A method for validating a conceptual model by natural language discourse generation

Hercules Dalianis

SYSLAB
Department of Computer and Systems Sciences
The Royal Institute of Technology and
Stockholm University
Electrum 230
S-164 40 Kista
SWEDEN
ph. (+46) 8 16 16 79
E-mail: hercules@dsv.su.se

Abstract. The support systems for conceptual modeling of today lack natural language feedback. The paper argues for the need of natural language discourse for the validation of a conceptual model. Based on this conclusion a suggestion is made on a natural language discourse generation system as a validation tool and also as a support tool in simulating a conceptual model. Various appropriate natural language discourses are then proposed in the paper. To conclude the paper a support system based on the natural language generation techniques of today and on previous working systems constructed by the author is suggested.

1. Introduction

During the construction of a large scale computer system, one should decide on the functionality of the system before starting its implementation. Constructing large computer systems is costly and when errors occur it becomes increasingly more difficult to correct them in later stages. Therefore a technique for modeling an information system has been developed, the so called conceptual modeling technique. This paper concerns the validation of a conceptual model.

Conceptual modeling is according to [ISO82] both a method for representing the user's view of the information and connect this view to the physical storage of the information that results from a system analysis. Since the conceptual model is described in a formal language the information can be difficult to understand for an inexperienced user. Therefore it is sometimes advisable to adapt the presentation of this information.

Most individuals are not well trained to understand formal descriptions, but every one understands at least one natural language, (NL). The advantage of using natural language is that a novice user does not have to learn a complicated language or formalism to understand the conceptual model.

Natural language is also justified to use because it will give the end users a direct feedback of the semantics of the formal representation, actually we will lower the

conceptual barrier of the end user by using NL. Furthermore another advantage of natural language generation, (NLG), from a computer is that people who can help the novice user with explanations in NL very often are occupied with more important tasks. This is, for example, a well known scenario in large companies, so a self instructing computer system would help the novice user to utilize the system.

This paper shows why it is important to do a natural language generation from a conceptual schema and why the NLG should be in discourse form. A discourse is a piece of text or a set of logical interconnected sentences. The reason for the need of discourses and not single sentences aroused from the previous approaches of creating natural language descriptions of a conceptual model in [Dali89], where the constructed natural language generation system was critiqued for generating a set of unordered sentences. A discourse is also necessary to use to explain the overall semantics of the conceptual model, while each part of the conceptual model is related to one or many other parts of the model.

A number of appropriate discourses is proposed to answer the questions which are supposed to be posed by the users of a natural language generation system. The proposed discourses are analyzed with Hobbs' coherence relations [Hobbs85, 90] and a discourse grammar will be generalized from the discourses.

The discourse grammar and the methodologies of previous constructed NLG-systems, will be used to propose a natural language discourse generation system for a conceptual model. The generation in the NLG-system will be carried out at deep level and not a surface level, (To be explained later). The discourse is created using a subset of Hobbs' coherence relations, [Hobbs85, 90]. The discourse grammar is written in a Definite Clause Grammar, (DCG), grammar formalism [Pereir80, Clock84] in a Prolog-style syntax, where the formalism is extended with various Prolog predicates and features which controls the execution of the grammar.

Paper outline

Section 2. is a short introduction to the field of conceptual modeling and an overview of some support tools for conceptual modeling. Arguments for why these tools are not powerful enough to fulfill their tasks will be presented, then follows examples on appropriate natural language discourse generation for validation. Section 3. discusses discourse structure and text generation technology. Section 4. makes a proposal on a system for text generation which uses the technology in section 3 and a discourse grammar for describing the proposed texts of section 2.

2. Support systems for information and data modeling

2.1. The information system development process

The purpose of an information system is to store and retrieve information about a real world domain. During the construction of an information system various small problem can emerge. To avoid flaws in later stages of the information system development process, an information system has to be described at a higher level abstraction than at the programming level. For this purpose various high level and abstraction languages have been developed. One of them is the so called Conceptual MOdeling Language, (CMOL) [Buben84, 86].

