, mainly due to the presence of an intentional structure. Once they are generated, the difference between the resulting plans is trivial, and they can be regarded as alternative data structures. Furthermore, in the planning approach, the problem-solving is interleaved with the generation of the rhetorical structure utilised to express the answer. This gives this approach more flexibility from the implementation point of view, since, when writing new problem-solving plan operators, the implementer can focus on what *atomic*[3] communicative goals to include in the operators without having to worry about *how* the rhetorical structure is going to be produced or what information is going to be *consumed*[4].

On the other hand, although it was appreciated that the ATN based approach could be more efficient in terms of speed of execution, the research strategy adopted was to trade off this potential risk in favour of the first two criteria[5].

Having decided to adopt a plan based approach to generate explanations, following a number of authors [Hovy 1991, Moore 1995, Wahlster et al. 1993], Mann and Thompson's rhetorical structure theory (RST) was operationalised. The adoption of Mann and Thompson's RST was based on the view expressed by a number of authors [Austin 1962, Searle 1969, Grosz & Sidner 1986, Allen 1983, Mann & Thompson 1988] that the relationship between the rhetorical structure and the intentions needs to be represented explicitly. Again, following the research strategy expressed in criterion 1, this theory was operationalised as fully as possible, eventhough not all the information represented by structures of the theory were utilised in this thesis (c.f. the section on future work which describes how the unused information could be utilised more fully).

The following presents the operationalisation of RST by means of plan operators, the summary section will contrast this operationalisation with that proposed by other authors.

The operationalisation developed in this thesis adopts plan operators that have the same fields as those described in Section 4.2 (p. 123), but with the following modifications and additions:

Effects: This field states the expected effects in the hearer's mental state. The effects stated here will be achieved by the combination of the achievement of the goals stated in the **Nucleus** and **Satellite** fields described below.

Nucleus: This field contains the goals that have to be reached to present the hearer with the central material of the explanation.

---

[3]As opposed to the general discourse purpose.

[4]Once an ATN, implementing a schema, is launched it *waits* for *specific* items of the relevant knowledge pool to be consumed in order to build up a desired explanation (as explained in Section 2.3 p.27).

[5]With hindsight, lack of efficiency did not prove to be a problem in the limited prototype implemented in BinProlog on a PC running Linux.

Satellite: This field contains the goals that have to be reached to present the hearer with the supporting material of the explanation.

In this operationalisation, an operator can have zero or more satellites, and some operators embody multinuclear RST relations, e.g., the *sequence* and the *contrast* rhetorical relations. Furthermore, the order in which the satellite's or satellites' material is displayed in relation to the nucleus, is determined according to the order in which it or they appear in the operator. Mann and Thompson's RST schemata do not constrain the order of the nucleus or the satellites in the text span to which the the schema is applied [Mann & Thompson 1988], since that theory was developed in the context of text analysis. Here, however, the system plays the role of a writer, and as Mann and Thompson themselves said: *"Ordering seems to be under the independent control of the writer."*

As an example, consider the planning operator for the *sequence* rhetorical relation which is presented below in *frame 5.1*:

Frame 5.1.

Operator: sequence(S, H, $\mathcal{P}$)
Restrictions:
is-a-set($\mathcal{P}$)
cardinality($\mathcal{P}$) ≥ 2
[P_1 is first($\mathcal{P}$), $\mathcal{P}'$ is rest($\mathcal{P}$), P_2 is first($\mathcal{P}'$)]
Preconditions:
bel(S, hold(true, $\mathcal{P}$))
bel(S, ordered-set($\mathcal{P}$))
intend(S, bel(H, ordered-set($\mathcal{P}$)))
Effects:
bel(H, ordered-set($\mathcal{P}$))
Nucleus:
bel(H, P_1)
bel(H, succeeds(P_2,P_1))
bel(H, ordered-set($\mathcal{P}'$))
Satellite:
<none>

This operator has two actors: a speaker or presenter *(S)*, and a hearer or receiver *(H)*, and it is designed to present the propositions contained in the set $\mathcal{P}$ as a sequence. Two restrictions are imposed for the application of this operator: that the argument $\mathcal{P}$ is a set, and that this set has at least two members. The statements that appear enclosed by brackets ([ ]) in the **Restrictions** field are not properly restrictions, but rather procedural declarations that allow simpler and clearer code within the operator. For instance, these declarations allow the second statement in the **Nucleus** field to be written as "bel(H, succeeds(P_2, P_1))", instead of the more verbose "bel(H, succeeds(first(rest($\mathcal{P}$)),first($\mathcal{P}$)))". The functions "first" and "rest" give the first element of a given set, and the set consisting of all the elements of the argument except the first, respectively. "bel(A, P)" is a proposition about the mental state of the agent A, that states that the agent believes proposition P. "hold(π, $\mathcal{P}$)" is a proposition that states that the predicate π is true for every member of $\mathcal{P}$, i.e., hold(π, $\mathcal{P}$)$\leftrightarrow$ ($\forall P \in \mathcal{P}\ \pi(P)$). Since, true($P$) $\leftrightarrow P$, it follows that hold(true, $\mathcal{P}$)$\leftrightarrow$ ($\forall P \in \mathcal{P}\ P$). "intend($A_1$, bel($A_2$, P))" is a proposition that states the intention of agent A_1 to change the mental state of agent A_2, so that he or she believes proposition P. The proposition "ordered-set($\mathcal{P}$)" is true if $\mathcal{P}$ is a *well-ordered set*[6]. "succeeds(P_2,P_1)" is true if P_2 succeeds P_1, i.e., $P_1$is the first element in the subset $\{P_1, P_2\} \subseteq \mathcal{P}$, in the order established in $\mathcal{P}$. According to Mann & Thompson (1988), *a succession relationship between the situations is presented in the nuclei,* and the effect is that *the reader* (hearer) *recognizes the succession relationships among the nuclei.* These conditions are specified in the **Nucleus** and **Effects** fields of this operator respectively. The **Effects** field establishes that, after the application of this operator, a state in which the hearer believes that $\mathcal{P}$ is an ordered-set, will be reached. In this example, the nucleus contains three communicative goals expressed in terms of the mental state of the hearer. The propositions expressing these goals embody the system's intentions to reach a state in which these propositions hold. To satisfy the nucleus goals, other operators whose **Effects** fields match the goals should be applied. Suppose, for now, that the first goal in the **Nucleus** field has been reached. Notice that, to satisfy the last goal, a recursive application of this same operator can be performed, unless, of course, the cardinality of the argument $\mathcal{P}'$ is less than 2. To cope with a unit set, the *trivial-seq* operator, shown in *frame 5.2*, is introduced. Finally, to make the link between two consecutive *sequence* operator applications, each bringing in one proposition (say, P_1, and P_2), the goal "bel(H, succeeds(P_2, P_1))" is stated. This goal can be satisfied by the *basic-seq* operator shown below in *frame 5.3*.

[6]An *ordered set* is a set S together with binary relation, usually written $<$, on S to S which is transitive and satisfies the trichotomy law (for s and t in S, exactly one of $s < t$, $s = t$, $t < s$ holds). An ordered set is *well-ordered* if *every* non-void subset of S has a first element [MacLane & Birkhoff 1967, p. 42]

Frame 5.2.

Operator:
trivial-seq(S, H, $\mathcal{P}$)
Restrictions:
is-a-set($\mathcal{P}$)
cardinality($\mathcal{P}$) = 1
[P is first($\mathcal{P}$)]
Preconditions:
bel(S, ordered-set($\mathcal{P}$))
intend(S, bel(H,
ordered-set($\mathcal{P}$)))
Effects:
bel(H, ordered-set($\mathcal{P}$))
Nucleus:
bel(H, P)
Satellite:
<none>

Frame 5.3.

Operator:
basic-seq(S, H, P_1, P_2)
Restrictions:
<none>
Preconditions:
bel(S, P_1)
bel(S, P_2)
bel(S, succeeds(P_2, P_1))
intend(S, bel(H,
succeeds(P_2, P_1)))
Effects:
bel(H, succeeds(P_2, P_1))
Nucleus:
bel(H, P_1)
cue({"Then", "Next"})
bel(H, P_2)
Satellite:
<none>

Note that when applying the *basic-seq* operator (*frame 5.3*), in the context of the application of the *sequence* operator (*frame 5.1*), the goal "bel(H, P_1)" will already be satisfied, requiring no further satisfaction within this operator. The proposition "succeeds(P_2, P_1)" states that proposition P_2 succeeds proposition P_1 in an established order. Finally, "cue({W_1, W_2, ..., W_n})" invokes the simple speech act *cue-phrase* which writes one of the given phrases in the current textual output.

5.2.1 Example: generating state synthesis explanations

An application of the *sequence* operator was shown in the example dialogue from displays **[5.19]** to **[5.24]**. This sequence, that presents two steps of a problem-solving process, and that involves the synthesis of two states, is the result of satisfying the goal:

$$\begin{array}{l}\text{bel}(H, \text{ordered-set}(\{\\ \quad \text{solved}(outside_*(c_1)(l_1), 1,\\ \qquad past\text{-}move(system, c_1)(from(v_1), to(v_2)), 2),\\ \quad \text{solved}(outside_*(c_1)(l_8), 2,\\ \qquad\qquad past\text{-}move(system, p_2)(from(v_3), to(v_4)), 3)\})\end{array} \tag{5.4}$$

When the *sequence* operator is applied with this information, it posts two new goals:

$$\text{bel}(H, \text{solved}(outside_*(c_1)(l_1), 1,\ past\text{-}move(system, c_1)(from(v_1), to(v_2)), 2)) \quad (5.5)$$

$$\text{bel}(H, \text{solved}(outside_*(c_1)(l_8), 2,\ past\text{-}move(system, p_2)(from(v_3), to(v_4)), 3)) \quad (5.6)$$

Now, consider how the goal in expression (5.5) is satisfied producing the text and graphics from dialogue portions **[5.19]** to **[5.21]**. A proposition in the form "bel(H, solved(P, S_1, M, S_2))" means that the agent H believes that the problem expressed by the fact that proposition P is not satisfied in state S_1, is solved by the method M producing state S_2. States are identified by integer numbers in sequence. Mann and Thompson's *solutionhood* schema embodies a rhetorical relation that satisfies a requirement like this: the nucleus presents the solution of a problem which is presented in a satellite. By means of this presentation, the hearer recognises the material presented in the nucleus as a solution to that presented in the satellite. This schema is operationalised here by a new operator as shown in *frame 5.4*.

Frame 5.4.

Operator:
  solutionhood(S, H, P, S_1, M, S_2)
Restrictions:
  state(S_1)
  state(S_2)
  M = A(F, T)
  [current(S_2)]
Preconditions:
  bel(S, unsatisfied(P, S_1))
  bel(S, solved(P, S_1, M, S_2))
  intend(S, bel(H, solved(P, S_1, M, S_2)))
Effects:
  bel(H, solved(P, S_1, M, S_2))
Satellite: (* optional *)
  bel(H, unsatisfied(P, S_1))
Nucleus:
  bel(H, action(S_1,M, S_2))
Satellite:
  cue({"so that now", "consequently"})
  bel(H, satisfied(P, S_2))

Consider, for now, that the goals in the satellites are satisfied[7]. Just note that state S_2 has been signaled as the current state: this means that sentences based on this state will be rendered in the present tense while those based on S_1 will be uttered in the past tense. Now, consider the goal stated in the **Nucleus** field of the operator: "bel(*H*, action(S_1,M, S_2))". This goal states that the agent H believes that M is an action in state S_1 which will produce as a result the new state S_2. To satisfy a goal like this, the following operator is introduced:

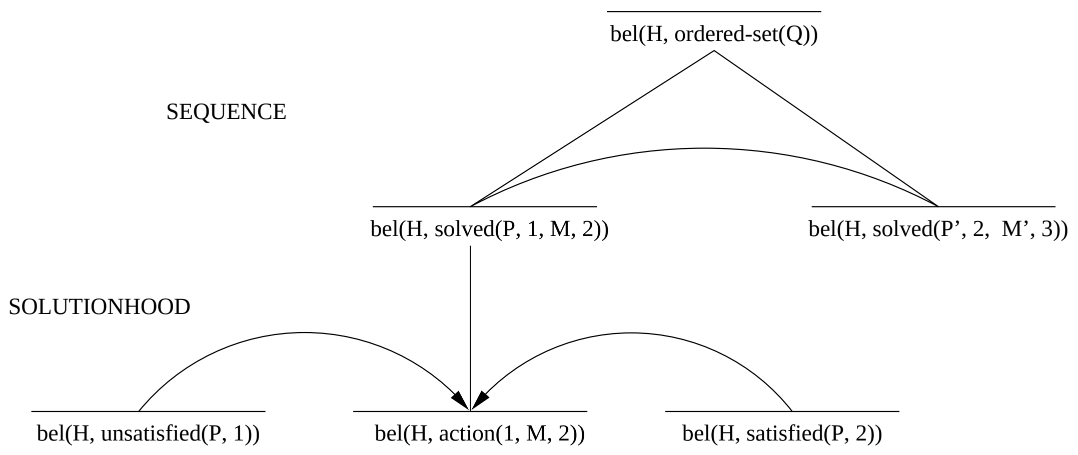
Frame 5.5.

Operator:
 report-action(*S, H,* S_1, M, S_2)
Restrictions:
 state(S_1)
 state(S_2)
 M = A(F, T)
Preconditions:
 bel(*S,* action(S_1, M, S_2))
 intend(*S,* bel(*H,* action(S_1, M, S_2)))
Effects:
 bel(*H,* action(S_1, M, S_2))
Nucleus:
 display(S_1)
 bel(*H,* A)
 inform-from(*S, H,* F)
 display(S_2)
 inform-to(*S, H,* T)
Satellite:
 <none>

In the case of the example, A = *past-move*(*system*,c_1), F = *from*(v_1), T = *to*(v_2). The communicative act "display(S)", draws the graphical representation corresponding to the given state S. In the nucleus, "inform-from" and "inform-to" are basic speech acts that produce the utterances "from here" and "to here" respectively, while the corresponding locations v_1 and v_2 are highlighted in the graphical representation, as shown in dialogue portions **[5.19]** and **[5.20]**.

The simple goal "bel(*H,* A)" = "bel(*H*, *past-move*(*system*, c_1))" will call for its satisfaction several operators to get down to a speech act that will produce:

[7] Goals of the type "bel(*H*, unsatisfied(P, S))", and "bel(*H*, satisfied(P, S))" serve the generation of explanations about the status of design constraints, and they will be studied later in sections 5.3.2 and 5.3.3.



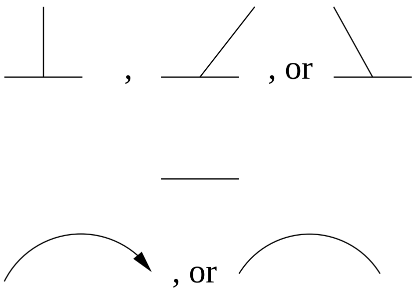
Where: P = outside$_*(c_1)(l_1)$
P' = outside$_*(c_1)(l_8)$
M = past-move(system, c_1)(from(v_1), to (v_2))
M' = past-move(system, p_2)(from(v_3), to (v_4))
Q = {solved(P, 1, M, 2), solved(P', 2, M', 3)}

Figure 5.2: Partial view of rhetorical structure to explain a problem-solving process

$$\text{“}I\ moved\text{”} \left\{ \begin{array}{c} highlight(c_1) \\ \text{“}this\ dielectric\text{-}area\text{”} \end{array} \right\} \tag{5.7}$$

which indicates that the system generates the sentence *"I moved this dielectric-area"* while the circle c_1 is highlighted.

A partial view of the rhetorical structure built up by the posting of these goals is shown in Fig. 5.2. In this figure,



stand for nuclei

stand for a satellite

stand for relations between spans

5.3 State Analysis Explanations

Rhetorical structures similar to those presented in the previous section can also be used to produce explanations about a particular state of interpretation. Many things could be asked and then explained about a particular configuration represented by a state of analysis like the one depicted in Fig. 5.1. From the design point of view, however, the most interesting explanations are about the design constraints. Three things the designer usually wants to know about the design constraints are:

1. the *active* constraints operating in a particular design, i.e., the elements of the *constraints set*,

2. the set of constraints currently satisfied, i.e., those that are currently **true**, and

3. the set of constraints yet to be satisfied, i.e., those that are currently **false**.

In order for a system to provide this information, three different types of explanation should be available: (i) description of the design constraints, (ii) explanation of the status of satisfied constraints, and (iii) explanation of the status of unsatisfied constraints.

The following subsections describe how each of these three types of explanation are achieved.

5.3.1 Describing design constraints

Let $\mathcal{P}= \{P_1, P_2, \ldots, P_n\}$ be a set of propositions representing the design constraints. Since this set groups an indefinite number of elements of the same category, it is straightforward to think about them as a group of propositions with a common property, in this case that all of them are rules, and explain them as such. For this purpose, the *description-grp* operator is introduced. This operator is similar to the *sequence* operator in that it deals with a group of items. In this case, however, instead of a succession relation among its members, a common property that all of them share is established by the operator. The operator is defined as follows:

Frame 5.6.

Operator:
  description-grp(S, H, π, $\mathcal{P}$)
Restrictions:
  is-a-set($\mathcal{P}$)
Preconditions:
  bel(S, hold(π, $\mathcal{P}$))
  intend(S, bel(H, hold(π, $\mathcal{P}$)))
Effects:
  bel(H, hold(π, $\mathcal{P}$))
Nucleus:
  FORALL P $\in \mathcal{P}$
    bel(H, π(P))
Satellite:
  <none>

To provide some background to the series of explanations generated by the *description-grp* operator, another operator is introduced. This new operator, called *background-of-grp,* is based on Mann & Thompson's (1988) background rhetorical relation, though, in this work it is specifically directed to provide background information for the series of descriptions of items with a common property.

In general, a background rhetorical relation is required to increase the ability of the addressee to comprehend an element in the *nucleus* part [Mann & Thompson 1988]. Thus, the effect of this operator must be so that the hearer *(H)* will be competent to comprehend the material presented in the nucleus. This requirement is specified by a proposition of the form "competent(H, comprehend(H, <*something-in-the-nucleus*>))".

Given this proposition, the rhetorical structure for explaining a set of constraints can be defined as shown in *frame 5.7*.

Frame 5.7.

Operator:
background-of-grp(S, H, π, $\mathcal{P}$)
Restrictions:
is-a-set($\mathcal{P}$)
Preconditions:
bel(S, hold(π, $\mathcal{P}$))
intend(S, competent(H,
comprehend(H, hold(π, $\mathcal{P}$))))
Effects:
competent(H, comprehend(H, hold(π, $\mathcal{P}$)))
Satellite:
introducePl(S, H, π)
Nucleus:
bel(H, hold(π, $\mathcal{P}$))

In this operator, introducePl(S, H, π) is a speech act that introduces the series of items in terms of the predicate π, which is true of each item. For instance, to introduce a sequence of propositions: $P_1, P_2, ..., P_n$, such that each is a design constraint, and then the proposition rule(P_i) is true for $i = 1, 2, ..., n$, introducePl(S, H, rule) will produce the text:

> "The following are the design constraints[8]:"

Using the two operators described here along with the basic speech acts operators presented in section 4.2, explanations to describe the design rules one by one can be easily produced.

For instance, supposing that the speaker (S) is the system (sys), the hearer (H) is the designer, and the set that contains the design rules is "constraints-set", the initial goal to be satisfied could be:

$$\text{competent(designer,} \quad \text{comprehend(designer, hold(rule, constraints-set)))} \tag{5.8}$$

with the example constraints-set as given in expression (5.3), and the configuration given in Fig. 5.1, when the goal in expression (5.8) is posted, the operator background-of-grp is instantiated as follows:

[8] The lexicon provided in Appendix A (see Table A.2), renders the term *rule* as the text "design constraints".

Frame 5.8. Instantiated operator

Operator:
background-of-grp(sys, designer, rule, constraints-set)
Restrictions:
is-a-set(constraints-set)
Preconditions:
bel(sys, hold(rule, constraints-set))
intend(sys, competent(designer,
comprehend(designer, hold(rule, constraints-set))))
Effects:
competent(designer,
comprehend(designer, hold(rule, constraints-set)))
Satellite:
introducePl(sys, designer, rule)
Nucleus:
bel(designer, hold(rule, constraints-set))

This operator posts two goals, the first one is the satellite of the rhetorical relation:

introducePl(sys, designer, rule) (5.9)

which is a speech act that directly produces the text:

"The following are the design constraints:"

This corresponds to the utterance **[5.2]** of the example dialogue (p. 132). The other goal, which will develop the nucleus of the rhetorical relation, is:

bel(designer, hold(rule, constraints-set)) (5.10)

There are several operators that can be used to satisfy this goal. For instance, a speech act *inform* operator instantiated as "inform(sys, designer, hold(rule, constraints-set))"[9] could, in theory, be selected, since its **Effects** field matches the goal.
However, to prevent this, the *more specific* heuristic[10] is used to decide among several operators that match a given goal. This heuristic captures the fact that

[9]See definition of this operator in *frame 4.1* p. 124.

[10]That an operator is *more specific* than another with respect to a goal, means that when the effects field of the operator is unified with the goal, the number of successful matches of terms is higher than that of the other operator. For example, consider the goal "informed(designer, rule(a(X)))", and suppose that there are three operators whose effects fields are:
operator 1: "informed(A, P)",
operator 2: "informed(A, rule(a(Q)))", and

one operator can be defined to satisfy a goal by means of strategies that are more specific to the structure of the given goal than those employed by other more general operators. When the cited *inform* operator, whose **Effects** field is "bel(H, P)", is compared with the description-grp operator (*frame 5.6*), whose **Effects** field is "bel(H, hold(π, P))", the latter is more specific to the satisfaction of the goal given in expression (5.10). This causes the selection of operator in *frame 5.6*, whose instantiation is as follows:

Frame 5.9. Instantiated operator

Operator:
description-grp(sys, designer, rule, constraints-set)
Restrictions:
set(constraints-set)
Preconditions:
bel(sys, hold(rule, constraint-set))
intend(sys, bel(designer, hold(rule, constraints-set)))
Effects:
bel(designer, hold(rule, constraints-set))
Nucleus:
FORALL P $\in$ constraints-set
bel(designer, rule(P))

With the definition of constraints-set given in expression (5.3), this operator posts three goals to be satisfied in sequence:

$$\text{bel(designer, rule}(\forall x[line(x) \to \forall y[circle(y) \to outside_*(y)(x)]])) \tag{5.11}$$
$$\text{bel(designer, rule}(ghorizontal(l_5))) \tag{5.12}$$
$$\text{bel(designer, rule}(painted\_with(anticorrosive\_paint)(l_3))) \tag{5.13}$$

The goal in expression (5.11) leads to the selection of the operator in *frame 4.7,* instantiated as "state-rule(sys, designer, ($\forall x[line(x) \to \forall y[circle(y) \to outside_*(y)(x)]]$))", to satisfy it. The nucleus of this operator states a new goal, namely:

$$\text{informed(designer, rule}(\forall x[line(x) \to \forall y[circle(y) \to outside_*(y)(x)]])) \tag{5.14}$$

operator 3: "informed(A, rule(P))"
To unify these expressions with the given goal, the matches required for each operator are as follows:
operator 1: informed→informed, A→designer, P→rule(a(X)) = 3 matches in total.
operator 2: informed→informed, A→designer, rule→rule, a→a, Q→X = 5 matches in total.
operator 3: informed→informed, A→designer, rule→rule, P→a(X) = 4 matches in total.
Therefore, in the presented case, **operator 2** is the most specific.

Several speech act operators[11] seem appropriate to satisfy this goal. For instance, an operator "inform1(sys, designer, P_1)" (*frame 4.2*), with P_1 = rule($\forall x[line(x) \rightarrow \forall y[circle(y) \rightarrow outside_*(y)(x)]]$)), could seem viable to satisfy this goal, rendering something like: *"'Every bar lies outside every dielectric-area' is a design constraint"*. Due to the *more-specific* heuristic, however, this operator is discarded in favour of the operator "state-rule1(sys, designer, ($\forall x[line(x) \rightarrow \forall y[circle(y) \rightarrow outside_* (y)(x)]]$))" (*frame 4.8*). Notice that this is the only applicable operator, since all the other operators of the form "state-ruleg<*i*>" are designed for propositions of the form "p(<*args*>)", i.e., they do not accept quantified expressions.

The selected operator finally posts the goal:

$$\text{specify(sys, designer, }(\forall x[line(x) \rightarrow \forall y[circle(y) \rightarrow outside_*(y)(x)]])) \quad (5.15)$$

which is a speech act that, according to expressions (4.15), and (4.14) discussed above, is realised as:

$$\text{"}\textit{every bar must lie outside every dielectric area}\text{"} \quad (5.16)$$

This corresponds to the utterance **[5.3]** of the example dialogue (p. 132).

The goal in expression (5.12) is satisfied by a similar procedure. The operator selected in this case is "state-rule(sys, designer, $ghorizontal(l_5)$)" (*frame 4.7*), which is preferred over "inform(sys, designer, rule($ghorizontal(l_5)$))" due to the *more-specific* heuristic. The selected "state-rule" operator posts the new goal:

$$\text{informed(designer, rule(}ghorizontal(l_5))) \quad (5.17)$$

The operators "state-rule1(sys, designer, $ghorizontal(l_5)$)" (*frame 4.8*), and "state-ruleg1(sys, designer, $ghorizontal(l_5)$)" (*frame 4.9*) are qualified to satisfy this goal. The strategy applied here, to select between these two operators, is that operators that involve deictic acts that can be realised, are preferred over other operators. In this case then, the operator "state-ruleg1" is selected, posting the goal:

$$\text{d-specify(sys, designer, }ghorizontal(\text{deictic}(l_5))) \quad (5.18)$$

which is a surface speech act that, according to expressions (4.19) and (4.20), is realised as:

$$\left\{ \begin{array}{c} highlight(l_5) \\ \text{"}\textit{this bar}\text{"} \end{array} \right\} \text{"}\textit{must be horizontal}\text{"} \quad (5.19)$$

This corresponds to the display **[5.4]** in the example dialogue (p. 132).

Note that a different strategy could be tried with a slight modification of the "state-ruleg1" operator of *frame 4.9*. This strategy presupposes a labelling action before the speech act is issued. The modified operator for this strategy is:

[11]These operators have been detailed in Chapter 4, p. 115.

Frame 5.10.

Operator: state-rulegL(S, H, p(a))
Restrictions:
graphical(a)
Preconditions:
bel(S, not informed(H, rule(p(a))))
intend(S, informed(H, rule(p(a))))
labelled(a)
Effects:
informed(H, rule(p(a)))
Nucleus:
specify(S, H, p(a))
Satellite:
<none>

Here, "labelled(a)" states a goal that is satisfied by a graphical operator that labels the graphical depiction of a with its corresponding name in the linguistic domain. Thus, using the referred strategy, to satisfy the goal in expression (5.17) instead of the display given in expression (5.19), the system would produce the text *"bar_5 must be horizontal"* along with the labelling of the corresponding graphical element with the name "bar_5".

By a similar procedure, the final goal, stated in expression (5.13), is satisfied through the application of the sequence of operators "state-rule(sys, designer, *painted_with*(*anticorrosive_paint*)(l_3))" (*frame 4.7*), and "state-ruleg2(sys, designer, *painted_with*(*anticorrosive_paint*)(l_3))" (*frame 4.10*). Note that, since graphical(*anticorrosive_paint*) is false, neither the operator "state-ruleg3" nor the operator "state-ruleg4" can be applied to satisfy this goal. Then the last operator applied posts the goal:

$$\text{d-specify(sys, designer, } \mathit{painted\_with}(\mathit{anticorrosive\_paint})(\text{deictic}(l_3))) \tag{5.20}$$

which is a speech act that is rendered as

$$\left\{ \begin{array}{c} \mathit{highlight}(l_3) \\ \text{“}\mathit{this\ bar}\text{”} \end{array} \right\} \text{“}\mathit{must\ be\ painted\ with\ anticorrosive\text{-}paint}\text{”} \tag{5.21}$$

This corresponds to the display **[5.5]** of the example dialogue (p. 132).

5.3.2 Explaining the status of satisfied constraints

Regarding the current state of interpretation, constraints are split into two sets: satisfied and unsatisfied constraints. According to this partition, two predicates are introduced to qualify a constraint: *satisfied* and *unsatisfied*. "satisfied(P)" is **true** if P evaluates to **true** in the current state of interpretation, and **false** otherwise. "unsatisfied(P)" is **true** if P evaluates to **false** in the current state of interpretation and **false** otherwise.

Having grouped all satisfied constraints in a set "satisfied-set", explaining them, one by one, to the *designer*, is a matter of posting the goal:

competent(designer,
    comprehend(designer, hold(satisfied, satisfied-set))) (5.22)

which is similar to the goal initially posted to describe the whole set of design rules, given in expression (5.8). As in that case, the most adequate operator to satisfy this goal is "background-of-grp" (*frame 5.7*), which is instantiated as follows:

Frame 5.11. Instantiated operator

Operator:
background-of-grp(sys, designer, satisfied, satisfied-set)
Restrictions:
is-a-set(satisfied-set)
Preconditions:
bel(sys, hold(satisfied, satisfied-set))
intend(sys, competent(designer,
    comprehend(designer, hold(satisfied, satisfied-set))))
Effects:
competent(designer,
    comprehend(designer, hold(satisfied, satisfied-set)))
Satellite:
introducePl(sys, designer, satisfied)
Nucleus:
bel(designer, hold(satisfied, satisfied-set))

The speech act indicated in the satellite of this operator directly produces the text:

"The following are the satisfied-constraints:", (5.23)

and posts the new goal:

bel(designer, hold(satisfied, satisfied-set)) (5.24)

which, in turn, calls for the operator in *frame 5.6*, whose instantiation is as follows:

Frame 5.12. Instantiated operator

Operator:
  description-grp(sys, designer, satisfied, satisfied-set)
Restrictions:
  set(satisfied-set)
Preconditions:
  bel(sys, hold(satisfied, constraint-set))
  intend(sys, bel(designer, hold(satisfied, satisfied-set)))
Effects:
  bel(designer, hold(satisfied, satisfied-set))
Nucleus:
  FORALL P ∈ satisfied-set
    bel(designer, satisfied(P))

All new goals that this operator posts have the form "bel(designer, satisfied(P))". Although there are other operators that can satisfy goals like these, a new more specific operator is introduced here:

Frame 5.13.

Operator: present-satisfied(*S, H,* P)
Restrictions:
  <none>
Preconditions:
  bel(*S,* satisfied(P))
  bel(*S,* not bel(*H,* satisfied(P)))
  intend(*S,* bel(*H,* satisfied(P)))
Effects:
  bel(*H,* satisfied(P))
Nucleus:
  bel(*H,* P)
Satellite:
  <none>

Note that this operator states that for a hearer H to believe that a certain constraint, P, is satisfied, he or she must believe that the proposition P, which embodies the constraint, is **true**.

For instance, taking the constraints set as defined in expression (5.3), in relation to the interpretation state represented in Fig. 5.1, and supposing that "bar_3" is in fact painted with anticorrosive paint in that state, the problem solving module would select the following set of satisfied constraints:

$$\text{satisfied-set} = \{ghorizontal(l_5), painted\_with(anticorrosive\_paint)(l_3)\} \tag{5.25}$$

and, therefore, the goals posted by the operator "present-satisfied" would be as follows:

$$\text{bel(designer, } ghorizontal(l_5)) \tag{5.26}$$
$$\text{bel(designer, } painted\_with(anticorrosive\_paint)(l_3)) \tag{5.27}$$

The operators that best satisfy the goal in expression (5.26) are, applied in sequence: "inform(sys, designer, $ghorizontal(l_5)$)" (*frame 4.1*) and "informg1(sys, designer, $ghorizontal(l_5)$" (*frame 4.3*), and the operators that best satisfy the goal in expression (5.27) are, in sequence: "inform(sys, designer, $painted\_with(anticorrosive\_paint)(l_3)$)" (*frame 4.1*), and "informg2(sys, designer, $painted\_with(anticorrosive\_paint)(l_3)$)" (*frame 4.4*). The application of these operators will produce the following explanations:

$$\left\{ \begin{array}{c} highlight(l_5) \\ \text{“}This\ bar\text{”} \end{array} \right\} \text{“}is\ horizontal\text{”} \tag{5.28}$$

$$\left\{ \begin{array}{c} highlight(l_3) \\ \text{“}This\ bar\text{”} \end{array} \right\} \text{“}is\ painted\ with\ anticorrosive\text{-}paint\text{”} \tag{5.29}$$

which correspond to dialogue portions **[5.8]** and **[5.9]**, respectively.

5.3.3 Explaining the status of unsatisfied constraints

In the example dialogue, the explanation of the status of unsatisfied constraints was shown from dialogue portions **[5.11]** to **[5.12]**. The goal that is posted to produce such explanation was:

$$\begin{array}{l} \text{competent(designer,} \\ \qquad \text{comprehend(designer, hold(unsatisfied, unsatisfied-set)))} \end{array} \tag{5.30}$$

To satisfy this goal, in this case, the *background-grp* (*frame 5.7*), and the *description-grp* (*frame 5.6*) operators are used. When applied, they finally render the goal:

$$\text{bel(designer, unsatisfied(P))} \tag{5.31}$$

where

$$P = \forall x[line(x) \rightarrow \forall y[circle(y) \rightarrow outside_*(y)(x)]] \tag{5.32}$$

A *contrasting* rhetorical relation would help the designer to understand better the contents of a proposition like "unsatisfied(P)". This RST relation is operationalised through the following two operators:

Frame 5.14.

Operator:
present-unsatisfied(S, H, P)
Restrictions:
<none>
Preconditions:
bel(S, unsatisfied(P))
intend(S, bel(H, unsatisfied(P)))
Effects:
bel(H, unsatisfied(P))
Nucleus:
bel(H, differing(
was-rule(P), not(P)))
Satellite:
<none>

Frame 5.15.

Operator: contrast(S, H, π_1, π_2, P)
Restrictions:
<none>
Preconditions:
bel(S, π_1(P))
bel(S, π_2(P))
bel(S, differing(π_1(P), π_2(P)))
intend(S, bel(H, differing(
π_1(P), π_2(P))))
Effects:
bel(H, differing(π_1(P), π_2(P)))
Nucleus:
bel(H, π_1(P))
cue({"but", "however"})
bel(H, π_2(P))
Satellite:
<none>

When applied in conjunction, these operators will yield goals of the form:

$$\text{bel}(H, \text{was-rule(P)}) \tag{5.33}$$
$$\text{bel}(H, \text{not(P)}) \tag{5.34}$$

which, at the end, will in turn be satisfied by the speech act operators of the type *expect*, and *negate* respectively. For the example analysed, the speech acts selected are "expect(S, H, P)", and "negate(S, H, P)", which render the following sentences, respectively:

"Every bar should lie outside every dielectric area",
"but",
"some bars do not lie outside some dielectric areas"

which correspond to utterance **[5.12]** of the example dialogue (p. 135).

When the constraints involving quantified expressions are evaluated against the current state of interpretation, the system records the set of basic propositions (those free of variables) that are responsible for the failure of the universally quantified expression. In the example, the set of individual expressions provoking the failure of the quantified expression (5.32), is:

fail-set =

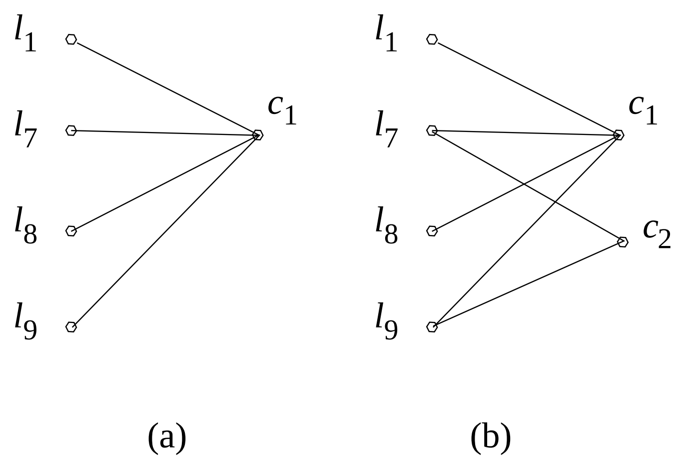


Figure 5.3: Relationships between lines and circles

$$\{outside_*(c_1)(l_1),\ outside_*(c_1)(l_7),\ outside_*(c_1)(l_8),\ outside_*(c_1)(l_9)\} \tag{5.35}$$

This information can be used to construct the explanation that runs from dialogue portions **[5.14]** to **[5.15]** in the example dialogue (p.136). With this aim, it has to be observed what is the relationship among the arguments of the predicate "outside" in the "fail-set" of expression (5.35). This relationship can be represented by the bipartite graph shown in Fig. 5.3(a). Since, just one c_1 is related to all elements in the set $\{l_1, l_7, l_8, l_9\}$, the *definite explanation* that extends the *indefinite* explanation *"some bars do not lie outside some dielectric areas"*, can be generated from a goal like:

$$\text{bel}(H, \text{not}(outside_*(c_1)(\{l_1, l_7, l_8, l_9\}))) \tag{5.36}$$

With the appropriate operators, this goal is satisfied resulting in the following explanation:

$$\left\{ \begin{array}{c} highlight(\{l_1, l_7, l_8, l_9\}) \\ \text{“}these\ bars\text{”} \end{array} \right\} \text{“}do\ not\ lie\ outside\text{”} \left\{ \begin{array}{c} highlight(c_1) \\ \text{“}this\ dielectric\ area\text{”} \end{array} \right\} \tag{5.37}$$

It can be the case, however, that the situation is not as simple as the one depicted in Fig. 5.3(a), but rather something more complex like the relation represented by the graph of Fig. 5.3(b), in which there is more than one dielectric-area. Several other communicative goals can be stated given this relationship. One of these, for instance, would render an explanation like the following:

$$\left\{ \begin{array}{c} highlight(l_9) \\ \text{“}this\ bar\text{”} \end{array} \right\} \text{“}does\ not\ lie\ outside\text{”} \left\{ \begin{array}{c} highlight(\{c_1, c_2\}) \\ \text{“}these\ dielectric\ areas\text{”} \end{array} \right\} \tag{5.38}$$

5.4 State Synthesis Explanations

State synthesis explanations are elaborated to explain how a state was produced from another, i.e., they explain the steps in a problem-solving process. In Section 5.2 the development of one such explanation, starting with the goal in expression 5.4, was illustrated. Two questions however remain: what produced that goal and how was it produced?

In the model proposed here, the synthesis of a state from another is the outcome of the application of one or more problem-solving planning operators. This section shows how the embedding of communicative goals into the problem-solving operators helps the generation of explanations about such syntheses.

Since the problem domain involves a non discrete geometrical space, not all the consequences of any planning action can be stated beforehand. For this reason, the problem-solving module operates following an *interleaved planning and execution* strategy [Tate et al. 1990, McDermott 1978], i.e., each step is executed as soon as it is planned.

To show how the problem-solving operators produce explanations about the problem-solving process, consider the same example proposed in Section 5.1, and the goal in expression 5.4, whose satisfaction produces one such explanation.

The situation depicted in Fig. 5.1, and the set of design constraints given in expression 5.3 show that the set of unsatisfied constraints is:

$$\text{unsatisfied-set} = \{\forall x[line(x) \rightarrow \forall y[circle(y) \rightarrow outside_*(y)(x)]]\} \tag{5.39}$$

This set is the starting point to the problem-solving planning module. This module, looks for an operator, whose **Intended-Effects** field matches one or more of the members in the unsatisfied constraints set. In the case of this example, there is only one unsatisfied constraint, and the operator selected to satisfy it is the following:

Frame 5.16.

P-Operator:
fix-forall-failed(E)
Restrictions:
E = $\forall V\ P$
first-failed(E, Indiv-proposition)
Preconditions:
Indiv-proposition
Actions:
currentState(Sc)
PostCondition = Sc
finalState(PostCondition)
Intended-Effects:
$\forall V\ P$

Problem-solving operators will be called **P-Operators** to distinguish them from the explanation generation operators. In the case shown, there are two restrictions: that E has the form "$\forall V\ P$", where V is a variable and P is a proposition, and the statement "first-failed(E, Indiv-proposition)". This statement looks for the first individual proposition, free of variables, that provoked the failure of the quantified expression. It can be the case that P is in turn a quantified expression: the statement "first-failed(E, Indiv-proposition)", however, will look as deep as necessary to find an individual proposition free of variables. For instance, take E to be the expression "$\forall x[line(x) \rightarrow \forall y[circle(y) \rightarrow outside_*(y)(x)]]$". In this case $V = x$, and $P = line(x) \rightarrow \forall y[circle(y) \rightarrow outside_*(y)(x)]$. According to the example, the expression E is unsatisfied because the following propositions are false: "$outside_*(c_1)(l_1)$", "$outside_*(c_1)(l_7)$", "$outside_*(c_1)(l_8)$", and "$outside_*(c_1)(l_9)$". First-failed will then pick up one of these propositions. Let "$outside_*(c_1)(l_1)$" be the selected proposition as Indiv-proposition. Note that this proposition constitutes the body of the **Preconditions** field of the operator. This means that in order to reach the goals in the **Intended-Effects** field, the goal stated by Indiv-proposition has to be satisfied first. Once this goal is satisfied, the set of actions stated in the **Actions** field, are performed. In this case they are very simple: first, the current state is retrieved, and then it is stated as the post-condition and final state of the operator. In this example, then, the initial precondition is "$outside_*(c_1)(l_1)$", and to satisfy it the following operator is used:

Frame 5.17.

P-Operator:
move-circle-line(L, Ci)
Comment:
This operator will move the circle Ci so that the line L lies outside that circle
Restrictions:
line(L)
circle(Ci)
vertical(L)
Preconditions:
<none>
Actions:
currentState(Sc)
centre(Ci, Co)
radius(Ci, R)
compute-circ-hor-trans(Co, R, L, U, V)
diff-vt(V, U, T)
apply-translation(Sc, Ci, T, PostCondition)
finalState(PostCondition)
communicativeGoal(solved($outside_*$(Ci)(L), Si,
((*past-move*(Ci))(*system*))(*from*(U), *to*(V)), PostCondition))
Intended-Effects:
$outside_*$(Ci)(L)

This operator is applicable since its **Intended-Effects** field matches the posted goal, and the line L = l_1 is a vertical line (see the **Restrictions** field). To evaluate the statements in the **Actions** field the *graphical theory* corresponding to the current state has to be consulted (see expression (5.1)). The statement "compute-circ-hor-trans(Co, R, L, U, V)" computes a translation vector that starts in U and ends in V, as shown in Fig. 5.4. The actual translation vector, T, is found by the vectorial difference V minus U. This translation, T, is applied (apply-translation) to the graphical object Ci giving as a result a new state, called PostCondition, which is reported as the final state of the operator.

From the explanation generation standpoint, the keynote is the statement "communicativeGoal". A problem-solving operator can have as many statements like this as it requires to build up its explanation. The communicative goals stated by this statement, are written down in a list that will be used to build up the explanation of a problem-solving process step by step. The particular

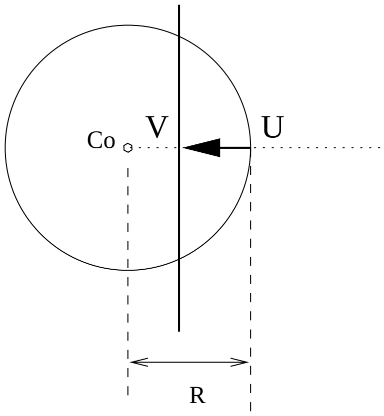


Figure 5.4: Computation of a circle displacement

communicative goal shown in this operator (*frame 5.17*), is satisfied as shown in the example of subsection 5.2.1.