A conceptual model, (CM), describes a piece of the real world, a domain, in an un-ambiguous and non-redundant way. A conceptual model is built and revised during a

comparatively long period of time. Building a conceptual model of a system or an organisation is by necessity an iterative process ranging from a vague idea of what it will do to a full fledged system. But even when the computer system is fully developed new extensions of the system will be needed in order to cope with changes is the domain (real world).

2.1.1. Validation

The validation process checks if the constructed conceptual model is correct according to the real world. The validation process shall detect flaws and give suggestions for correction. This can be achieved by analyzing the conceptual model with expert systems, by doing consistency checking, to simulate the information system from the conceptual model or by paraphrasing the conceptual model back to the user for validation. The graphic representation of the domain is also a part of the validation tool.

Validation is one of the most important tasks in the requirements engineering, since the validation will reveal if something has gone wrong in the conceptual modeling phase. In this phase the domain expert will make his judgement if the conceptual model is the one he intended and that in respect to real world or domain.

A method of validation is to paraphrase the model into natural language, (NL). NL is a reference for all people involved in the development of the system. The paraphrasing is usually performed by the system engineer, but it would be convenient if it could be carried out automatic, so the domain expert himself could validate the conceptual model without having deeper knowledge in the conceptual modeling formalism.

2.2. A Conceptual model

According to [Boman91] and [Buben84,86], an information system contains a *conceptual model* and *an information processor*. A conceptual model, (CM), contains a conceptual schema, (CS) and *an information base* or a fact base. The conceptual schema describes the language used for reasoning about the object system and decides which statements are allowed to be included in the information base. The conceptual schema can be considered to be a *skeleton* description of the real world or domain, i.e. which entities possible can exist. The information base contains statements describing an object system i.e. the domain or real world. The purpose of the information processor is to enable users to query and update the conceptual schema and the information base.

A conceptual model consists of a static and a dynamic part. Certain rules concern the static properties of an object system, whereas other rules describe its dynamics. By a *static rule* is meant an expression, which takes into account only a single state. By a *dynamic rule* is meant an expression, which takes into account several states.

A conceptual schema consists of entity types, (objects) , relations, attributes, ISA-links, events, Static- and Dynamic integrity constraint rules and Derivation rules, (SDD-rules), and finally the information base contains instances, (facts). The SDD-rules are also mentioned in [Lloyd87].

2.3. Different support tools

A large number of support tools for conceptual modeling have been constructed. Some of them are called explicitly CASE, (Computer Aided Software Engineering), tools. What these tools have in common is that they are support systems for conceptual modeling. Examples on some of them are ALECSI, [Cauv91], RIDL*, [DeTro88], AMADEUS,

[Black87], MOLOC, [Johan90], etc. To carry out the tasks of requirements and design engineering, with the subtask of knowledge acquisition and validation respectively design and verification they use a broad spectrum of techniques as for example: natural language input, graphics, expert system techniques, simulating, concistency checking etc. [Kuntz89, Tauzo89, Wohed88].

2.3.1. Natural language input

Natural language, (NL), makes a system user friendly because NL lowers the conceptual barrier such that the user easier can approach the system. Systems working with natural language input and sometimes in combination with graphics are for example:AMADEUS, [Black87], a support tool which uses a combination of graphic and natural language input. Another system is ALECSI, [Cauv91], with its predecessor OICSI, [Cauv88], which supports both knowledge engineering and knowledge acquisition, modeling and validation and process engineering as guidance and explanation. But none of the NL-interfaces mentioned above has any natural language generation component.

2.3.2. Prototyping

MOLOC stands for MOdeling in LOgiC [Johan90], which is a prototype semantic database management system. MOLOC is a support system for conceptual modeling, where you can design and execute a conceptual model of a database. MOLOC shows how to do a fast prototype and testing without having to construct the real system. MOLOC has a graphical interface called MGI, MOLOC-Graphical Interface, You can design your conceptual schema in MGI, but the SDD-rules have to be stated directly in the MOLOC formalism.