To finish up with this example, once the operator *move-circle-line* (*frame 5.17*) has been applied, an evaluation of the original quantified expression in (5.39) is performed to see if it is already satisfied. Here, however, this will not be the case. Suppose now that "first-failed(E, Indiv-proposition)" makes the assignment "Indiv-proposition = $outside_*(c_1)(l_7)$". The satisfaction of this goal will require the application of operators like the following:

Frame 5.18.

P-Operator:
outwards-line-circle(L, Ci)
Restrictions:
not(marked(nearest(L, Ci)))
not(vertical(L))
currentState(Sc)
line(L)
circle(Ci)
is-part-of(L, P)
polyline(P)
centre(Ci, Co)
nearest-line(Sc, P, Co, Ln)
mark(nearest(Ln, Ci)
Preconditions:
outside$_*$(Ci)(Ln)
Actions:
currentState(PostCondition)
finalState(PostCondition)
Intended-Effects:
outside$_*$(Ci)(L)

Frame 5.19.

P-Operator:
vertical-move-obj(L, Obj, Ci)
Restrictions:
line(L)
circle(Ci)
(L = Obj **or** is-part-of(L, Obj))
Preconditions:
<none>
Actions:
currentState(Sc)
centre(Sc, Ci, Co)
radius(Sc, Ci, R)
compute-vert-translation(Co, R,
L, U, V)
diff-vt(V, U, T)
apply-translation(Sc, Obj, T,
PostCondition)
unmark(nearest(L, Ci)
finalState(PostCondition)
communicativeGoal(solved(
outside$_*$(Ci)(L), Sc,
((*past-move*(Obj))(*system*))
(*from*(U), *to*(V)),
PostCondition))
Intended-Effects:
outside$_*$(Ci)(L)

The *outwards-line-circle* operator (*frame 5.18*) is tried first to satisfy the goal "*outside*$_*(c_1)(l_7)$". This operator finds out that the line l_7 is part of another more complex object P, which is a polyline. Then, the nearest line to the center of the circle of those composing P is discovered, and marked as such. With this line, Ln, a precondition "*outside*$_*$(Ci)(Ln)" is stated. The statements "not(marked(nearest(L, Ci))", and "vertical(L)", in the **Restrictions** fields of the operators in *frames 5.17* and *5.18*, respectively, prevent their use to satisfy this precondition. Thus, the precondition is satisfied with the operator in *frame 5.19*. This operator will move the whole object containing the line by calculating and applying the vertical translation given by the vector that starts in U and ends in V, as shown in Fig. 5.5.

The communicative goal written down by the *vertical-move-obj* operator (*frame*

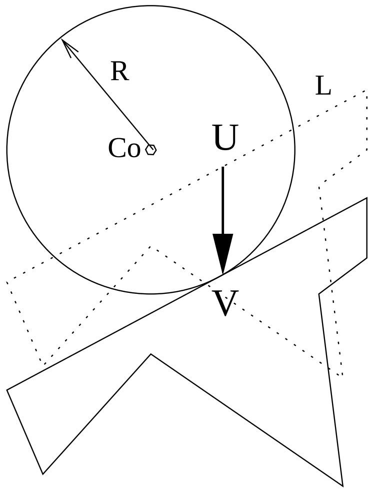


Figure 5.5: Moving a polyline outside of a circle

5.19), will be satisfied by the *solutionhood* operator of *frame 5.4*, as shown in the example of subsection 5.2.1.

5.5 Summary

An operationalisation of Mann and Thompson's RST schemata for the generation of structured discourses aimed at explaining design situations was motivated and developed. The planning operators for the production of such explanations, are capable of producing textual utterances that include deictic references to the graphical objects representing the domain of application. Furthermore, the designed *rhetorical* operators can deal with explanations about the steps in a problem-solving process.

It should be noted that the explanation plan operators designed in this work have some differences with respect to those developed in the previous work, described in Chapter *2* of this thesis, that also utilised a plan based approach to explanations generation. These differences are summarised in Table 5.1. In this table, references to fields in particular operators are shown in bold typeface. The first column identifies the system or the author: Hovy [Hovy 1991, Hovy 1993], Moore [Moore & Paris 1989, Moore 1995], Maybury [Maybury 1992, Maybury 1993*a*], WIP [Wahlster et al. 1993, André et al. 1993], and Xlogos stands for the work presented in this thesis. The second column states whether the operators designed by the author or for the system operationalise Mann & Thompson's rhetorical structure theory. The third column indicates the strategy followed to select an operator. The fourth column describes how the rhetorical relations are established by the plan operators. Finally, the fifth column states whether

Authors/ Systems	M&T's RST?	Match fields	Rhetorical relation established by	Conditions
Hovy	Yes	1 goal in **results** field	**Nucleus/Satellite** definable order	implicit in **Nucleus/Satellite**
Moore	Yes	1 goal in **effects** field	**Nucleus/Satellite** fixed order: N→S	unsatisfied **constraints** rule out operator
Maybury	No	either **header** field or multiple goals in **effects** field	Non-categorised subgoals in **decomposition** field	applicability **constraints** and satisfiable **preconditions**
WIP	Partial	either **header** field or 1 goal in **effects** field	Multiple subgoals in **Main/Subsidiary** acts fixed order Ma→Sa	satisfiable **applicability-conditions**
Xlogos	Yes	1 goal in **effects** field	**Nucleus/Satellites** definable order	applicability **restrictions** satisfiable **preconditions**

Table 5.1: Plan operator roles in different systems/authors

the operators provide a way to specify preconditions that require to be satisfied prior to the application of an operator, and whether they provide a way to specify restrictions or applicability conditions for an operator.

The operators designed in this work are selected according to one goal in the effects field, because this directly relates the intentional goals with the different spans of text produced. These operators provide a flexible way of establishing rhetorical relations by enabling the definition of a nucleus (composed of one or more subgoals), and as many satellites as required in a definable order. Finally, they provide a way to define when a particular operator is applicable, and a way to define a set of preconditions that can be satisfied prior to the application of a particular operator. This is in contrast to the approach adopted in McKeown's work, where ATN are used and result in a more operational scheme that focuses on defining *how* to generate an explanatory discourse.

The surface realisation of the *rhetorical* operators is accomplished by a set of basic operators that operationalise *speech acts* and *coordinated linguistic and graphical acts* (COLGRAs), as was described in Chapter 4. These operators are defined using logical forms that constitute the input of the natural language grammar module to produce the required sentences (as described in Section A.3.3 of Appendix A, 204).

The use of the set of operators was illustrated with an example taken from the geometrical design of overhead transmission line towers. The example exhibits the rendering of simple assertions with deictic references, sequences of assertions, contrasting paragraphs, assertions that include quantified expressions, deictic referring expressions with definite plural noun phrases, and displays explaining the changes of state produced by the problem-solving strategy. The example also shows how the embedding of communicative goals in the problem-solving planning operators can easily produce explanations about the steps followed in the problem-solving process.

Chapter 6

PROOF OF CONCEPT: THE XLOGOS SYSTEM

The previous chapters of the thesis have developed the underlying theory for a system for generating coordinated multimodal explanations. As described in Chapter 2, the main driving force for the theory was to develop an instance of the kind of system represented by the node named *multimodal detached modalities explanation generation system* in Fig. 2.23 (p. 55). The aim of this chapter is to demonstrate the feasibility of implementing such a system based on the developed theory. A full and complete implementation is not feasible in the time scale available for this research[1], and therefore the prototype was developed centred on illustrating the viability of generating the kind of explanations shown in Chapters 4 and 5 by implementing key aspects of the theory. This prototype was fully developed by the author of this thesis.

This chapter first presents the general architecture of the system and then describes the status of the modules in relation to the implementation and the theory. This is followed by a sample dialog with the implemented system.

Figure 6.1 extends the architecture shown previously in Fig. 1.6 (p. 17). This sketch exhibits modules of two kinds, namely: processes, represented by rectangular boxes, and knowledge bases or information sources, represented by *cylindrical* boxes. Depending on whether or not the user can modify them, the knowledge bases are dynamic or static. In this architecture, the graphical theory, the non-graphical theory, the COLGSUs knowledge base, the design constraints, and the system and user's beliefs model are dynamic, whereas the languages grammars, the problem solving operators, and the explanation generation operators are static.

---

[1]For example, the DenK project has numerous researchers and is a long term effort that started in 1989 and is expected to run until 1998 [Bunt et al. 1998], and the WIP system took nearly five years to be developed by at least five researchers [Wahlster et al. 1993].

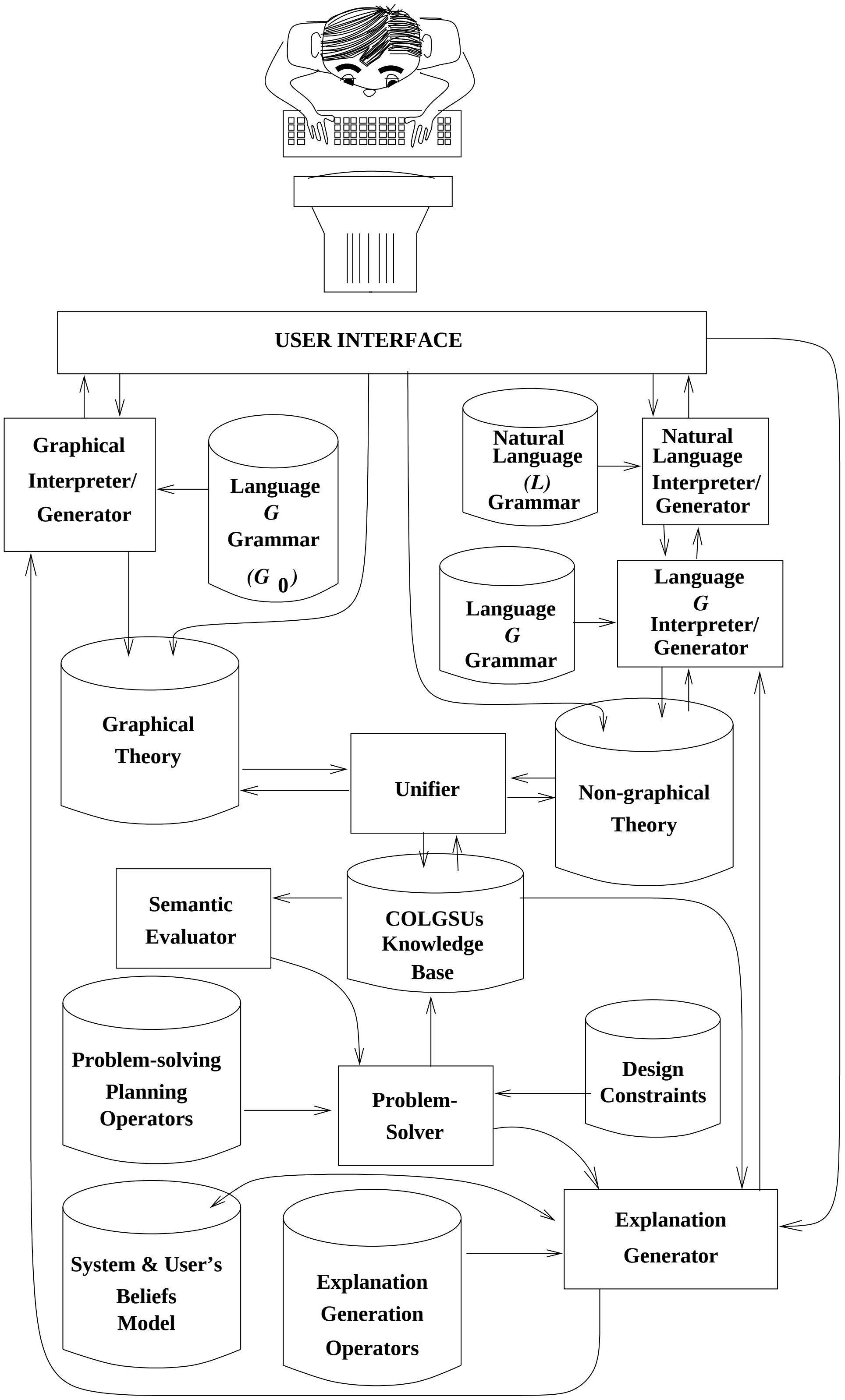


Figure 6.1: Architecture of the Xlogos system

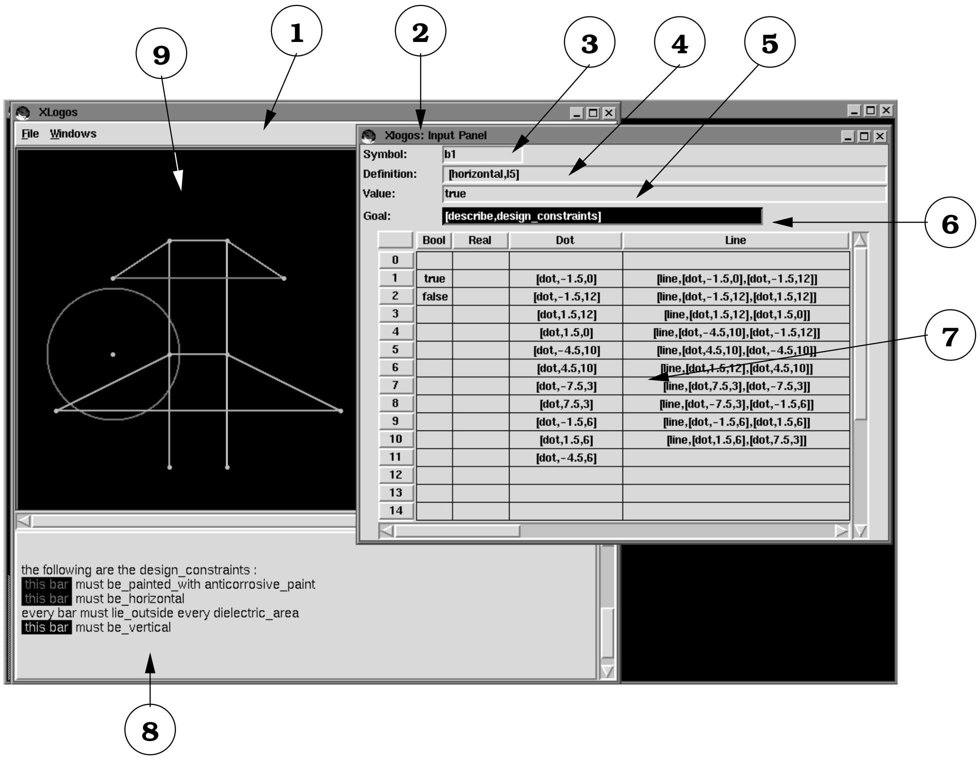


Figure 6.2: Xlogos user interface

In what follows, the modules composing this architecture and their interactions are explained.

6.1 Modules

The following subsections describe the modules, the extent to which they have been implemented and how they are related to the theory. The prototype system has been developed in BinProlog [Tarau 1997] and uses the script languages Tk/Tcl [Welch 1995].

6.1.1 User interface

The user interface module is the communication medium from and to the user. This module is designed to support input and output, and to enable a display of the graphical and non-graphical theories. The user interface also allows the direct manipulation of the graphical objects.

The interface was implemented using the script languages Tk/Tcl and takes the form shown in Fig. 6.2. This figure shows the following user interface components:

1. The main output panel. This groups the two main output components: the textual output window and the graphical output window.

2. The main input panel. This groups the main input components: a row for entry and display of the currently selected symbol, a row for entry and display of the definition of the selected symbol, a row for the display of the semantic value of the selected symbol, a row for entries of user's commands or goals, and a table for presentation and selection of graphical symbols.

3. Entry row and display of the current selected symbol. This row allows the introduction of a graphical symbol, which becomes the *current selected symbol*. This symbol can also be defined by two other methods:

   (a) pointing to a graphical depiction of the graphical symbol with the cursor in the graphical output window and clicking the left button of the mouse,

   (b) selecting a cell in the spread-sheet style table and clicking the left button of the mouse.

   In both cases the selected symbol will be displayed in this entry and display row of the current selected symbol.

4. Entry row and display of the current symbol definition. When a symbol is selected, its current definition is displayed in this row. A new definition of the selected symbol can be introduced in this row.

5. Display of the current symbol's semantic value. The semantic value of the current selected symbol is displayed in this row.

6. Entry row for a user command or goal. The user's commands are introduced in this row. This row also allows the introduction of communicative goals to be resolved by the explanation generation planner (explanation generator module), and it even allows the introduction of Prolog goals for debugging purposes. This feature was added to overcome the limitations of the simple command language provided.

7. Spread-sheet style table for presentation and selection of graphical symbols. This table shows the semantic values of the graphical symbols. The table is organised by columns representing graphical types, and rows representing order numbers. For instance, the cell under the column labelled with **Line**, and row two, represents the second symbol of type *line*, i.e.,

the symbol l_2 of the language G. If this cell is selected by pressing the left button of the mouse while the cursor is in the cell, the symbol becomes the current selected symbol.

8. Textual output window. This window shows the natural language textual output.

9. Graphical output window. This window shows the graphical depictions of the symbols of G that are graphically interpretable.

6.1.2 Natural language interpreter/generator and grammar

The natural language (NL) module is designed to have two functionalities resting on the same grammar: the interpretation and the generation of the natural language texts. The NL generator has been built up so that it is able to synthesise the surface text from the logical forms provided by the logical language G.

The NL grammar is a set of production rules written in the PATR formalism, which is given in Appendix A, and is restricted to a small fragment of the English language.

The current implementation does not includes the NL interpretation part, since the main focus of the thesis is on generation. Thus at present, constraints and propositions are input directly as expressions of language G, as defined in Chapter 3.

6.1.3 Language G interpreter/generator and grammar

Language G serves as the semantic logical form input for the natural language generator during the NL generation phase, and it is also designed to enable a semantic evaluation of natural language during a NL interpretation phase, though this phase has not been implemented.

6.1.4 Graphical interpreter/generator

The graphical interpreter/generator has been designed to interpret the expressions of language G that are graphically interpretable, producing a visualisation of them in the graphical area provided by the user interface. In the current implementation, it is possible to define the value of a symbol in the input panel and its graphical depiction will be displayed in the graphical window. The present implementation includes definitions of circle, line, dot, and polyline. This interface has been implemented by means of the graphical facilities provided by the script language Tk, and could be extended so that graphical objects could be added in the graphical window and result in expressions in the language G.

6.1.5 Graphical and non-graphical theories

The graphical and non-graphical theories contain the facts from the language G that represent the model of the world. The graphical and non-graphical theories are then the expressions specified by the user or the system in each knowledge-base and their logical consequences.

6.1.6 Unifier and COLGSUs knowledge base

Since both the graphical and the non-graphical theories are about the same universe of discourse, there must be a unified or *coordinated* theory. The purpose of the unifier module is to produce such a unified theory by means of the principles described in Chapter 3. The unifier will write each *coordinated linguistic and graphical semantic unit* (COLGSU) corresponding to each interpretation state in the corresponding knowledge base.

6.1.7 Semantic evaluator and design constraints

The semantic evaluator is a module that tests the consistency of the current COLGSU against a set of design constraints. This aspect of the system fully implements the semantic evaluations described in Chapter 3.

6.1.8 Problem solver and problem solving operators

The problem solver is a module that operates when one or more of the design constraints are not satisfied. In this case, it consults its problem-solving operators knowledge base and issues a plan to satisfy the unsatisfied constraints. This module can update the COLGSUs knowledge base. The problem-solving planner has been implemented based on means-ends analysis [Bratko 1990] and operators were defined for solving a particular problem namely: to ensure that objects are outside a dielectric area. The operators defined include all those presented in Chapter 5.

6.1.9 Explanation generator and explanation generation operators

This module can generate explanations about a particular state represented by a COLGSU, or about each of the steps performed by the problem solver in the problem-solving process. To generate such explanations, it makes use of the explanation generation operators, as described and illustrated in Chapter 5. This is a significant part of the thesis and therefore this component was fully implemented and all the operators described in Chapters 4 and 5 have been defined.

6.1.10 System and user's beliefs model

In Chapter 5 (p. 142) it was observed that a number of authors expressed the view that tracking intentions explicitly is an important part of the discourse representation. Hence, although this is not the focus of this thesis, a simple user belief model was implemented to take advantage of the information available in the operationalisation of the RST adopted in this thesis. That is, the information about what the user knows is used to avoid explaining what the user already believes.

6.2 Interactions and Sample Dialogue

This subsection presents a sample dialogue to illustrate what is possible with the current implementation.

Initially, the user introduces a series of expressions of language G, representing the first state of his or her design. These expressions can be introduced in the form of two files, as shown in Fig. 6.3, which the system is instructed to read, or, alternatively, they can be directly typed in the input panel. These expressions constitute the initial graphical and non-graphical theories which are stored in the corresponding knowledge bases. The unifier takes a pair composed of a graphical theory and a non-graphical theory, along with a translation function and produces a COLGSU representing the current state of interpretation. This state is evaluated semantically paying special attention to a set of propositions that were defined to be the design constraints. If some of these constraints are not satisfied the problem solver is invoked. The problem solver takes these constraints one by one and tries to satisfy them using the problem solving planning operators stored in the corresponding knowledge base. The new states, represented by new COLGSUs, produced by this problem solving process are stored in the COLGSUs' knowledge base and evaluated again. This cycle stops when all the design constraints are satisfied. In order to elaborate a subsequent explanation about the problem solving process carried out, the problem solver annotates a series of communicative goals that will be used in case the user requests such explanation.

The user can ask the system to produce an explanation about the current state of interpretation or about the problem solving process carried out. The explanation generator receives the user's query and posts a communicative goal that is interpreted according to the available explanation generation operators and the user's beliefs to produce a plan that will generate the required explanation. In Fig. 6.4, for instance, the user has asked the system to describe the unsatisfied design constraints. The system has issued an explanation about the

```
d1 = [dot, -1.5, 0.0].
d2 = [dot, -1.5, 12.0].
d3 = [dot, 1.5, 12.0].
d4 = [dot, 1.5, 0.0].
d5 = [dot, -4.5, 10.0].
d6 = [dot, 4.5, 10.0].
d7 = [dot, -7.5, 3.0].
d8 = [dot, 7.5, 3.0].
d9 = [dot, -1.5, 6.0].
d10 = [dot, 1.5, 6.0].
d11 = [dot, -4.5, 6.0].
l1 = [line, d1, d2].
l2 = [line, d2, d3].
l3 = [line, d3, d4].
l4 = [line, d5, d2].
l5 = [line, d6, d5].
l6 = [line, d3, d6].
l7 = [line, d8, d7].
l8 = [line, d7, d9].
l9 = [line, d9, d10].
l10 = [line, d10, d8].
c1 = [circle, d11, 3.5].
k1 = [path, l4, [path, l2, [path, l6, l5]]].
k2 = [path, l7, [path, l8, [path, l9, l10]]].
b1 = [horizontal, l5].
b2 = [forall(vl1),[forall(vc1),[outside,vl1,vc1]]].
```

```
nlFact([be_painted_with, bar1, luminous_paint]).
nlFact([be_painted_with, bar2, luminous_paint]).
nlFact([be_painted_with, bar3, anticorrosive_paint]).
nlFact([be_painted_with, bar4, anticorrosive_paint]).
nlFact([be_painted_with, bar5, anticorrosive_paint]).
nlFact([be_painted_with, bar6, anticorrosive_paint]).
nlFact([be_painted_with, bar7, anticorrosive_paint]).
nlFact([be_painted_with, bar8, luminous_paint]).
nlFact([be_painted_with, bar9, anticorrosive_paint]).
nlFact([be_painted_with, bar10, luminous_paint]).
```

Figure 6.3: Input files containing expressions of language *G*.

only unsatisfied constraint, namely, that some bars do not lie outside the specified dielectric area. Since this constraint is not directly observable in the graphical depiction of the model, in Fig. 6.5, the user asks for further details about the previous explanation. The new explanation, shown in that figure, specifically indicates, by means of a deictic reference, which bars do not lie outside the dielectric area.

Currently, the user instructs the system to build up an explanation or to solve a problem using the following commands that cover the examples presented in this thesis:

1. **[describe, design_constraints]**. Instructs the system to describe the design constraints one by one .

2. **[describe, satisfied]**. Instructs the system to describe the design constraints that are satisfied in the current state of interpretation.

3. **[describe, unsatisfied]**. Instructs the system to describe the design constraints that are not satisfied in the current state of interpretation.

4. **[detail, quantified_expression]**. Instructs the system to give details about the last quantified expression computed by the system.

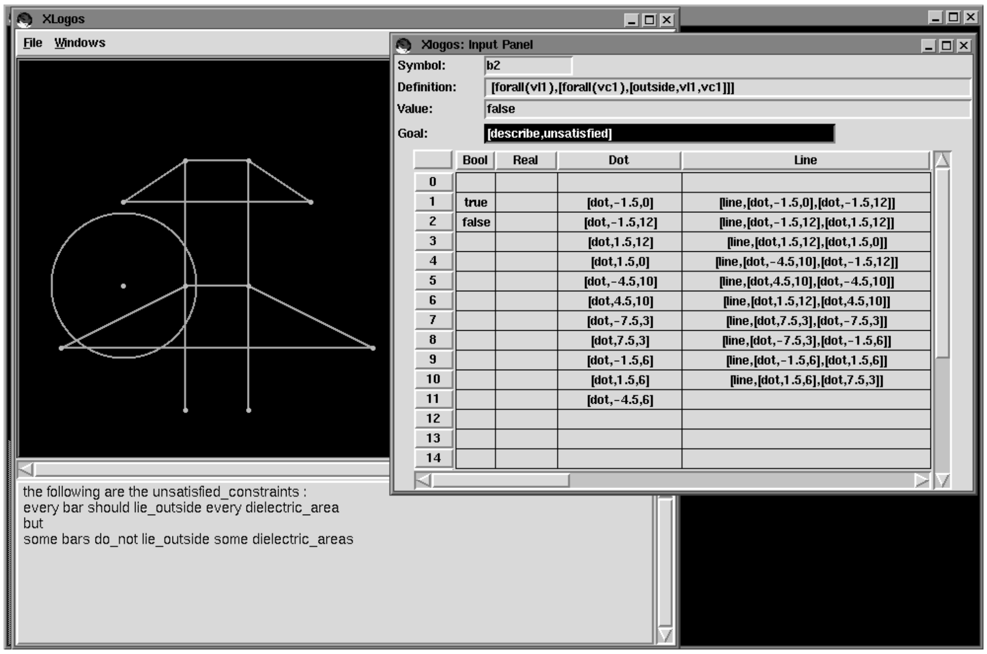


Figure 6.4: Asking for unsatisfied design constraints

5. **[solve, problem]**. Instructs the system to solve the problem of satisfying the unsatisfied design constraints.

6. **[explain, problem_solution]**. Instructs the system to explain, step by step the process followed to solve a design constraint satisfaction problem.

Since the above *command language* is rather limited, the system provides the possibility of directly introduce a communicative goal that can be used to produce an explanation plan. This plan, can in turn, be linearised as the desired explanation. For instance, if the following line is introduced in the entry row for a user command or goal,

(nPlanner([informed(user,[horizontal,l5])],P), execute_plan(P))

the system will generate an explanation informing that bar_5 (which corresponds to symbol l_5 of language G) is horizontal.

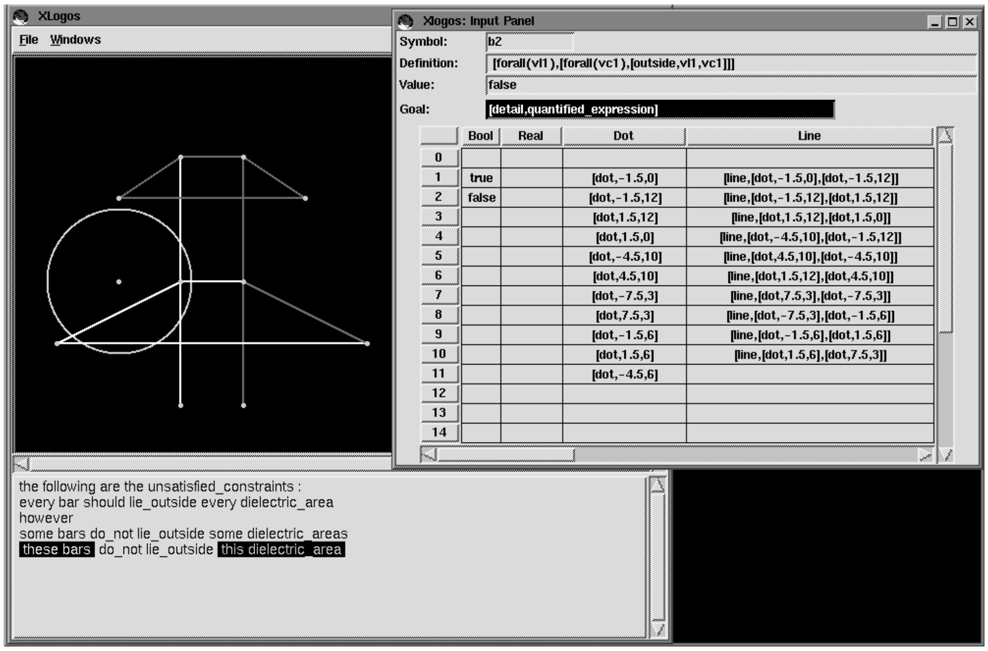


Figure 6.5: Asking for details about a quantified expression

Chapter 7

CONCLUSIONS AND FUTURE WORK

The main goal of this thesis was to introduce a new category of explanation generation systems with multimodal capabilities that is able to answer new types of questions with a structured discourse, such as those required by design oriented tasks. With this purpose, the development of a multimodal explanation generation prototype system was outlined in the foregoing chapters. This concluding chapter summarises the most important issues that have arisen in the development of the thesis. It also presents a concise comparison with other multimodal explanation generation systems, and, finally, defines some future directions for the work presented here.

7.1 Concluding Remarks

The development of a linguistic and graphical system for design requires the solution of two main problems:

1. What is a good way of representing and relating the two modalities so that the system is in a position to answer a wide range of queries?

2. How can explanations, developing the answers to these queries, be structured and include deictic references?

The first question was addressed in Chapter 3 of the thesis where a *coordinated linguistic and graphical semantic unit* (COLGSU) was introduced for representing either linguistic or graphical facts or assertions. This representation enabled the framework shown in Fig. 7.1. The natural language (L) semantics is defined through its translation into a logical language G, whose model-theoretic interpretation provides indirectly the semantics of L, following in this way the so

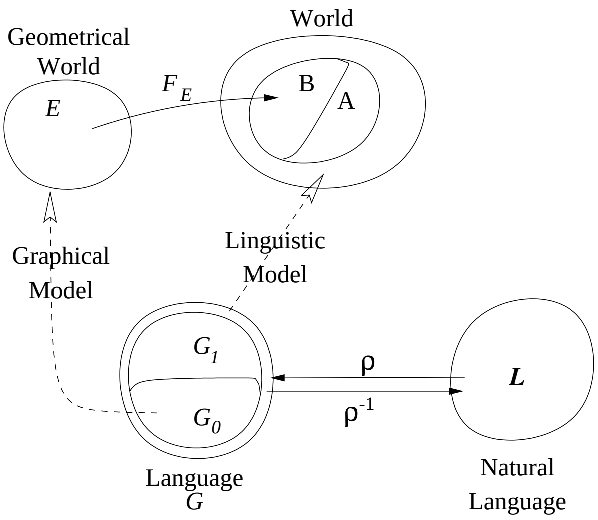


Figure 7.1: Models of a universe of discourse

called Montague's *indirect via* for the interpretation of natural language [Montague 1974*d*]. Language G has two interpretation domains, namely: the world (W), and a geometrical world (E), which is related to the world through function F_E. Language G expressions are partitioned into those expressions that are geometrically interpretable (G_0), and those that are not (G_1). The interpretation of G (or more precisely, of G_0) with respect to the geometrical world (E) constitutes the graphical model, whereas, the interpretation of G with respect to the world constitutes the linguistic model.

A COLGSU is defined in Chapter 3 in terms of a logical language, G, a natural language, L, and a translation relation between them, ρ. Both languages, G and L, along with their translation relation, are introduced through the definition of their syntax and semantics in Chapter 3. Specifically, the semantics of G can be evaluated according to either the graphical or the linguistic models. Separating the linguistic and the graphical parts in this way, results in modular semantics and in better foundations. In particular, it facilitates better utilisation of the semantics of the graphical part of the languages. That is, the graphical modality is used as a knowledge source and not just as an illustrative vehicle. To exemplify this, and to clarify what is meant by *illustrative vehicle*, consider what most multimodal systems do to introduce the graphical modality: they call a graphical synthesiser which is in charge of illustrating the depictable facts asserted in the knowledge base. An object, graphical or not, is described in one block that groups all its attributes. For instance, consider a representation of a particular town, that could fit Maybury's multimodal explanation planner [Maybury 1993*a*]:

- Chemnitz: Town

- **Class** Town:
  - Gtype: Cartographic
  - Location: dot

This description says that *Chemnitz* is an entity of type *Town*, which is in turn a class of *Cartographic* objects, whose *Location* attribute is a dot. With this sort of representation, the geometrical location of the alluded object can be recovered by invoking something like *Chemnitz.Location*. For this example, say that *Chemnitz.Location = the_dot(12.80, 50.82)*. Thus, one knows that the location of Chemnitz is a dot whose coordinates are *(12.80, 50.82)*, and this dot can be *depicted*, *highlighted*, or *the region around it can be displayed*, etc., in the graphical representation. In other words, the fact that Chemnitz has a certain location can be graphically *illustrated*. However, if it is asked what a particular dot, say *the_dot(12.80, 50.82)*, represents, it is hard to see how, from the given representation, such information can be easily extracted. This approach, then, allows easily to go from the application domain to the graphical representation, but not the other way around, i.e., to extract conclusions from the graphical representation is a hard task. On the contrary, the approach presented in this thesis facilitates this sort of task. For instance, that some dots or lines are not outside a given circle, and its corresponding meaning in the domain of application, is not something stated beforehand in the knowledge base. However, by a semantic evaluation of the expressions stating it, the conclusion can easily be extracted, as was shown in Chapter 3.

The second problem, that of how to generate structured explanations, was addressed in Chapters 4 and 5 by taking advantage of previous work on speech acts and rhetorical structure theory. Chapter 4 operationalised Searle's speech acts theory by defining operators for rendering text and graphics. Deictic references were introduced as part of speech acts. Chapter 5 took advantage of Mann & Thompson's work on rhetorical structure theory (RST) and developed operators for explaining design constraints and problem solving steps. These operators are able to deal with simple assertions, deictic expressions, sequences of assertions, contrasting paragraphs, quantified expressions, and definite plural noun phrases.

Although some of the problems addressed in this thesis have been individually dealt with by others, here a coherent integration of all them has been outlined and examined through the development of a prototype system that demonstrates the viability of the proposed approach.

7.2 Comparison with Related Work

Chapter 2 of the thesis presented a detailed survey of current research on systems for generating explanations. This section highlights the main differences

	XTRA	MAP	COMET	GRAFLOG	WIP	Xlogos
Graphics generation	static 2D	static 2D	dynamic 3D	dynamic 2D	dynamic 3D	dynamic 2D
Geometric model	static	static	static	dynamic	static	dynamic
Semantic representation	formal NL sem ↓ geometric model	formal NL sem ↓ geometric model	formal NL sem ↓ geometric model	formal NL sem $*$ ↓ World ↑ formal GL sem	formal NL sem ↓ geometric model	formal NL sem ↕ formal lang.G sem ↙ ↘ *E* *W*
User model	Yes	User beliefs model		No	User beliefs model	User beliefs model
Use of graphics	illustrative	illustrative	illustrative	source of knowledge	illustrative	source of knowledge
Intentional structure	No	Intentional goals	No	No	Intentional goals	Intentional goals
Rhetorical structure	?	Rhetorical operators	Schemata	No	Rhetorical operators	Rhetorical operators
Deictic references	Yes	Yes	No	No	No	Yes

Table 7.1: Comparison with other approaches

between the research presented in this thesis and a selection of the representative multimodal systems presented in Chapter 2. Table 7.1 presents a comparison of the approach presented in this thesis, under column *Xlogos*, with the approaches followed by the other multimodal systems: XTRA [Wahlster 1991], MAP [Maybury 1993*a*], COMET [Feiner & McKeown 1991], GRAFLOG [Pineda 1989], and WIP [Wahlster et al. 1993]. The main differences, with respect to the criteria presented in the first column, are described below.

Graphics generation and geometric model. Most current multimodal systems work with a static geometrical model of the objects. Even though COMET and WIP dynamically generate different views of these objects as part of their explanations, the geometrical models they use are perfectly defined and known beforehand. In a system aimed at geometrical design, like the one presented here, the geometrical model of the objects being defined are always evolving. In general, the geometric model will be different from one state to another. This is why these systems do not address the problem of explaining how one state was synthesised from another, and, consequently, they are not aimed at responding to questions of this nature. Even very recent systems like DenK [Bunt et al. 1998], that deal with a dynamic model, do not address the generation of this sort of explanation.

Semantic representation. The symbology shown in the table in this row means that most of the other systems do not utilise graphical objects as a source of information for the semantic evaluation of natural language expressions. Instead, having assigned semantic values to natural language expressions, they compose a geometric model that reflects the natural language semantic evaluations. That is, there is no formal language, in the logical sense, to represent the graphical entities describing the world. Instead, the geometrical objects are manipulated directly when required in a pragmatic manner. This means that all the reasoning process has to be done on the natural language side. It is not possible, for instance, to answer whether a certain switch is turned on or off unless the fact is asserted in the semantics of the natural language part of the system. In the work proposed here, *Xlogos*, two formal languages were defined: a natural language (a fragment of the English language), and a logical language with two possible domains of interpretation: the world (W) and a geometrical world (E), so that the reasoning process can take place on either side of the scene. A question like the previous one, could be answered not only by utilising information expressed in natural language, but also by the semantic evaluation of the geometrical configuration of the objects represented in the logical language. In this case, instead of having the graphical part as an illustration of the natural language semantics, both parts, the linguistic and the graphical, provide semantic information about the world or universe of discourse for the natural language expressions to be evaluated. Incipiently, the GRAFLOG system had a representation similar to this. It, however, had a language associated to each modality and did not reach the point of producing natural language explanations to answer questions like these.

User model. Most of the systems reported in the table work with a user model that is updated as the session proceeds. WIP, for instance, as part of its user model, maintains a representation about what the user believes. This representation is updated with the "effect" field of every presentation strategy which succeeds in the course of a session. *Xlogos* follows a similar approach. The changes of state, however, add some complexity to the maintenance of this model, since facts that the user believed to be true in a state, could be false in a new generated state. However, none of these systems are focused on user modelling to the extent that they utilise recent developments in user modelling (e.g., anticipation feedback [Wahlster & Kobsa 1986, Wahlster 1988]).

Use of graphics. Due to the semantic representation adopted, most multimodal explanation generation systems fall into an illustrative use of the graphical modality. *Xlogos* exploits the semantics of the graphical modality as a source of knowledge, enabling the extraction of conclusions from the graphical representation. Although during an interactive session, GRAFLOG associates each graphical object with a concept that can be expressed through natural language, and thus the graphical information can be used in the problem-solving process, it does not utilise this information to produce a combined natural language and graphics explanation as *Xlogos* does.

Intentional structure. This row shows how the intentions of the system to change the mental state of the user are taken into account for the generation of the explanation. That is, behind the discourse structure there is a structure of intentions explicitly established by the resolution of intentional goals. MAP and WIP build up an intentional structure based on the intentional goals stated in their explanation planning operators. However, since Xlogos conforms to Mann & Thompson (1988) *rhetorical structure theory*, the way in which it links the intentions to the rhetorical structure differs from that of those systems.

Rhetorical structure. In this entry it is shown if the explanations produced by the systems have a rhetorical structure, and what is the strategy to generate such a structure. Basically, there are two approaches: a rhetorical structure achieved by the application of planning operators, and McKeown (1982) style schemata, which are augmented transition networks for the automated production of the explanation according to a general purpose of the discourse. The approach followed in this thesis is that of rhetorical planning operators, which offers a more flexible way of generating the explanation plan, since the explanation generation plan is divided into small parts, each corresponding to a single operator, whereas the schemata approach produces the whole plan in one shot. Although MAP and WIP followed a similar approach based on rhetorical operators, the one presented in this thesis tried to comply more with Mann & Thompson (1988) *rhetorical structure theory*, which has the support of numerous analyses of real discourses.

Deictic references. In this entry it is signaled which systems include deictic references in the explanations produced.

7.3 Future Directions

The first two chapters of this thesis introduced several of the desirable features for multimodal systems oriented towards design tasks. Some of them were directly addressed in the foregoing chapters of this thesis. This section is devoted to discuss the future directions of the work presented here having as background those features not fully covered in this thesis.

Most design tasks require a cooperative participation of both interlocutors in the problem-solving process. This implies that the system has to play the role of either the instructor, who knows about the design rules and restrictions, or the designer, who actually performs the design task. For a system to play any of these roles, it must be able not only to generate plans to solve problems and produce explanations, but to recognise the underlying plan in the discourse and interaction of the other interlocutor. Hence, it is necessary not only to recognise the natural language utterances with their corresponding graphical interactions, but to recognise them in the context of a plan to perform a design task [Allen 1983, Carberry 1990, Kautz 1991]. Since the ground for interpreting at the natural language utterances level was given in Chapter 3, an interesting line of future investigation is open, namely, how the framework developed in this thesis can help the recognition of such plans in the context of multimodal systems.

The identification of the referent in a referring expression is a crucial problem of natural language interpretation systems addressed by several authors from different perspectives [Kaplan 1978, Appelt & Kronfeld 1987, Kamp & Reyle 1993, André & Rist 1994, Pineda & Garza 1997]. A desirable feature of multimodal systems for design tasks is the capability of associating the natural language referring expressions to their corresponding graphical depictions, no matter which ones are introduced first. In the simple model presented in this thesis such association is made ostensively, thus not requiring the system to infer it. Therefore, an issue that requires further investigation is that of the incorporation of a mechanism for the resolution of multimodal references, probably following the lines proposed by Pineda & Garza (1998).

Another problem related to the previous one is that of the generation of referring expressions. An interesting matter could be to explore how the theoretical material about *emergent objects* presented in Section 3.4 (p. 104) can be used to support algorithms for the generation of referring expressions such as the one developed by Dale & Reiter (1995).

The rhetorical operators developed in Chapter 5 are used to construct a structure of the intentions behind the generated discourse. In the work presented here, however, such rich structure was not exploited in full. Moore (1995) showed how this structure along with the *design rationale*[1] of the system, can be utilised to answer follow-up questions, i.e., questions about the reasons of the system's actions, and to provide alternative explanations when the system detects that the given one was not understood. An extension of this thesis' work in such a direction can be envisaged. Furthermore, taking advantage of the interleaving of the problem solving plan with the communicative goals, this extension could improve over the proposal of alternative explanations to the failed ones, by proposing alternative solutions to the given design problem.

Pineda & Garza (1998) have worked with a multimodal framework similar to the one presented in this thesis, though aimed at different goals and domain

[1] A *design rationale* of a system can be understood as a record of the reasons underlying each particular decision followed by the system during a problem solving process.

of application. Their work has explored the use of a pictorial language, based in some of the principles of Mackinlay (1987), whose lexical elements are the geometrical objects, and whose syntax and semantics are based on geometrical relations among these objects. In the work presented in this thesis, however, the geometrically interpretable symbols of G are depicted in a graphical medium without any regard to the possible *syntactic* relations of the resulting geometrical objects. Since a pictorial language formally endows geometrical objects with semantic values, an interesting field of investigation could be to explore how the inclusion of such a language can help to enable an intelligent graphical input, say, by relating the introduced graphical objects with their corresponding set of properties in language G.