2.4. Reasons for having natural language generation in conceptual modeling

To understand something requires a reference point. Without background knowledge it is difficult to understand. People are not tutored and do not gain understanding of a conceptual model, (CM), just by changing the model, there must also be examples and references, but of the systems discussed systems above, only the MOLOC system provides this. Further none of the above discussed systems has any natural language generation component.

Natural language is used by man both for communication and for reasoning. NL is for the brain, what the hand is for handcraft. NL is a tool which without there would not be communication between humans. We are using NL during a great part of the conceptual modeling process, and therefore it would be convenient with automatic NL input and output.

Many of the support systems are developed by different manufacturers and do not use a standard notation neither in the input nor in the output, therefore the validation would be easier carried out using a natural language generation system, since natural language is rather standardized.

The different users

Three groups of individuals which are using conceptual models and consequently could use a natural language generation system can be distinguished, namely: the domain expert, the system engineer and the end user. They have different needs and knowledge.

1) The *domain expert* , (DE), will need the model paraphrased to check if everything is correct represented, if all facts are present, if the concepts have correct names and if the model is logical.

2) The *system engineer,* (SE), is interested in the function of the conceptual model . The function for building a computer system. If the purpose of the model is correct.

3) The *end user,* (EU), wants to get a quick overview of the model to know how the knowledge is stored and how to navigate in the system.

Here we have separated three users of the conceptual model and three various types of information which need to be paraphrased into natural language.

2.5. Dictionary writing and knowledge acquisition

Domain experts maybe think they know everything in their area, but they have not structured all their knowledge. The process of knowledge acquisition and construction of the conceptual model, (CM) and validation of the CM will help them to structure their knowledge. Sometimes concepts can be merged together and sometimes they have to be separated. This is a typical conceptual modeling situation.

When the NLG-system generates a NL-description of a part of the CM for a domain expert, then s/he will discover that wrong words or concepts has been used. This explains why also the dictionary writing part is important. Another reason for the dictionary writing is important is the different users of the conceptual modeling tool will need to have concordance about the meaning of the different concepts. After this process the defined dictionary can be used for paraphrasing the conceptual model.

The system design must be transportable i.e. easy to adapt to different users and domains and there must be possibilities to define new words easily, [Grosz87].

There are various reasons for paraphrasing a CM to NL.

- To lower the conceptual barrier of the user.
- To ease the understanding of the CM-formalism for a DE.
- To give possibility for a DE to validate the model to himself.
- To ease the understanding of the domain for a SA.
- A method for detecting errors and traps in the CM.
- To focus on certain aspects of a CM.
- To have a reference language (NL) which the DE, SE and EU understands.
- To teach the conceptual model formalism for a DE or a SE.
- To introduce a newly assigned person to the domain.
- To give a quick overview in the beginning of the conceptual modeling phase where the persons involved in the modeling phase need to know what has been modeled until now.
- To inform an end user of a natural language interface to a database how the database is organized and which questions s/he can ask to obtain information from the database.
- The dictionary writing for the NLG-system will enhance the validation of the CM.

The generation will be divided and combined between generated information from both the CS, which describes the type of information and how it is stored in the database, and corresponding instances from the database. This combination will help both the design engineer to validate the CM and the end users to navigate in the computer system.

The natural language generation can do a sorting, selection or enumerating of e.g. subclasses, which will help the DE to remember if he has forgotten anything.

The generation from a CM can be performed at two levels one at a general conceptual schema level and the other on a conceptual model level where also instances of objects are used for explaining.