Since language G is a very expressive language and not all its well formed expressions are translatable into L, another interesting area for future research is the extension of the natural language grammar to support more structures of the logical language G. For instance, the inclusion of the conjunctions, *and* and *or*, in the grammar rules, not only allows to translate into L the logical conjunctions of G, but gives rise to many interesting problems such as the resolution of anaphoric references between sentences, the multimodal semantic evaluation of more than one sentence, etc.

Although the selected domain of application did not require any special treatment of emergent objects, the theory and algorithms developed in Section 3.4 (p. 104) enabled their inclusion and handling in multimodal environments. The usefulness of this concept in different domains of application is an area that requires further investigation.

From the implementation point of view, the input of the prototype system developed in this thesis is rather primitive. It allows the introduction of a defined number of commands which are almost directly translated into the communicative goals that will bring about the required explanation. The system also allows, as a matter of test, the direct introduction of the communicative goals. Although a direct manipulation of the graphical part is provided, a more intelligent input method would make the system more interesting from the user's point of view. Two approaches seem feasible to improve the input interface of the system: a natural language interpreter, as proposed in the architecture and based on the theoretical ground given in Chapter 3 (p. 100), and a direct manipulation interface, like the one proposed by Moore (1995) for her explanation generation system. In both cases the possibility of deictic references to the graphical objects should be provided.

Appendix A

THE PATR FORMALISM

The PATR formalism for the declarative representation of grammars was an important work developed in the 1980s [Shieber 1986]. In this formalism, a grammar consists of a set of rules and constraints that specify how valid words can be combined. This appendix is devoted to the description of this current commonly used formalism for the specification of grammars (see the book by Gazdar & Mellish (1989) for a more detailed description).

A.1 Grammars and the PATR Formalism

A language can be thought of as a set of symbol's sequences whose membership is precisely specifiable by rules. Natural languages are by nature infinite, therefore it is impossible to elaborate a list containing all their members. This introduces the necessity of using grammars to describe natural languages.

A grammar is a formal system that defines the membership of the infinite set of linguistic expressions and assign a structure to each member of this set. In a modern sense of natural language processing (NLP), grammars contain only information about which combinations of objects are permitted and what the properties of the resultant objects are. They do not contain extraneous procedural information about how to put the objects together. The structure of each legal string is provided implicitly in the description without the necessity of explicit structure-building annotations.

During the last twenty years, several formalisms have been developed to express natural language grammars. Context-free phrase structure grammars are one class of such formalisms. Within this class, the PATR formalism [Shieber 1986] is relatively widely used in the NLP community, and many other grammatical formalisms can be readily translated into it.

A.1.1 Rules

A grammar is composed by rules that show how the parts of the speech or syntactic categories can be combined. A syntactic category can be realised simply as a word, or it can be realised, according to the grammar rules, as a compound of several other syntactic categories. In the PATR formalism a rule tells what a particular syntactic category (the *left-hand side* — LHS) can be composed of (the *right-hand side* — RHS). As an example, consider the next rule:

Rule {1 – simple sentence formation}
  S $\rightarrow$ NP VP.

The first row is only informative, it says that the following row is a rule and it also includes a comment. The second row is the actual rule, which is divided by the arrow in the RHS, consisting of the symbol S, and the LHS, consisting of the symbols NP and VP. The three symbols S, NP, and VP stand for the syntactic categories *sentence, noun phrase* and *verb phrase* respectively, and the rule states that a sentence can consist of a noun phrase followed by a verb phrase. A rule like this is known as a *phrase structure rule* since it reveals somehow the structure of a phrase.

A phrase that satisfies the previous rule can be "Peter swims". However, this conclusion can only be reached if some more rules are added to the grammar:

Rule {2 – intransitive verb}
  VP $\rightarrow$ V.
Rule {(a) – a verb}
  V $\rightarrow$ "swims".
Rule {(b) – a name}
  NP $\rightarrow$ "Peter".

The first rule says that a verb phrase can simply consist of an intransitive verb. The remaining two rules are quite singular in the sense that both have as RHS a word instead of having a sequence of syntactic categories. This means that the syntactic category in their LHS can consist of the word in their RHS. Applying all these rules it can be concluded that the analysed phrase is a sentence, and its syntactic structure arises implicitly from the rules. It can be depicted as a tree showing how the syntactic categories are composed of (Fig. A.1). The tree's root is an S and it has two children: NP and VP. This subtree represents the first rule, and each two-level subtree represents a grammar rule. The leaves of the tree correspond to the words of the phrase and they represent the application of the two final rules. These rules were used to introduce lexical items,

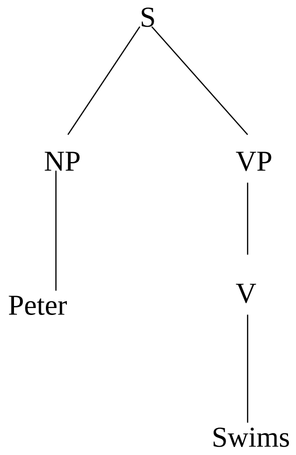


Figure A.1: A phrase structure tree

namely, the words “swims" and “Peter". Although phrase structure rules can be used in this way, computational linguists have conveniently separated lexical material in a kind of dictionary: the lexicon.

A.1.2 The lexicon

A lexicon can be viewed as a list of words that associates each word with its syntactic properties. The equivalent entries in the lexicon for the two final rules in the previous example might look as follows:

Word swims:
<cat> = V.
Word Peter:
<cat> = NP.

These lexical entries only tell us that the syntactic categories of the words *swims* and *Peter* are V and NP, respectively. In a less simple grammar a lexical entry can carry some more information as is shown below:

Word swims:
<cat> = V
<tense> = pres
<per> = 3
<num> = sing.

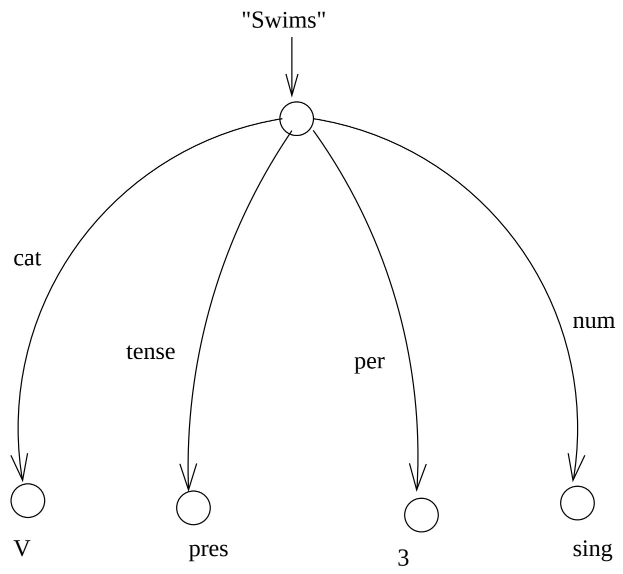


Figure A.2: A DAG for the word *swims*

This says that the word *swims* is a verb in present tense for the third person singular. Each row in the description of the word represents a syntactic characteristic of it. These syntactic characteristics are called *features*. A feature is typically an attribute value pair, with the attribute name at the left of the "=" symbol and its value at the right. A way of depicting a bundle of such features is as a *directed acyclic graph* (DAG). Such a bundle represents a syntactic category. The DAG corresponding to the word *swims* is shown in Fig. A.2.

In a grammar intended to represent a significant fragment of a natural language, the lexicon will contain more complex information for each lexical entry, which could include pure syntactic information, morphological information and even semantic information. As an example consider a fragment of a lexical entry for the transitive verb *to give*:

```
Lexeme give:
    <syn cat> = V
    <syn arg0 cat> = NP
    <syn arg0 case> = nom
    <syn arg1 cat> = NP
    <syn arg1 case> = acc
    <mor form5> = gave
    <mor form6> = given.
```

This description is quite different from the lexical entry presented before. First, note that the entry has as a heading the label Lexeme instead of the usual

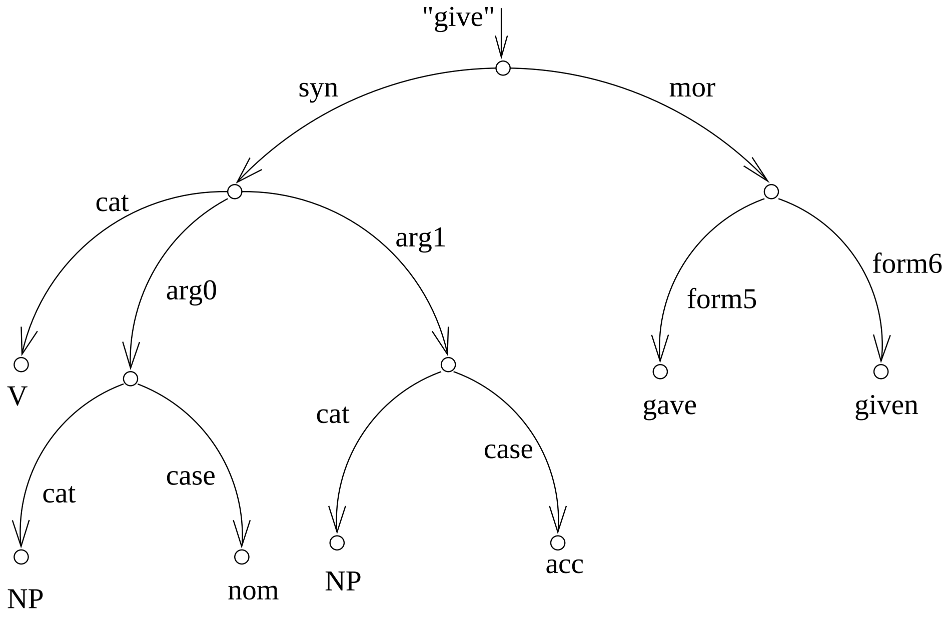


Figure A.3: A DAG for a complex lexical entry

Word. This is because the final instance of the word is to be assembled using the morphological information contained in the description. Second, instead of having simple attribute names, there is a sequence of names identifying each attribute. As before, each name is associated with one DAG arc (see Fig. A.3), but now an entire name sequence stands for a path in the DAG. So, for instance, the row <syn arg0 case> = nom specifies the path that goes from the root to the ending node labeled with nom. Finally, note that five of the name sequences begin with the name syn, and two of them begin with the name mor. Instead of having seven arcs, one for each name sequence, there are only two, one for each common sequence starting name, one labeled with syn and the other with mor. This means that the lexeme has two main features and their respective values are bundles of "lower level" features. Any feature whose value is not an atom is known as a *category valued feature*. In the example syn, mor, arg0 and arg1 are category valued features, and cat, case, form5 and form6 are atom valued features.

Lexical descriptions, as the previous, do not only contain syntactic information about the word itself but also about the context in which it can appear within a phrase. This is the purpose of the syntactic features arg0 and arg1. They say that the transitive verb *to give* should combine with two noun phrases: a subject and a direct object.

Lexical entries can even be more complex; however this is enough to give an idea of the information that can be represented in this way. How the information represented in the lexicon is to be used, is an issue that will be discussed afterwards.

A.2 Feature-based Grammars

The rules in the previous section describe a very simple grammar. Now expand it a little bit adding another rule to accept transitive verbs:

Rule {3 – transitive verb}
VP $\rightarrow$ V NP.

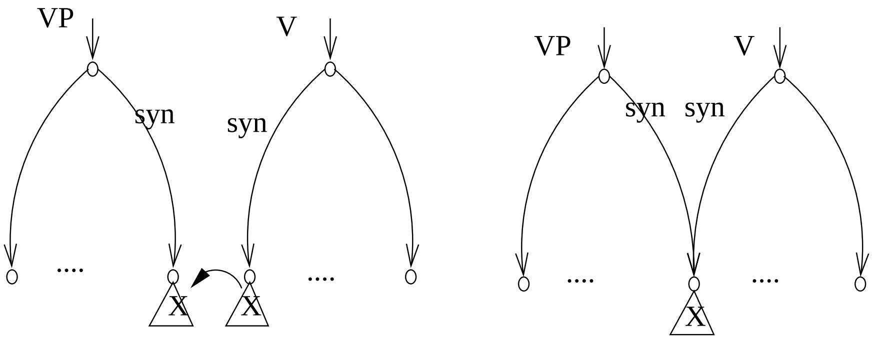
Rules 1 and 3 could be realised as the single rule: S $\rightarrow$ NP V NP. This rule says that a sentence can consist of a noun phrase followed by a verb and a noun phrase. With the exception of the syntactical category V, it does not say anything about the class of verb involved in the sentence. Then it is not a surprise that this grammar accepts sentences like: "*John died Mary". This introduces the necessity of a new mechanism to manage this difficulty: the employment of syntactic features within the rules.

A.2.1 Rule constraints

A constraint is a condition, specified in the body of the rule, that must be satisfied when the rule is applied. Since a rule is a constraint itself, it can be rewritten using syntactic features expressions. So the first rule can be expanded into its full form as follows:

Rule {1a – simple sentence formation}
X0 $\rightarrow$ X1 X2
<X0 cat> = S
<X1 cat> = NP
<X2 cat> = VP.

This says that X0 can consist of X1 followed by X2 provided that the value of X0's cat feature is S, the value of X1's cat feature is NP, and the value of X2's cat feature is VP. In writing rules, for the sake of conciseness the no-expanded form is preferred to this. However the previous example illustrates the use of syntactic features as constraints. This same approach can be applied in the specification of transitive verb's role within a sentence, by modifying rules 1 and 3:



(a) a copy of a feature (b) feature sharing

Figure A.4: Feature sharing

Rule {1b – simple sentence formation}
  S → X VP
    <X cat> = <VP syn arg0 cat>.
Rule {3a – transitive verb }
  VP → V X
    <VP syn> = <V syn>
    <V syn arg1 cat> = <X cat>.

The first constraint in rule 3a requires that the value of the syn feature on the VP and that on the V be identical. Since so far no syn feature for VP has been defined, this constraint is equivalent to having VP inheriting all feature information contained in syn feature from V. This explains the first constraint in rule 1b. Because VP has taken all information under syn from V, assuming a lexical entry for a transitive verb as the one depicted in Fig. A.3, the expression <VP syn arg0 cat> is NP. So the constraint states that <X cat> must be equal to NP; this is the subject of the sentence. With a similar analysis it can be concluded that the last constraint in rule 3a states that <X cat> must be equal to NP; this is the direct object of the sentence. A lexical entry for an intransitive verb, like *die*, will lack of the arg1 feature, so it would not admit a direct object.

A.2.2 Unification

In the previous discussion, nothing was said about how the inheritance from V to VP was made, nor a word about how comparisons between both sides of a '=' sign have to be done. These important operations depend upon notions of *subsumption* and *unification*. To explain these notions it is necessary to clarify the idea of *feature sharing*.

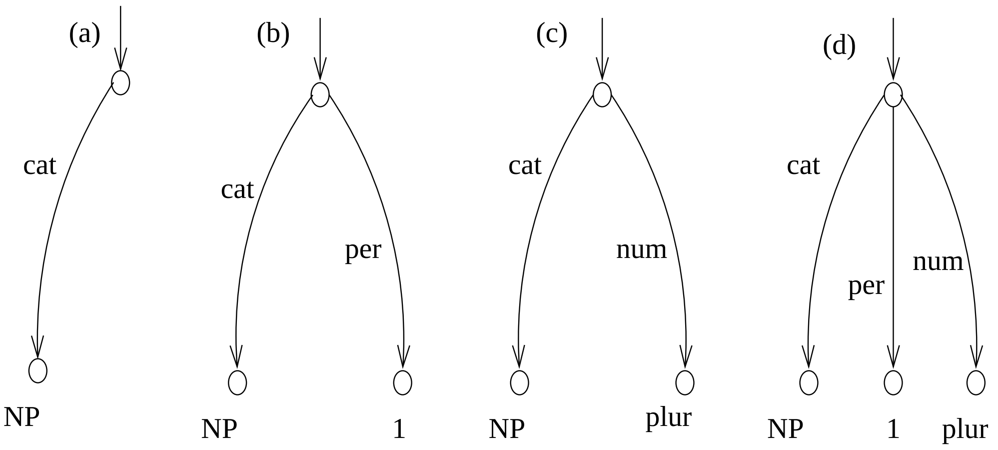


Figure A.5: Subsumption and unification

An expression like <VP syn> = <V syn> states that VP inherits the syn feature from V. This does not mean that a whole copy of the bundle of features represented by <V syn> is made; instead, the value of the syn feature is *shared* by both categories, as is shown in Fig. A.4. Assuming this mechanism, subsumption can be defined as follows:

A category **A** subsumes a category **B** if and only if:

1. Every atom valued feature specification in **A** is in **B**.

2. For every two feature values that share anywhere in **A**, the corresponding values share in **B**.

3. For every category valued specification in **A**, there is a value for that feature in **B**, and the value of the feature in **A** subsumes the value in **B**.

If **A** subsumes **B**, then **B** may be referred as an *extension* of **A** or it may be said that **B** *extends* **A**.

The unification is an operation very similar to the union of sets, and can be defined as follows:

The unification of two categories is the smallest category that extends both of them, if such a category exists, otherwise the unification is undefined.

To clarify these concepts, see Fig. A.5. The category labeled with (a) subsumes categories (b) and (c). The category (b) does not subsume the category (c), nor (c) subsumes (b), but both (b) and (c) subsume the category labeled with (d). Finally, the category (d) is the unification of (b) and (c), as well as it is the unification of the three (a), (b) and (c).

The two operations just described, and specially the unification, are of a key importance, since they are the basic conceptual tool for language recognition and generation based on grammar rules. There are different syntactic analysis strategies – e.g., top-down parsing or bottom-up parsing or any combination of them –, these, and the use of unification in parsing, can be consulted elsewhere [Gazdar & Mellish 1989, Santana 1994].

A.2.3 Unification and rules

To illustrate how the unification operation is used with grammar rules, the following rule, which embodies some agreements between sentence components, is introduced:

Rule {1c – simple sentence formation}
  S → NP VP
    <NP per> = <VP per>
    <NP num> = <VP num>
    <S subj> = <NP>.

This rule is a shorthand of the more complete rule given below:

Rule {1d – simple sentence formation}
  X → X1 X2
    <X cat> = S
    <X1 cat> = NP
    <X2 cat> = VP
    <X1 per> = <X2 per>
    <X1 num> = <X2 num>
    <X subj> = <X1>.

As said above, the equal sign, ‘=’, in this rule stands for a unification operation, and when that operation is performed, the result is illustrated in Fig. A.6.

A.3 The PATR Formalism in Natural Language Generation

Having encoded the semantics of the natural language (or at least some sort of logical representation) in this formalism, it is easy to adapt it for natural language generation. In fact, this process does not differ very much from that of parsing, and similar strategies can be applied. The main difference between

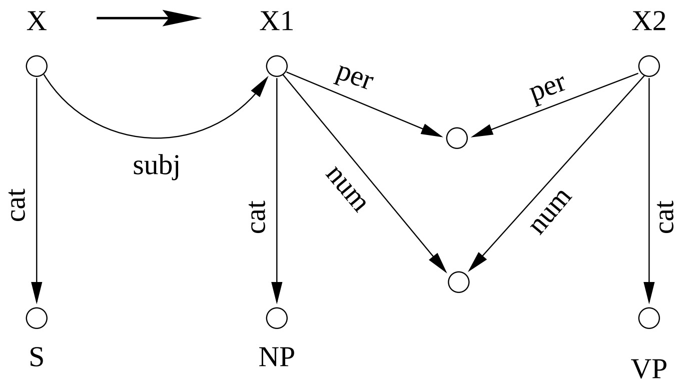


Figure A.6: Minimal DAGs satisfying a rule

parsing and generation is that instead of initiating the unification process from the <cat> feature, in the case of generation, it starts from the feature corresponding to the logical or semantic representation.

In the present study, the feature <lform> will play the role of the semantic representation.

While the parsing process faces problems like ambiguity, in which a single input sentence can have several semantic values, the generation process faces the problem that, given a single logical form, there can be several surface texts to realise it. In this case, pragmatical factors would contribute to decide which surface text better realises the given logical form.

A.3.1 From terms to DAGs

To facilitate the input to the generation process, it will be given as terms. Since the PATR formalism relies on a DAG representation, a procedure to translate from terms to DAGs is described here.

Let *abc(x, y, z, ...)* be a term whose functor is *abc* and whose arguments are in order *x, y, z, ...*. This term is translated into the DAG shown in Fig. A.7. In the PATR notation for this DAG is described as:

<funct> = *abc*
<arg1> = *x*
<arg2> = *y*
<arg3> = *z*.

With this convention it is possible to translate more complex terms such as *abc(def(x, y), z, t)*. This term is translated into the DAG whose graphical representation is shown in Fig. A.8, and its corresponding PATR notation is shown below:

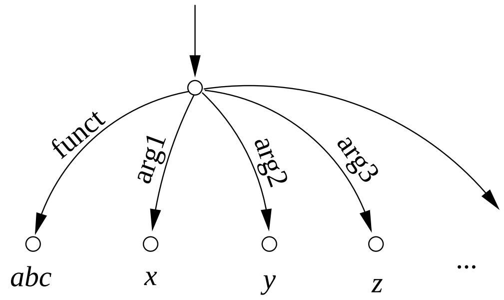


Figure A.7: A DAG corresponding to the term *abc(x, y, z, ...)*

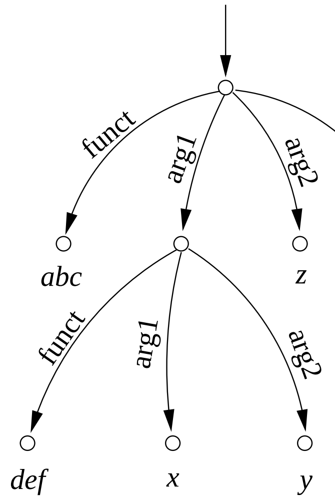


Figure A.8: A DAG corresponding to the term *abc(def(x, y), z, t)*

<funct> = *abc*
<arg1 funct> = *def*
<arg1 arg1> = *x*
<arg1 arg2> = *y*
<arg2> = *z*
<arg3> = *t*.

To cope with the sort of terms utilised in this work to describe the semantics within the grammar, the correspondences between terms and DAGs shown in Table A.1 are introduced.

For terms involving lambda or quantified expressions the feature (edge) corresponding to the variable in the DAG, i.e., <arg1>, must be unified with the features (i.e., the edges) corresponding to such variable in the <expr> subDAG. Several examples of this case are given in Fig. A.9. In this figure some edges have two labels: one corresponds to the name of the edge, and the other, which

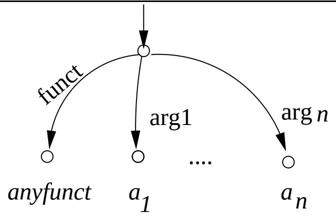
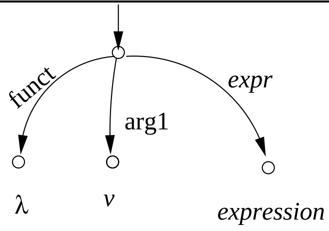
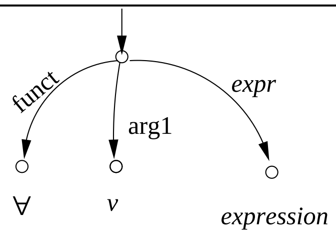
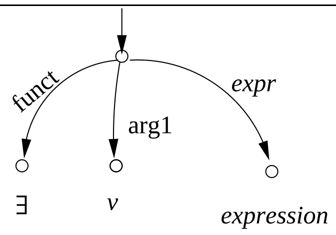
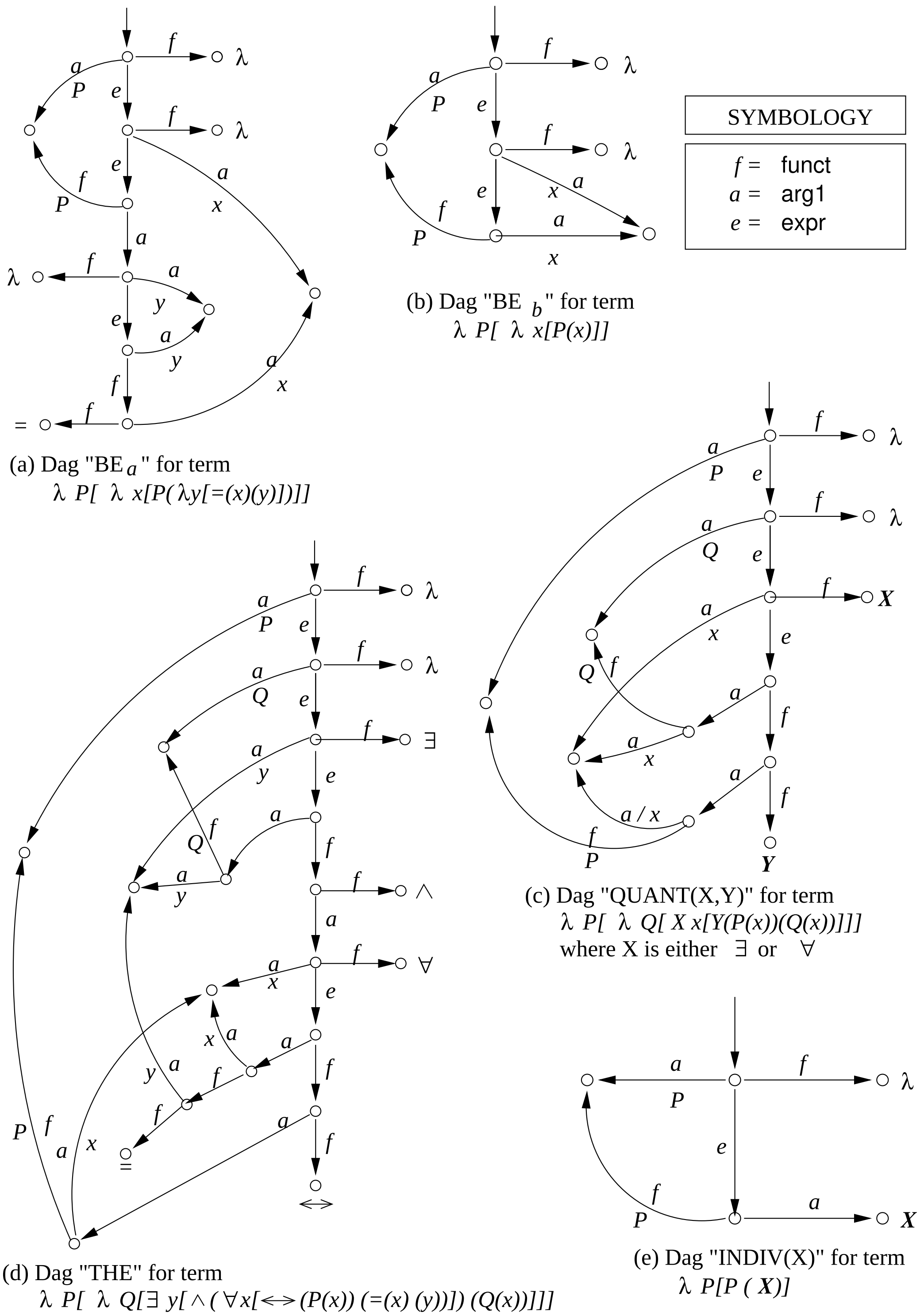
Term	PATR notation	DAG
$anyfunct(a_1, \ldots a_n)$	<funct> = $anyfunct$ <arg1> = a_1 $\vdots$ <arg$n$> = a_n.	
$\lambda v[expression]$	<funct> = λ <arg1> = v <expr> = $expression$.	
$\forall v[expression]$	<funct> = $\forall$ <arg1> = v <expr> = $expression$.	
$\exists v[expression]$	<funct> = $\exists$ <arg1> = v <expr> = $expression$.	

Table A.1: Terms and their corresponding DAGs



(a) Dag "BE$_a$" for term
$\lambda\, P[\ \lambda\, x[P(\lambda y[=(x)(y)])]]$

(b) Dag "BE$_b$" for term
$\lambda\, P[\ \lambda\, x[P(x)]]$

(c) Dag "QUANT(X,Y)" for term
$\lambda\, P[\ \lambda\, Q[\ X\, x[Y(P(x))(Q(x))]]]$
where X is either $\exists$ or $\forall$

(d) Dag "THE" for term
$\lambda\, P[\ \lambda\, Q[\exists\, y[\wedge(\forall x[\leftrightarrow (P(x))\ (=(x)\ (y))])\ (Q(x))]]]$

(e) Dag "INDIV(X)" for term
$\lambda\, P[P\ (\boldsymbol{X})]$

Figure A.9: DAGs corresponding to lambda and quantified expressions corresponds to the name of a variable in the original term, is written down just for the sake of clarity.

A.3.2 A simple grammar

The grammar described here serves the purpose of producing most of the natural language sentences required in this thesis. The macro inherit is used to to pass some information from the right hand side (RHS) of a rule to the left hand side (LHS). The macros pp_mac and pp1_macro are used to build up some parts of the DAGs corresponding to prepositional phrases. The first nine rules correspond to the rules of language L discussed on Chapter 3. The remaining rules are used to introduce negations and modal verbs, which are required only in the natural language generation phase.

The notation <... feature^n ...> is used to mean <... feature feature ... feature ...>, where feature appears n times in sequence.

```
Macro inherit(X,Y):
    <X agr> = <Y agr>
    <X neg> = <Y neg>
    <X lform> = <Y lform>
    <X modfs> = <Y modfs>
    <X v> = <Y v>.
Macro pp_mac(X):
    <X lform funct> = λ
    <X lform expr funct> = λ
    <X lform expr^2 funct arg1> = <T lform>
    <X lform expr^2 arg1> = <X lform expr arg1>.
Macro pp1_macro:
    pp_mac(PP')
    <PP' lform expr^2 funct^2> = <PP' lform arg1>.
Rule {  A.1. Sentences }
    t → T IV
        <T agr> = <IV agr>
        <t modf> = <IV modf>
        <t lform funct> = <T lform>
        <t lform arg1> = <IV lform>.
Rule {  A.2. Transitive verb phrases }
    IV → TV" T
        <IV modf> = <TV" modf>
        <IV agr> = <TV" agr>
        <IV neg> = <TV" neg>
        <IV v> = <TV" v>
```

< IV lform funct > = < TV” lform >
< IV lform arg1 > = < T lform >.
Rule { A.3. Attributive verb phrases }
IV → IV/ADJ” ADJ
< IV modf > = < IV/ADJ” modf >
< IV agr > = < IV/ADJ” agr >
< IV neg > = < IV/ADJ” neg >
< IV v > = < IV/ADJ” v >
< IV lform funct > = < IV/ADJ” lform >
< IV lform arg1 > = < ADJ lform >.
Rule { A.4. Terms }
T → T/CN CN
< T agr > = < T/CN agr >
< T/CN num > = < CN num >
< T lform funct > = < T/CN lform >
< T lform arg1 > = < CN lform >.
Rule { A.5. Common nouns }
CN → CN_1 PP
< CN_1 cat > = CN
< CN lform funct > = < PP lform >
< CN lform arg1 > = < CN_1 lform >.
Rule { A.6. Compound adjectives }
ADJ → ADJ_1 PP’
< ADJ_1 cat > = ADJ
< ADJ lform funct > = < PP’ lform >
< ADJ lform arg1 > = < ADJ_1 xfunctor >.
Rule { A.7. ‘Between’ prepositional phrases }
PP → between T and T_1
< T_1 cat > = T
pp_mac(PP)
< PP lform $expr^2$ $funct^3$ > = $intersection\_between$
< PP lform $expr^2$ $funct^2$ arg1 > = < T_1 >
Rule { A.8. ‘With’ prepositional phrases }
PP’ → with T
pp1_macro.
Rule { A.9. ‘To’ prepositional phrases }
PP’ → to T
pp1_macro.
Macro modifier_mac(X,Y):
< X’ agr > = < X agr >
< X’ lform > = < X lform >
< X’ modf funct > = < X modf >

```
    <X' modf arg1> = <Y lform>.
Rule {  A.10. Modified verbs 1 }
    TV" → Aux TV'
        <TV' v> = aux
        modifier_mac(TV',Aux).
Rule {  A.11. Modified verbs 1' }
    TV" → TV'
        inherit(TV",TV').
Rule {  A.12. Modified verbs 2 }
    TV' → Neg TV
        <TV neg> = +
        modifier_mac(TV, Neg).
Rule {  A.13. Modified verbs 2' }
    TV' → TV
        inherit(TV',TV).
Rule {  A.14. Modified verbs 3 }
    IV/ADJ" → Aux IV/ADJ'
        <IV/ADJ' v> = aux
        modifier_mac(IV/ADJ',Aux).
Rule {  A.15. Modified verbs 3' }
    IV/ADJ" → IV/ADJ'
        inherit(IV/ADJ",IV/ADJ').
Rule {  A.16. Modified verbs 4 }
    IV/ADJ' → Neg IV/ADJ
        <IV/ADJ neg> = +
        modifier_mac(IV/ADJ, Neg).
Rule {  A.17. Modified verbs 4' }
    IV/ADJ' → IV/ADJ
        inherit(IV/ADJ',IV/ADJ).
```

A.3.2.1 Lexicon

For the sake of space, the lexicon is given in Tables A.2, A.3, A.4, A.5, A.6, A.7, A.8, and A.9. To write one of the entries of the previous tables in PATR notation is an easy task. Take, for instance, entry A.61 of Table A.7. This entry is written in PATR as follows:

```
Word lie-outside:
    <cat> = TV
    <lform> = outside
    <neg> = +
```

<v> = do
<inv> = -
<modf> = mdflist.

Note that there is no <agr> feature indicated in this transcription. In Table A.7 this is signaled by a dot in the corresponding cell. This means that, in the process of unification, the DAG corresponding to this entry can accept that feature with any value.

Some cells of the lexicon tables make reference to the macros BE_a, BE_b, INDIV(X), QUANT(X,Y), and THE, which correspond to the DAGs of the same names illustrated in Fig. A.9. To distinguish in the table the application of these macros from the normal entries, each application of a macro is preceded by the single quotation mark (‘). The definition of the macros is as follows:

Macro BE_a:
<funct> = λ
<expr funct> = λ
<expr^2 funct> = <arg1>
<expr^2 arg1 funct> = λ
<expr^2 arg1 expr arg1> = <expr^2 arg1 arg1>
<expr^2 arg1 expr funct^2> = ‘=’
<expr^2 arg1 expr funct arg1> = <expr arg1>.
Macro BE_b:
<funct> = λ
<expr funct> = λ
<expr^2 funct> = <arg1>
<expr^2 arg1> = <expr arg1>.
Macro INDIV(X):
<funct> = λ
<expr arg1> = X
<expr funct> = <arg1>.
Macro QUANT(X,Y):
<funct> = λ
<expr funct> = λ
<expr^2 funct> = X
<expr^3 funct^2> = Y
<expr^3 funct arg1 funct> = <arg1>
<expr^3 funct arg1^2> = <expr^2 arg1>
<expr^3 arg1 funct> = <expr arg1>
<expr^3 arg1^2> = <expr^2 arg1>.
Macro THE:
<funct> = λ

<expr funct> = λ
<expr^2 funct> = $\exists$
<expr^3 funct^2> = $\wedge$
<expr^3 funct arg1 funct> = $\forall$
<expr^3 funct arg1 expr funct^2> = $\leftrightarrow$
<expr^3 funct arg1 expr funct arg1 funct> = <arg1>
<expr^3 funct arg1 expr funct arg1^2> = <expr^3 funct arg1^2>
<expr^3 funct arg1 expr arg1 funct^2> = '='
<expr^3 funct arg1 expr arg1 funct arg1> = <expr^2 arg1>
<expr^3 funct arg1 expr arg1^2> = <expr^3 funct arg1^2>
<expr^3 arg1 funct> = <expr arg1>
<expr^3 arg1^2> = <expr^2 arg1>.

In order to interpret these macros in relation to the lexicon tables, it should be noted that the equal symbol (=) is utilised to mean a *unification* operation. Thus, for instance, the cell corresponding to the feature <lform> of row A.19 in Table A.2 is interpreted as follows.

<lform> = 'INDIV(l_i)

$$<\text{lform}> = \begin{cases} <\text{funct}> = \lambda \\ <\text{expr arg1}> = l_i \\ <\text{expr funct}> = <\text{arg1}> \end{cases}$$

and once the unification is applied:

<lform funct> = λ
<lform expr arg1> = l_i
<lform expr funct> = <lform arg1>

With this operation, the whole lexicon entry for the word bar_5, using row A.19 of Table A.2, is transcribed in PATR as follows:

Word bar_5:
<cat> = T
<lform funct> = λ
<lform expr arg1> = l_5
<lform expr funct> = <lform arg1>
<agr> = 3s.

A.3.3 Producing the surface text

The production of the natural language surface text requires a fully instantiated logical form that embodies the semantics of the desired text and the syntactic

No.	Word	<cat>	<lform>	<agr>
A.18.	$joint_i$	T	‘INDIV(d_i)	3s
A.19.	bar_i	T	‘INDIV(l_i)	3s
A.20.	$crossarm_i$	T	‘INDIV(p_i)	3s
A.21.	$dielectric\text{-}area_i$	T	‘INDIV(c_i)	3s
A.22.	*luminous-paint*	T	‘INDIV(*gluminous_paint*)	3s
A.23.	*anticorrosive-paint*	T	‘INDIV(*ganticorrosive_paint*)	3s
A.24.	*design-constraints*	T	*rule*	3p
A.25.	*satisfied-constraints*	T	*satisfied*	3p
A.26.	*unsatisfied-constraints*	T	*unsatisfied*	3p
A.27.	*here*	T	*ghere*	3s
A.28.	*there*	T	*gthere*	3s
A.29.	*it*	T	*git*	3s

Table A.2: Lexicon entries for terms

No.	Word	<cat>	<lform>	<num>
A.30.	*joint*	CN	*dot*	s
A.31.	*joint*	CN	*dot*	p
A.32.	*intersection*	CN	*gintersection*	s
A.33.	*intersections*	CN	*gintersection*	p
A.34.	*bar*	CN	*line*	s
A.35.	*bars*	CN	*line*	p
A.36.	*crossarm*	CN	*polygon*	s
A.37.	*crossarms*	CN	*polygon*	p
A.38.	*dielectric-area*	CN	*circle*	s
A.39.	*dielectric-areas*	CN	*circle*	p
A.40.	*paint*	CN	*gpaint*	s
A.41.	*paints*	CN	*gpaint*	p

Table A.3: Lexicon entries for common names

No.	Word	<cat>	<lform>	<agr>	<num>
A.42.	*a*	T/CN	‘QUANT($\exists$,$\wedge$)	3s	s
A.43.	*some*	T/CN	‘QUANT($\exists$,$\wedge$)	3p	p
A.44.	*the*	T/CN	‘THE	3s	s
A.45.	*every*	T/CN	‘QUANT($\forall$,$\rightarrow$)	3s	s
A.46.	*this*	T/CN	*gthis*	3s	s
A.47.	*these*	T/CN	*gthis*	3p	p

Table A.4: Lexicon entries for determiners

No.	Word	<cat>	<lform>	<agr>	<aux>
A.48.	*is*	Neg	*gnot*	3s	be
A.49.	*are*	Neg	*gnot*	3p	be
A.50.	*does-not*	Neg	*gnot*	3s	do
A.51.	*do-not*	Neg	*gnot*	3p	do
A.52.	*not*	Neg	*gnot*	.	aux

Table A.5: Lexicon entries for negations

No.	Word	<cat>	<lform>
A.53.	*must*	Aux	*gmust*
A.54.	*should*	Aux	*gshould*

Table A.6: Lexicon entries for auxiliary verbs

category of the desired text. Thus in this thesis, the input to the procedure for the production of the surface text will be given in any of the two forms:

X[*lform*]
X[*lform*, *vmodifiers*] (A.1)

where X is a syntactic category, typically *sentence* signaled with a t, *lform* is a term representing the logical form corresponding to the <lform> feature of the LHS of the grammar rules corresponding to the category X, and *vmodifiers* is a term representing the verb modifiers. In this formula, the category X receives the name of *target category*, and the logical form is designated as the *input logical form*. It should be noted that *lform* in this formula is an expression of language G (see Chapter 3) translatable to the natural language L.

The second form in expression (A.1) is used when it is desired to produce an expression whose verb is modified by auxiliary verbs or negations.

When the input logical form contains subscripted symbols, e.g., $outside_*$, $parallel\_to_*$, $painted\_with_*$, etc., (see Section 3.2.2.3 p. 87) which are not in the lexicon, it is converted into an equivalent logical form in terms of the unsubscripted version of the symbols, i.e., *outside*, *parallel_to*, *painted_with*, etc., by means of the following *inverse* of the meaning postulates, given in p. 88.

InvMP1. $\forall x \forall y[\delta_*(y)(x) \leftrightarrow \delta(\lambda P[P(y)])(x)]$, where $\delta \in$ {*inside, outside, perpendicular_to, parallel_to, painted_with*}.

InvMP2. $\forall x \forall u \forall v[(\delta_*(v)(u) = x) \leftrightarrow \delta(\lambda P_1[P_1(v)])(\lambda P_2[P_2(u)])(x)]$, where $\delta =$*intersection_between*.

No.	Word	<cat>	<lform>	<agr> <v> <neg>			<inv>	<modf>
A.55.	*lies-inside*	TV	*inside*	3s	-	-	-	mdflist
A.56.	*lie-inside*			3p	-	-	-	mdflist
A.57.	*lie-inside*			.	do	+	-	mdflist
A.58.	*lie-inside*			.	aux	.	-	mdflist
A.59.	*lies-outside*	TV	*outside*	3s	-	-	-	mdflist
A.60.	*lie-outside*			3p	-	-	-	mdflist
A.61.	*lie-outside*			.	do	+	-	mdflist
A.62.	*lie-outside*			.	aux	.	-	mdflist
A.63.	*lies-inside*	TV	*r_inside*	3s	-	-	+	mdflist
A.64.	*lie-inside*			3p	-	-	+	mdflist
A.65.	*lie-inside*			.	do	+	+	mdflist
A.66.	*lie-inside*			.	aux	.	+	mdflist
A.67.	*lies-outside*	TV	*r_outside*	3s	-	-	+	mdflist
A.68.	*lie-outside*			3p	-	-	+	mdflist
A.69.	*lie-outside*			.	do	+	+	mdflist
A.70.	*lie-outside*			.	aux	.	+	mdflist
A.71.	*is*	TV	'BE$_a$	3s	-	-	-	mdflist
A.72.	*are*			3p	-	-	-	mdflist
A.73.	*not*			.	be	+	-	mdflist
A.74.	*be*			.	aux	.	-	mdflist
A.75.	*is*	IV/ADJ	'BE$_b$	3s	-	-	.	mdflist
A.76.	*are*			3p	-	-	.	mdflist
A.77.	*not*			.	be	+	.	mdflist
A.78.	*be*			.	aux	.	.	mdflist

Table A.7: Lexicon entries for verbs

No.	Word	<cat>	<lform>	<xfunctor>
A.79.	*horizontal*	ADJ	*ghorizontal*	-
A.80.	*vertical*	ADJ	*gvertical*	-
A.81.	*painted*	ADJ	*gpainted*	*painted_with*
A.82.	*perpendicular*	ADJ	*gperpendicular*	*perpendicular_to*
A.83.	*parallel*	ADJ	*gparallel*	*parallel_to*

Table A.8: Adjectives

No.	Word	<cat>
A.84.	*I*	I
A.85.	*moved*	Moved
A.86.	*from*	From
A.87.	*between*	between
A.88.	*and*	and
A.89.	*with*	with
A.90.	*to*	to

Table A.9: Some syncategorematic words

A.3.3.1 Treatment of lambda expressions

Since the grammar developed here admits lambda expressions, the input logical form may contain solved or unsolved lambda expressions. For instance, the expressions

$\lambda x[horizontal(x)](l_5)$, and

$horizontal(l_5)$

are equivalent, and therefore any of them might be used as input to produce the same text.