The questions which the system should handle

Here follows a set of questions and commands:

- What do you know ?
- Describe an entity type !
- Describe instances of entity types !
- What is the relationship between different entity types ?
- What is the relationship between different instances ?
- What events are there ?
- What SDD-rules constraint the CM ?
- Which entities types are affected by which event ?
- List all entity types, (events, SDD-rules, instances) !

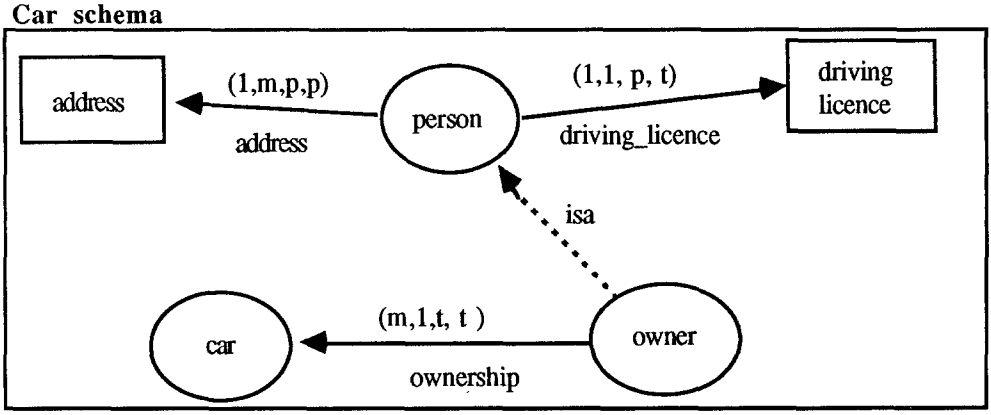
2.6. Various proposed discourses

Here follows examples on proposed discourses from both the author and from the users of the conceptual modeling tool MOLOC and MGI. These text examples should help the system engineer, (SE), the domain engineer, (DE) and the end user, (EU) in their various tasks. The proposed texts are written in *italics*.

These following proposed ideas are partly implemented on sentence level in AAIS-Prolog on a Macintosh. [Dali89,90] the sentence level translation has also been discussed in [Chen83].

The question types could for example by selected from menus, and the objects by pointing with a mouse on a graphical conceptual schema. The input to the question types could be extended with some limited text input.

Car schema



Car information base or fact base

```
person(carl).
driving_licence(carl,121).

person(lisa).
address(lisa,211).

owner(robert).
address(robert,311).
driving_licence(robert,321).

car(volvo).
ownership(robert,volvo).
```

What do you know ? (A sort of help function)
What do you know ?

list all * (* stands for parts of the CM)
entity types
attributes
events
relationships between different entity types
relationships between different instances
relationships between entities types and events
SDD-rules
Static integrity constraint rules
Dynamic integrity constraint rules
Derivation rules.
facts:
instances of entity types

List all [A] questions (The user will get an overview of the CM contain)
List all entity types !

person
owner
car

What-is-[A] ? questions (The user asks about an entity type or instance)
What is a person ?

1) *Some persons are owners*
A person can have exactly one address
and exactly one driving licence
Carl is a person.
He has a driving licence 121 and no address
Lisa is a person
She has an address 211 and no driving licence (Informing about no existence of facts is optional)

What is Carl ?

2) *Carl is a person.* Facts
He has a driving licence 121 and no address

A person can have exactly one address
and exactly one driving licence — Schema (This part is optional)

What does following SDD-rule mean ?	
The Static rule expressed in MOLOC inconsistent :- ownership(P,C), not(address(P,A)).	In Logic IF P owns C AND NOT P has_address A THEN inconsistent
The inconsistency is equivalent with that the static integrity constraint rule is false	
3) *Every owner (person) who owns a car must have an address.*	

What is an owner ?

4) *An owner is a person,* — Schema
An owner has at least one car.
Every owner who owns a car must have an address.
Robert is an owner and
has a address 311 and a car Volvo — Facts

What is Robert ?