The grammar rules do not have lambda expressions solved beforehand. Therefore, during the process of matching between grammar rules and the input logical form, each DAG, coming from the rules, and corresponding to an instantiated lambda expression, should be transformed to a new DAG representing the solved lambda expression. This process, illustrated in Fig. A.10, consists of the following steps:

1. The DAG is explored to verify that it is an instantiated lambda expression, i.e., to verify that the condition '<funct funct> = λ' holds (see Fig. A.10(a)).

2. The argument is passed to the the lambda function. That is, the unification '<funct arg1> = <arg1>' is performed (see Fig. A.10(b)).

3. The lambda function is determined. This is done by means of the unification operation '<funct expr> = <alt>' (see Fig. A.10(b)).

The last step introduces the feature <alt>, and its corresponding edge in Fig. A.10(b), which allows an empty transition during a unification operation. Although this feature, which is permitted only in the context of lambda expressions, introduces an apparent ambiguity, only serves the purpose of interpreting the whole DAG either as an unsolved lambda expression or as its corresponding solved expression, which consists of a functor '<alt funct>' and an argument '<alt arg1>'. This means that if the current logical form against which the DAG is

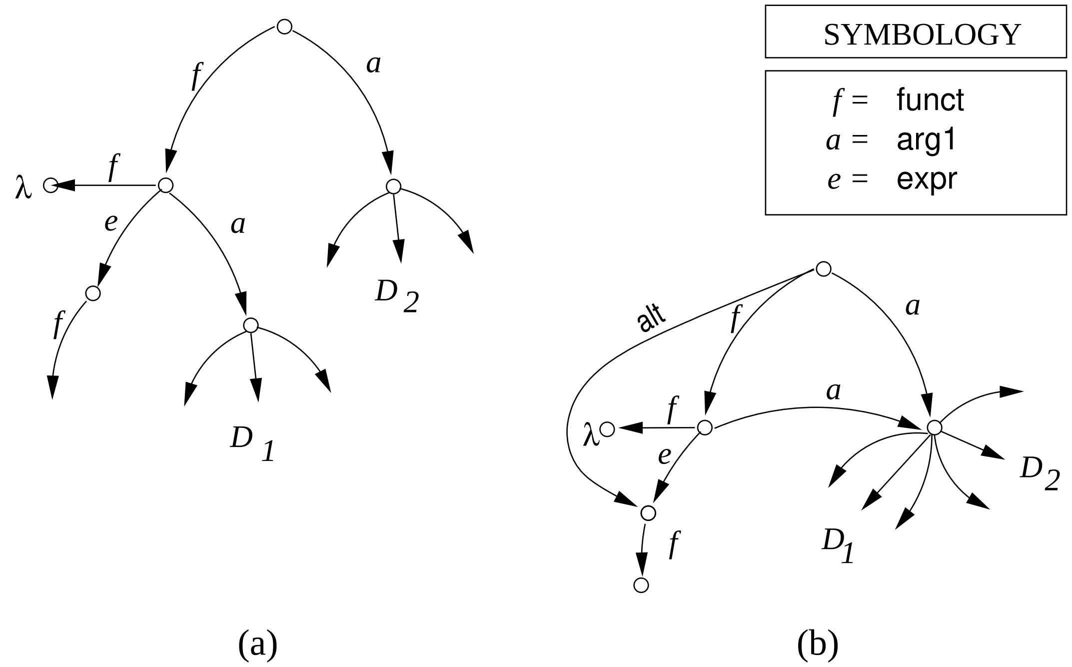


Figure A.10: Lambda resolution

being compared, does not unify with '<funct funct> = λ', the alternate possibility '<alt funct>' is tried.

A.3.3.2 Example

Consider, for instance, how to produce the text corresponding to the following expression:

$$\mathsf{t}[ghorizontal(l_5)] \tag{A.2}$$

this expression indicates the generation of a sentence whose <lform> feature matches the one corresponding to the term $ghorizontal(l_5)$.

The generation process starts with the symbols in the original formula given in expression (A.2), namely: $ghorizontal$, and l_5. The lexicon, is explored to locate the entries related to these symbols: entry A.79 in Table A.8 for $ghorizontal$, and entry A.19 in Table A.2 for l_5. These lexicon entries correspond to syntactic categories ADJ and T respectively. Therefore, the grammar rules whose RHS contain any of these categories are marked to be tried first. In this example the selected ones are rule A.3, which refers to category ADJ, and rules A.1 and A.2, which refer to category T[1]. The rules without their feature constraints are repeated here:

rule A.3: IV $\rightarrow$ IV/ADJ ADJ

[1]For clarity, in this explanation secondary rules A.10 to A.17 are not considered.

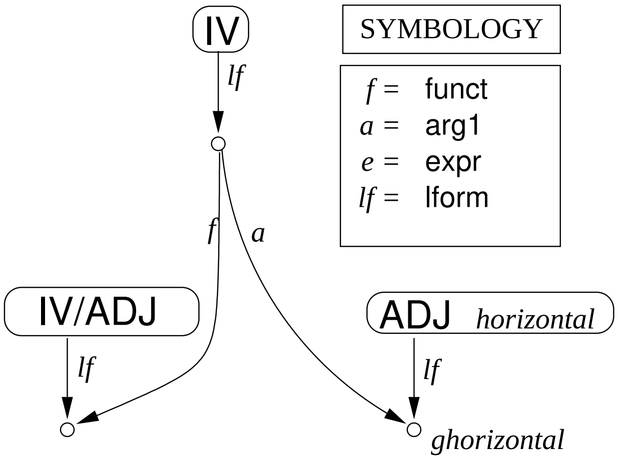


Figure A.11: Unification of a rule with a lexicon entry

rule A.1: t → T IV
rule A.2: IV → TV T

These rules are unified with the lexicon entries related to the symbols. Part of the resulting DAG corresponding to the unification of rule A.3 with lexicon entry A.79 of Table A.8 is shown in Fig. A.11. For this rule to be completely determined, the syntactic category IV/ADJ should have been unified with another rule with LHS of the same category, or with a lexical entry of the same category containing a symbol of the input logical form. However, there are no more input symbols left. Therefore, the lexicon is explored for entries of category IV/ADJ that are not associated to any particular symbol. The only cases are the lexicon entries corresponding to the different forms of the verb "to be" of category IV/ADJ (entries A.75 to A.78 of Table A.7). Although the four cases can be tried in the generation process, in Fig. A.12 it is shown only the case selected according to feature constraints coming from other rules that will be applied afterwards. Figure A.12(a) shows the unification of rule A.3 with lexicon entry A.75, and illustrates the operations described in Section A.3.3.1 to represent the solved lambda expression. Figure A.12(b) depicts the final DAG resulting from these operations.

This bottom-up process is carried on until a rule expanding the target category (t in this case) is reached. For the example presented here, a complete instantiation of rule A.1, which expands the target category, is illustrated in Fig. A.13.

Finally, the input logical form is compared with the resulting DAG of Fig. A.13(b). The DAG corresponding to the input logical form is given by the following PATR expression:

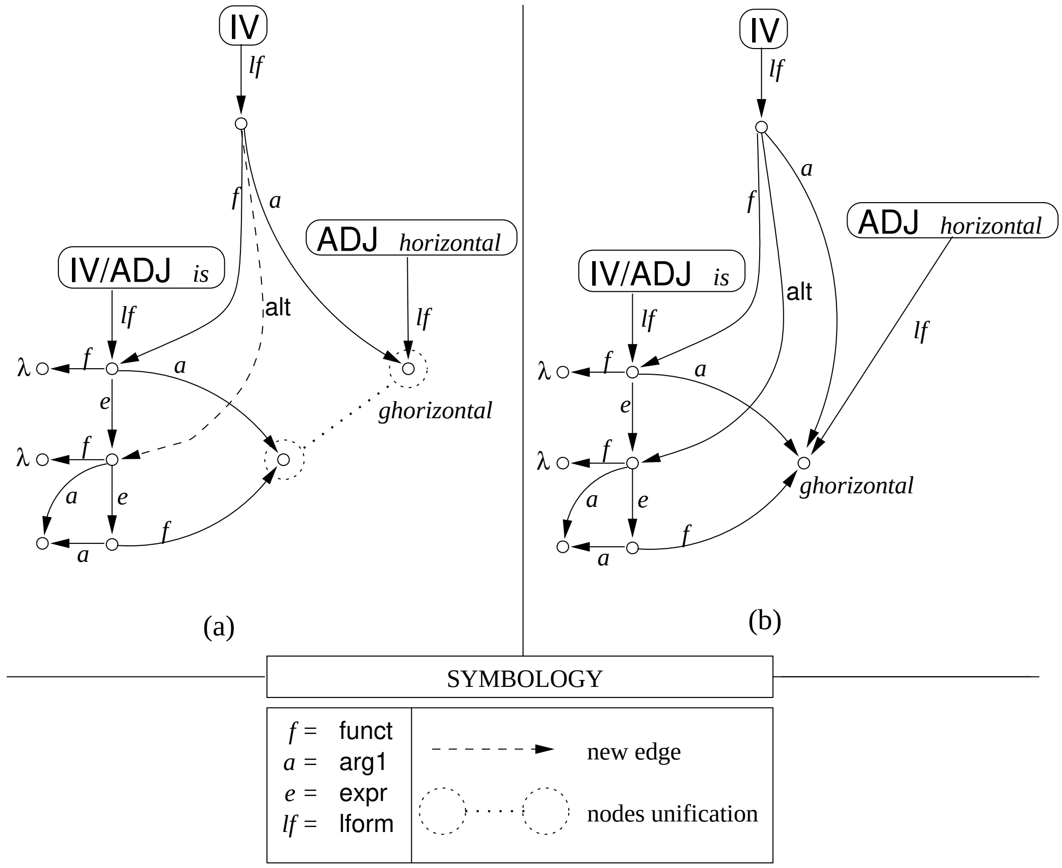


Figure A.12: Completing the instantiation of a rule

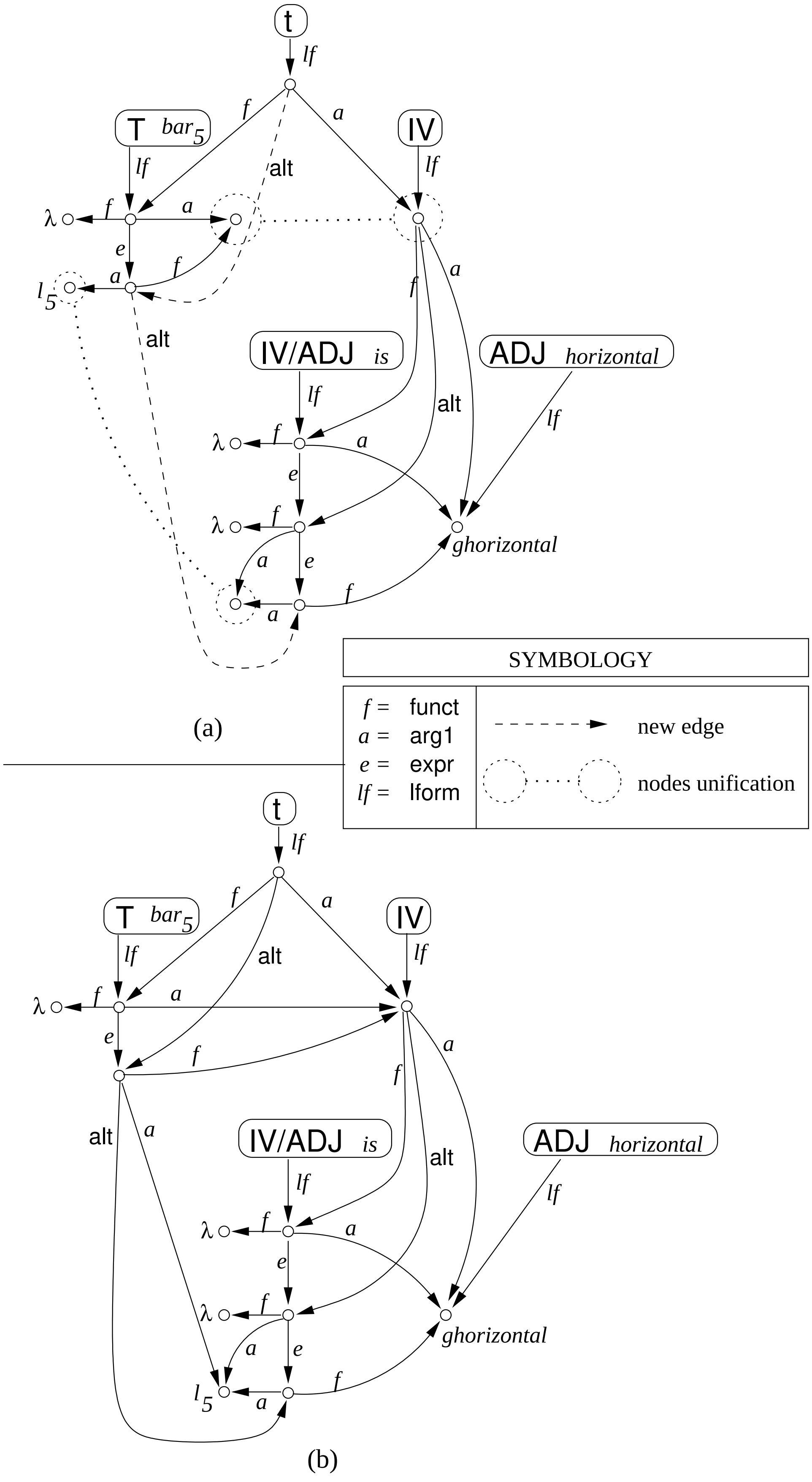


Figure A.13: Generation process for expression t[$ghorizontal(l_5)$]

<inplf funct> = *ghorizontal*
<inplf arg1> = l_5

Since, <t lform alt alt> = <inplf>, it can be concluded that the DAG in Fig. A.13(b) corresponds to the input logical form given in expression (A.2), and therefore, the generated text will be:

bar$_5$ *is horizontal*

With the second form of expression (A.1), additional constraints can be introduced to produce a natural language expression whose verb is modified by auxiliary verbs and/or negations. For instance, the expression t[*ghorizontal*(l_5), mdflist(*gnot*)(*gmust*)] would have generated the sentence:

bar$_5$ *must not be horizontal*

In this case, not only the feature <lform> of the sentence is checked for unification with the DAG corresponding to the input logical form, but its feature <modf> is also checked for unification with the DAG corresponding to the input term *vmodifiers* given in the input expression (*vmodifiers* is 'mdflist(*gnot*)(*gmust*)' in this case).

A.3.4 Special cases

In order that the system can generate a special case of a kind of sentence in the past tense and with a prepositional phrase, the following simplified adaptation was made.

Two new rules were added:

```
Rule {  A.91. Sentences }
   t → I Moved T PP”
      <t lform funct funct > = <PP” lform>
      <t lform funct arg1> = past-move
      <t lform arg1> = system
      <t lform arg2> = <T lform>.
Rule {  A.92. Prepositional Phrase }
   PP” → From T1 to T2
      <PP” lform funct> = from-to
      <PP” lform arg1> = <T1 lform>
      <PP” lform arg2> = <T2 lform>.
```

The words "I", "moved", "from", and "to" are treated syncategorematically, as shown in Table A.9. This simplified addition allows to produce sentences like

"I moved this bar from there to here"

from logical forms like

t[*(from-to(ghere, gthere)(past-move))(system, gthis(line))*]

To cope with some cases in cases in which the object of a sentence is quantified prior to the subject the following two rules are introduced:

Rule { A.93. Sentences }
t $\rightarrow$ ThereIsSome CN SuchThat T TV" T_1
$<T_1$ cat$>$ = T
$<T_1$ lform$>$ = *git*
$<$TV" inv$>$ = +
$<$CN num$>$ = s
$<$T agr$>$ = $<$TV" agr$>$
$<$t lform funct$^2>$ = 'QUANT($\exists, \wedge$)
$<$t lform funct arg1$>$ = $<$CN lform$>$
$<$t lform arg1 funct$>$ = $<$TV" lform$>$
$<$t lform arg1$^2>$ = $<T_1$ lform$>$.
Rule { A.94. Sentences }
t $\rightarrow$ ForEach CN T TV" T_1
$<T_1$ cat$>$ = T
$<T_1$ lform$>$ = *git*
$<$TV" inv$>$ = +
$<$CN num$>$ = s
$<$T agr$>$ = $<$TV" agr$>$
$<$t lform funct$^2>$ = 'QUANT($\forall, \rightarrow$)
$<$t lform funct arg1$>$ = $<$CN lform$>$
$<$t lform arg1 funct$>$ = $<$TV" lform$>$
$<$t lform arg1$^2>$ = $<T_1$ lform$>$.

Bibliography

Allen, J. (1983), Recognizing intentions from natural language utterances, *in* M. Brady & R. C. Berwick, eds, 'Computational models of discourse', Artificial Intelligence, The MIT Press, Cambridge, MA, U.S.A., pp. 107–166.

Allen, J. (1995), *Natural language understanding*, 2nd. edn, The Benjamin/Cummings Publishing Company, Inc., 390 Bridge Parkway, Redwood City, CA 94065, U.S.A.

André, E. & Rist, T. (1993), The design of illustrated documents as a planning task, *in* Maybury (1993*b*).

André, E. & Rist, T. (1994), Referring to world objects with text and pictures, *in* 'Proc. International Conference on Computational Linguistics. COLING', Association of Computational Linguistics. ACL, Kyoto, Japan, pp. 530–534.

André, E., Finkler, W., Graf, W., Rist, T., Shauder, A. & Wahlster, W. (1993), WIP: The automatic synthesis of multimodal presentations, *in* Maybury (1993*b*).

Appelt, D. & Kronfeld, A. (1987), A computational model of referring, *in* 'Proc. of the 10th. IJCAI', International Joint Conference on Artificial Intelligence, Milan, Italy, pp. 640–647.

Aristotle (1954), *Rhetoric*, number 246 *in* 'The Modern library of the world's best books', 1st modern library edn, Modern Library, New York, NY, U.S.A. Rhetoric; translated by W. Rhys Roberts. Poetics; translated by Ingram Bywater. Introd. by Friedrich Solmsen.

Austin, J. L. (1962), *How to do things with words*, Oxford University Press, New York, NY, U.S.A.

Barwise, J. & Etchemendy, J. (1989), *The language of first-order logic*, Vol. 34 of *CSLI Lecture Notes*, University of Chicago Press, Chicago, IL, U.S.A.

Blattner, M. M. & Glinert, E. P. (1996), 'Multimodal intergration', *IEEE Multimedia* **3**(4), 14–24.

Boolos, G. S. & Jeffrey, R. C. (1989), *Computability and logic*, Open University Set Book, 3rd. edn, Cambridge University Press.

Borning, A. (1981), 'The programming language aspects of ThingLab, a constraint-oriented simulation laboratory', *ACM Transactions on Programming Languages and Systems* **3**(4), 353–387.

Borning, A. (1986), 'Graphically defining new building blocks in ThingLab', *Human Computer Interaction* **2**, 269–295.

Brachman, R. J. (1985), On the epistemological status of semantic networks, *in* R. J. Brachman & H. J. Levesque, eds, 'Readings in knowledge representation', Morgan Kaufmann Publishers, Inc., San Mateo, CA, U.S.A., pp. 191–215.

Bratko, I. (1990), *Prolog programming for artificial intelligence*, International Computing Science Series, 2nd edn, Addison Wesley Publishing Company, Reading MA, U.S.A.

Brown, D. C. & Birmingham, W. P. (1997), 'Guest editor's introduction: understanding the nature of design', *IEEE Expert* **12**(2), 14–16.

Brown, D. C. & Chandrasekaran, B. (1986), 'Knowledge and control for a mechanical design expert system', *Computer* **19**(7), 92–100.

Brown, D. C., Kwasny, S. C., Chandrasekaran, B. & Sondheimer, N. K. (1979), 'An experimental graphics system with natural-language input', *Computer and Graphics* **4**, 13–22.

Buchanan, B. G. & Feigenbaum, E. A. (1981), Dendral and Metadendral: their applications dimension, *in* B. L. Webber & N. J. Nilsson, eds, 'Readings in artificial intelligence', Tioga, Palo Alto, CA, U.S.A., pp. 313–322.

Buchanan, B. G. & Shortliffe, E. H., eds (1984), *Rule-based expert systems: The MYCIN experiments of the Stanford heuristic programming project*, Addison-Wesley, Reading, MA, U.S.A.

Bunt, H., Ahn, R., Kievit, L., Piwek, P., Verlinden, M., Beun, R. J. & Borghuis, T. (1998), Multimodal cooperation with the denk system, *in* 'Proc. of the Second International Conference on Cooperative Multimodal Communication (CMC98)', The Computational Linguistics and Artificial Intelligence Group at Tilburg University,, Tilburg, The Netherlands.

Carberry, S. (1990), *Plan recognition in natural language dialogue*, The MIT Press, Cambrige, MA, U.S.A.

Carbonell, J. R. (1970), Mixed-initiative man-computer instructional dialogues, Technical Report BBN No. 1971, Bolt, Beranek & Newman, Cambridge, MA, U.S.A.

Chandrasekaran, B. (1986), 'Generic tasks in knowledge-based reasoning: high-level building blocks for expert system design', *IEEE Expert* **1**(3), 23–30.

Chandrasekaran, B. (1998), Diagrammatic representations and how they fit into multimodal perceptual representations and reasoning, *in* 'Proceedings of the Workshop on Interpretation and Generation in Intelligent Multimodal Systems and Graphical Reasoning in Expert Systems, in conjunction with the 4th World Congress on Expert Systems', International Society for Intelligent Systems, Mexican Society for Artificial Intelligence, and Instituto Tecnológico y de Estudios Superiores de Monterrey (ITESM), Mexico City, Mexico, pp. 1–5.

Chandrasekaran, B. & Swartout, W. (1991), 'Explanations in knowledge systems: the role of explicit representation of design knowledge', *IEEE Expert* **6**(3), 47–49.

Chandrasekaran, B., Tanner, M. C. & Josephson, J. R. (1989), 'Explaining control strategies in problem solving', *IEEE Expert* **4**(1), 9–24.

Codd, E. F., Arnold, R. S., Cadiou, J. M., Chang, C. L. & Roussopoulos, N. (1978), Rendezvous Version 1: an experimental English-language query formulation system for casual users of relational data bases, Technical Report RJ2144(29407), IBM Research Laboratory, San Jose, CA, U.S.A.

Cohen, J. (1990), 'Constraint logic programming languages', *Communications of the ACM* **33**(7), 52–68.

Cohen, P. R. (1978), On knowing what to say: planning speech acts, PhD thesis, Departament of Computer Science, University of Toronto, Canada. Published by University of Toronto as Technical Report 118.

Cohen, P. R. & Perrault, C. R. (1979), 'Elements of a plan-based theory of speech acts', *Cognitive Science* **3**(3), 177–212.

Cohen, P. R., Dalrymple, M., Moran, D. B., Pereira, F. C. N., Sullivan, J. W., Jr., R. A. G., Schlossberg, J. L. & Tyler, S. W. (1989), Synergistic use of direct manipulation and natural language, *in* 'Proceedings of the conference on Computer Human Interaction: CHI'89', ACM SIGCHI, Addison-Wesley, New York, NY, U.S.A., pp. 227–233. (Austin, TX, April 30 – May 4).

Colmerauer, A. (1990), 'An introduction to Prolog III', *Communications of the ACM* **33**(7), 69–90.

Dale, R. & Reiter, E. (1995), 'Computational interpretations of the gricean maxims in the generation of referring expressions', *Cognitive Science* **19**, 233–263.

Davis, R., Buchanan, B. G. & Shortliffe, E. H. (1977), 'Production rules as a representation for a knowledge-based consultation program', *Artificial Intelligence*.

Dowty, D. R., Wall, R. E. & Peters, S. (1981), *Introduction to Montague semantics*, Vol. 11 of *Studies in Linguistics and Philosophy*, D. Reidel Publishing Company, P.O.Box 17, 3300 AA Dordrecht, Holland.

Elhadad, M. (1989), Extended functional programmars, Technical Report CUC-420-89, Department of Computer Science, Columbia University, U.S.A.

Elhadad, M. (1993), Using argumentation to control lexical choice: a unification-based implementation, PhD thesis, Columbia University, New York, NY, U.S.A.

Feiner, S. K. & McKeown, K. R. (1991), 'Automating the generation of coordinated multimedia explanations', *IEEE Computer* **24**(10), 33–41.

Foley, J. D., van Dam, A., Feiner, S. K., Hughes, J. F. & Phillips, R. L. (1994), *Introduction to computer graphics*, Addison-Wesley Publishing Company, Inc., Reading, MA, U.S.A.

Fox, E. A. (1991), 'Advances in interactive digital multimedia systems', *IEEE Computer* **24**(10), 9–21.

Gallier, J. H. (1988), *Logic for computer science*, John Wiley & Sons, Inc., New York, U.S.A.

Gazdar, G. & Mellish, C. (1989), *Natural language processing in PROLOG*, Addison-Wesley, U.S.A.

Goel, A. K. (1997), 'Design, analogy, and creativity', *IEEE Expert* **12**(3), 62–70.

Gorry, G. A., Silverman, H. & Pauker, S. G. (1978), 'Capturing clinical expertise: a computer program that considers clinical responses to digitalis', *American Journal of Medicine* **64**, 452–460.

Grice, H. P. (1957), 'Meaning', *Philosophical Review* **66**, 377–388.

Grimes, J. E. (1975), *The thread of discourse*, Mouton, The Hague, The Netherlands.

Grosz, B. J. (1977), The representation and use of focus in dialogue understanding, Technical Report Technical note 151, Stanford Research Institute, Menlo Park, CA, U.S.A.

Grosz, B. J. & Sidner, C. L. (1986), 'Attention, intentions, and the structure of discourse', *Computational Linguistics* **12**(3), 175–204.

Güsgen, H. W. (1989), *CONSAT: a system for constraint satisfaction*, Research notes in artificial intelligence, 1st edn, Pitman Publishing, 128 Long Acre, London WC2E 9AN, U.K.

Hilton, M. L. & Anken, C. S. (1990), Map display system: an object-oriented design and implementation, Technical Report 90-54, Rome Air Development Center, Rome, U.S.A.

Hintikka, J. (1969), Semantics for propositional attitudes, *in* J. W. Davis, D. J. Hockney & K. W. Wilson, eds, 'Philosophical logic', D. Reidel Publishing Company, P.O.Box 17, 3300 AA Dordrecht, Holland.

Hovy, E. H. (1991), Approaches to the planning of coherent text, *in* C. L. Paris, W. R. Swartout & W. C. Mann, eds, 'Natural language generation in artificial intelligence and computational linguistics', Kluwer, Boston, MA, U.S.A., pp. 83–102.

Hovy, E. H. (1993), 'Automated discourse generation using discourse structure relations', *Artificial Intelligence* **63**, 341–385.

ILOG (1997), Optimization technology white paper: a comparative study of optimization techniques, Technical report, ILOG, Inc., 1901 Landings Drive, Mountain View, CA, 94043, U.S.A. This document can be retrieved from `http://www.ilog.com/`.

Iwasaki, Y. (1997), 'Qualitative reasoning and the science of design', *IEEE Expert* **12**(3), 2–4.

Jaffar, J. & Michaylov, S. (1987), Methodology and implementation of a CLP system, *in* J.-L. Lassez, ed., 'Logic programming:The 4th International Conference', Logic Programming, The MIT Press, Cambridge, MA, U.S.A.

Kamp, H. & Reyle, U. (1993), *From discourse to logic. Introduction to modeltheoretic semantics of natural language, formal logic and discourse representation theory*, Vol. 42 of *Studies in linguistics and philosophy*, Kluwer Academic Publishers, P.O.Box 17,3300 AA Dordrecht, Holland.

Kaplan, D. (1978), 'DTHAT', *Syntax and Semantics*. Vol. 9.

Kautz, H. A. (1991), A formal theory of plan recognition and its implementation, *in* 'Reasoning about plans', The Morgan Kaufmann Series in Representation and Reasoning, Morgan Kaufmann Publishers Inc., San Mateo, CA, U.S.A., chapter 2.

Lee, J. & Zeevat, H. (1990), Integrating natural language and graphics in dialogue, *in* D. Diaper, D. Gilmore, G. Cockton & B. Shackel, eds, 'Human-computer Interaction. Interact'90', IFIP, North-Holland, Amsterdam, Holland, pp. 479–484.

Leler, W. (1988), *Constraint programming languages: their specification and generation*, Addison-Wesley, U.S.A.

Lester, J. C. & Porter, B. W. (1997), 'Developing and empirically evaluating robust explanation generators: the KNIGHT experiments', *Computational Linguistics* **23**(1), 65–101.

MacGregor, R. M. & Brill, D. (1992), Recognition algorithms for the Loom classifier, *in* 'Proc. of the Tenth National Conference on Artificial Intelligence (NCAI92)', American Association for Artificial Intelligence, pp. 774–779.

Mackinlay, J. D. (1987), Automatic design of graphical presentations, PhD thesis, Stanford University, U.S.A.

MacLane, S. & Birkhoff, G. (1967), *Algebra*, The MacMillan Company, New York, NY, U.S.A.

Mann, W. C. & Thompson, S. A. (1988), 'Rhetorical structure theory: towards a functional theory of text organization', *Text* **8**(3), 243–281.

Maybury, M. T. (1990), Planning multisentential English text using communicative acts, PhD thesis, University of Cambridge, U.K. Available as Rome Air Development Center TR 90-411, In-House Report, December 1990, or as Cambridge University Computer Laboratory TR-239, December, 1991.

Maybury, M. T. (1992), 'Communicative acts for explanation generation', *International Journal of Man-Machine Studies* **37**(2), 135–172.

Maybury, M. T. (1993*a*), Planning multimedia explanations using communicative acts, in *Intelligent Multimedia Interfaces* [Maybury 1993*b*].

Maybury, M. T., ed. (1993*b*), *Intelligent multimedia interfaces*, AAAI Press/The MIT Press, Menlo Park, U.S.A.

McDermott, D. V. (1978), 'Planning and acting', *Cognitive Science* **2**, 71–109.

McKeown, K. R. (1982), Generating natural language text in response to questions about database structure, PhD thesis, University of Pennsylvania, U.S.A.

McKeown, K. R. (1985), *Text generation: using discourse strategies and focus constraints to generate natural language text*, Studies in Natural Language Processing, Cambridge University Press, Cambridge, U.K.

McKeown, K. R., Elhadad, M., Fukumoto, Y., Lim, J., Lombardi, C., Robin, J. & Smadja, F. (1990), Natural language generation in COMET, *in* C. Mellish, R. Dale & M. Zock, eds, 'Current research in language generation', Academic Press, London, U.K., pp. 103–136.

Montague, R. (1974*a*), English as a formal language, in *Formal philosophy: selected papers of Richard Montague* [Montague 1974*b*], pp. 188–221. Compiled and edited by R. H. Thomason.

Montague, R. (1974*b*), *Formal philosophy: selected papers of Richard Montague*, Yale University Press, London, U.K. Compiled and edited by R. H. Thomason.

Montague, R. (1974*c*), Pragmatics and intensional logic, in *Formal philosophy: selected papers of Richard Montague* [Montague 1974*b*], pp. 119–147. Compiled and edited by R. H. Thomason.

Montague, R. (1974*d*), The proper treatment of quantification in ordinary English, in *Formal philosophy: selected papers of Richard Montague* [Montague 1974*b*], pp. 247–270. Compiled and edited by R. H. Thomason.

Moore, J. D. (1989), A Reactive approach to explanation in expert and advice giving systems, PhD thesis, University of California at Los Angeles, U.S.A.

Moore, J. D. (1995), *Participating in explanatory dialogues: interpreting and responding to questions in context*, ACL-MIT Press Series in Natural Language Processing, A Bradford Book. The MIT Press, Cambrige, MA, U.S.A.

Moore, J. D. & Paris, C. L. (1989), Planning text for advisory dialogues, *in* 'Proceedings of the 27th. Conference of the ACL', Association for Computational Linguistics, Vancouver, B.C., Canada, pp. 203–211.

Neal, J. G. & Shapiro, S. C. (1991), Intelligent multimedia interface technology, *in* Sullivan & Tyler (1991).

Newman, W. M. & Sproull, R. F. (1979), *Principles of interactive computer graphics*, Computer Science, second edition edn, McGraw-Hill Book Company, New York, U.S.A.

Older, W. J. & Benhamou, F. (1993), Programming in CLP(BNR), *in* P. Kanellakis, J.-L. Lassez & V. Saraswat, eds, 'Proc. of the First Workshop on Principles and Practice of Constraint Programming, PPCP'93', Brown University, Box 1910, Providence, RI 02912, U.S.A. This document can be retrieved from `http://www.cs.unh.edu/ccc/archive/ppcp.html`.

Older, W. J. & Vellino, A. (1993), Constraint arithmetic on real intervals, *in* F. Benhamou & A. Colmerauer, eds, 'Constraint logic programming: selected research', Logic Programming, The MIT Press, Cambridge, MA, U.S.A., pp. 175–195.

Olmos-Sánchez, K. M. (1998), Implementación de un diálogo gráfico y lingüístico para sistemas multimodales inteligentes, Master's thesis, Centro Nacional de Investigación y Desarrollo Tecnológico. CENIDET/SEP, Cuernavaca, Mor., México. In Spanish. *TITLE: The implementation of a graphical and linguistic dialogue for intelligent multimodal systems.*

Paris, C. L. (1988), 'Tailoring object descriptions to the user's level of expertise', *Computational Linguistics* **14**(3), 64–78.

Pilkington, R. & Grierson, A. (1996), 'Generating explanations in a simulation-based learning environment', *International Journal of Human-Computer Studies* **45**, 527–551.

Pineda, L. A. (1989), GRAFLOG: a theory of semantics for graphics with applications to human-computer interaction and CAD systems, PhD thesis, University of Edinburgh, U.K.

Pineda, L. A. (1992*a*), 'Reference, synthesis and constraint satisfaction', *Computer Graphics Forum* **11**(3), 333–344.

Pineda, L. A. (1992*b*), Sketching, graphical anaphora and constraint satisfaction, *in* 'Proc. of the Samos workshop on task based explanation', Research Laboratory of Samos, University of the Aegean, Samos, Greece.

Pineda, L. A. (1993), 'On computational models of drafting and design', *Design Studies* **14**(2), 124–156.

Pineda, L. A. & Garza, E. G. (1997), A model for multimodal reference resolution, *in* E. Andre, ed., 'Proceedings of the workshop referring phenomena in a multimedia context and their computational treatment', ACL Special Interest Group on Multimedia Language Processing, pp. 99–117. In conjunction with ACL/EACL-97. Universidad Nacional de Educación a Distancia, Madird, Spain.

Pineda, L. A. & Garza, E. G. (1998), A model for multimodal reference resolution, Technical report, IIMAS, National Autonomous University of México, México. Submitted to *Computational Linguistics*. This report can be obtained from `http://leibniz.iimas.unam.mx/~luis/papers/cling-v98.ps`.

Pineda, L. A., Santana, J. S. & Massé, A. (1994), Satisfacción de restricciones geométricas: ¿problema numérico o simbólico?, *in* V. Estivill-Castro, ed., 'XI Reunión Nacional de Inteligencia Artificial', Sociedad Mexicana de Inteligencia Artificial, Universidad de Guadalajara, Guadalajara, Jal., México, pp. 105–123. In Spanish.

Puget, J. F. (1994), A C++ implementation of clp, *in* 'Proc. of the Second Singapore International Conference on Intelligent Systems'.

Roth, S. F. & Hefley, W. E. (1993), Intelligent multimedia presentation systems: research and principles, *in* Maybury (1993*b*).

Santana, J. S. (1987), La manipulación de un modelo geométrico bidimensional para el diseño de torres de transmisión, Master's thesis, Instituto Tecnológico y de Estudios Superiores de Monterrey (ITESM), Campus Morelos, Cuernavaca, Mor., México. In Spanish.

Santana, J. S. (1994), Parsing strategies, Technical Report IIE/Salford-SS-03, Instituto de Investigaciones Eléctricas, Apdo. 1-475, Cuernavaca, Mor., CP 62001, México. A monthly progress report on the IIE/University of Salford in-house PhD Programme. This report can be obtained via anonymous ftp from ftp://sgi.iie.org.mx/pub/checo/r3ss.ps.gz.

Santana, J. S. (1995), Quantification issues in a graphical language, Technical Report IIE/Salford-SS-14, Instituto de Investigaciones Eléctricas, Apdo. 1-475, Cuernavaca, Mor., CP 62001, México. A monthly progress report on the IIE/University of Salford in-house PhD Programme. This report can be obtained via anonymous ftp from ftp://sgi.iie.org.mx/pub/checo/r14ss.ps.gz.

Santana, J. S. (1996), Quantification revisited, Technical Report IIE/Salford-SS-17, Instituto de Investigaciones Eléctricas, Apdo. 1-475, Cuernavaca, Mor., CP 62001, México. A monthly progress report on the IIE/University of Salford in-house PhD Programme. This report can be obtained via anonymous ftp from ftp://sgi.iie.org.mx/pub/checo/r17ss.ps.gz.

Santana, S. & Pineda, L. A. (1995), Producing coordinated natural language and graphical explanations in the context of a geometric problem-solving task, *in* J. Lee, ed., 'Proc. of the First International Workshop on Intelligence and Multimodality in Multimedia Interfaces: Research and Applications.

IMMI-1', Human Communication Research Centre, 2 Buccleuch Place, Edinburgh EH8 9LW, Scotland, U.K.

Santana, S., Pineda, L. A. & Vadera, S. (1995), Textual and graphical explanations for geometric problem-solving systems, *in* L. E. Sucar & E. Morales, eds, 'XII Reunión Nacional de Inteligencia Artificial', Sociedad Mexicana de Inteligencia Artificial, Noriega Editores, Cuernavaca, Mor., México, pp. 106–112.

Santana, S., Pineda, L. A. & Vadera, S. (1998), An intelligent design system for overhead transmission line towers, *in* 'Proceedings of the 4th World Congress on Expert Systems', International Society for Intelligent Systems, Mexican Society for Artificial Intelligence, and Instituto Tecnológico y de Estudios Superiores de Monterrey (ITESM), Mexico City, Mexico, pp. 270–277.

Scott, A. C., Clancey, W. J., Davis, R. & Shortliffe, E. H. (1984), Methods for generating explanations, *in* Buchanan & Shortliffe (1984), pp. 338–362.

Searle, J. R. (1969), *Speech acts: an essay in the philosophy of language*, Cambridge University Press, Cambridge, MA, U.S.A.

Searle, J. R. (1979), *Expression and meaning: studies in the theory of speech acts*, Cambridge University Press, Cambridge, MA, U.S.A.

Shepherd, H. R. (1926), *The fine art of writing*, The Macmillan Co., New York, NY, U.S.A.

Shieber, S. M. (1986), *An introduction to unification-based approaches to grammar*, Chicago University Press, Chicago, U.S.A.

Shortliffe, E. H. (1976), *Computer-based medical consultations: MYCIN*, Elsevier, New York, U.S.A.

Simon, H. A. (1981), *The sciences of the artificial*, MIT Press, Cambridge, MA, U.S.A.

Steele, G. L. (1980), The definition and implentation of a computer programming language based on constraints, PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, U.S.A. Artificial Intelligence Tech. Rep. AITR.595.

Sullivan, J. W. & Tyler, S. W., eds (1991), *Intelligent user interfaces*, ACM Press/Addison-Wesley. New York, U.S.A.

Sutherland, I. E. (1963), 'SKETCHPAD: a man-machine graphical communication system', *Proc. Spring Joint Computer Conference. SJCC* **23**, 329–346.

Swartout, W. R. (1977), A digitalis therapy advisor with explanations, Technical Report TR-176, Massachusetts Institute of Technology, MA, U.S.A.

Swartout, W. R. (1983), 'XPLAIN: A system for creating and explaining expert consulting systems', *Artificial Intelligence* **21**(3), 285–325.

Swartout, W. R. (1990), Evaluation criteria for expert system explanation, *in* 'AAAI90 Workshop on Evaluation of Natural Language Generation Systems', American Association for Artificial Intelligence, Boston, MA, U.S.A.

Swartout, W. R., Paris, C. L. & Moore, J. D. (1991), 'Design for explainable expert systems', *IEEE Expert* **6**(3), 58–64.

Tanner, M. C. & Keuneke, A. M. (1991), 'The roles of the task structure and domain functional models', *IEEE Expert* **6**(3), 50–57.

Tarau, P. (1997), BinProlog 5.75 user guide, Technical report, Departament d'Informatique, Université de Moncton, Canada, E1A 3E9. This document can be obtained via ftp at clement.info.umoncton.ca /pub/BinProlog.

Tate, A., Hendler, J. & Drummond, M. (1990), A review of AI planning techniques, *in* J. Allen, J. Hendler & A. Tate, eds, 'Readings in planning', The Morgan Kaufmann Series in Representation and Reasoning, Morgan Kaufmann Publishers, Inc., pp. 26–49.

Thomson, C. (1986), 'Building menu-based natural language interfaces', *Texas Engineering Journal* **3**, 140–150.

Turing, A. M. (1950), 'Computing machinery and intelligence', *Mind* **49**, 433–460. Reprinted under the title *Can a Machine Think*, in *The World of Mathematics*, ed. James R. Newman, pp 2099-2123, Simon & Schuster (1956).

Wahlster, W. (1988), 'Distinguishing user models from discourse models', *Computational Linguistics* **14**(3), 101–103.

Wahlster, W. (1991), User and discourse models for multimodal communication, *in* Sullivan & Tyler (1991).

Wahlster, W. & Kobsa, A. (1986), 'Dialog-based user models', *Proceedings of the IEEE* **74**(7), 948–960.

Wahlster, W., André, E., Finkler, W., Profitlich, H. J. & Rist, T. (1993), 'Plan-based integration of natural language and graphics generation', *Artificial Intelligence* **63**, 387–427.

Waltz, D. L. (1978), 'An English language question answering system for a large relational database', *Communications of the ACM*.

Weizenbaum, J. (1966), 'ELIZA – a computer program for the study of natural language communication between man and machine', *Communications of the ACM* **9**(1), 36–45.

Welch, B. B. (1995), *Practical programming in Tcl and Tk*, Prentice Hall PTR, New Jersey, U.S.A.

Williams, W. (1893), *Composition and rhetoric*, D. C. Heath and Co., Boston, MA, U.S.A.

Winograd, T. (1972), *Understanding natural language*, Academic Press, New York, U.S.A.

Winograd, T. (1973), A procedural model of language understanding, *in* R. C. Schank & K. M. Colby, eds, 'Computer models of thought and language', W. H. Freeman and Company, Publishers, New York, U.S.A., pp. 152–186.

Woods, W. A. (1970), 'Transition network grammars for natural language analysis', *Communications of the ACM* **13**(10), 591–606.

Woods, W. A. (1987), Augmented transition network grammar, *in* S. C. Shapiro, ed., 'Encyclopaedia of artificial intelligence', Wiley, New York, U.S.A., pp. 323–333.