5) *Robert is an owner and has a car Volvo,* — Facts
he has an address 311 and a driving licence no 321
An owner is a person and
can have exactly one address — Schema (This part is optional)
and exactly one driving licence

What-is-the-relation-between [A] and [B] and [C....] questions

What is the relation between a person and a car ?

6) Some persons are owners, who must *own one or more cars (at least one car)*

What is the relation between a car and a person ?

7) *A car can be owned by exactly one person, who must be an owner*

What is the relation between a owner and a car ?

8) *Every owner is a person,who must own one or more cars and*
every owner must have an address
Schema — S-rule

What is the relation between a car and a owner ?

9) *A car is owned by exactly one owner.*

What is the relation between a person and an owner ?

10) *Every person is an owner* — Error in the ISA-relation
(A person is an owner)
Robert is a person and an owner

If the ISA-relation was in the wrong direction, then the NL-sentence above would be generated otherwise the NL-sentence below would be generated.

11) *Some persons are owners* OK in the ISA-relation
(A person can be an owner)
Robert is a person and an owner

What is the relation between an owner and a person ?
12) *Every owner is a person*
Robert is an owner and a person

What is the relation between Robert and car ?
13) *Robert is an owner of a car Volvo*

What is the relation between a car and a Volvo ?
14) *A car can be a Volvo*
A Volvo is a car

What is the relation between a driving licence and a car ?
15) *A driving licence belongs to a person who can be an owner who has a car.*

What is the relation between a car and a driving licence ?
16) *A car is owned by an owner who is an person who can have a driving licence.*

What is the relation between a car and an address ?
17) *A car is owned by an owner who is a person who can have an address.*

What is the relation between a person and an address ?
18) *A person can have exactly one (and only one) address*

What is the relation between an address and a person ?
19) *An address can have at least one person (living there)*

Events
Here follows questions which concern the dynamic part of the model.
What events are there on cars ?
buy_car Enumerating

What are affected by the event buy_car ?
buy_car affects:
car
person
owner
address
and
a static integrity constraint rule

What are the relations between the event buy_car and car, person, owner, address ?
20) *If a person buys a car*
then he must became an owner
Every person who owns a car must have an address.

Here follows an execution with MOLOC [Johan90] enhanced with a proposed NLG-system which gives explanation of what is carried out during an event.

The user executes an event.

Which event to perform ?
Let Carl buy a car VOLVO !
21) *If you let Carl buy the car VOLVO*
then the totality between attributes will be violated
A car must be owned by exactly one owner
Carl owns the car VOLVO
Robert owns the same car VOLVO

The user executes an event.

Which event to perform ?
Let Carl buy a car SAAB !
22) *If you let Carl buy a car*
then the static integrity constraint rule will not hold which says
For a person to became an owner of a car s/he must have an address
Carl has no address

3. Text generation

3.1. Discourse structure and analysis

Syntax and semantics of sentences have been well-studied and the syntax of a sentence is well defined. Previous constructed systems which generated natural language sentences from conceptual models are described in [Dali89,90], however, given that the information contained in a conceptual model is context dependent there has been a demand to describe the relationships between natural language sentences. i.e. a discourse which is a set of related and interconnected natural language sentences.

If we look at a discourse we know that the sentences there are more loosely kept together, than the parts of the sentences themselves. For example: if we mix a set of sentences in a discourse we would probably still understand the message but this would require a large effort and some information would of course be lost. A discourse which is easy to understand with less effort is called *coherent.*

There exists a large amount of methods for analyzing discourses and understand how sentences are connected. The main principle is to find so called key words or rhetorical primitives which are described in Rhetorical Structure Theory, (RST), [Mann84, Mann88] or the coherence relations of Hobbs, [Hobbs85,90]. The coherence relations, for example, relates two or more sentences to one unit, and this unit in turn is ordered in a higher hierarchical structure, which describes the entire discourse.

The assumption made is: A discourse is coherent if its sentences can be fit into one overreaching relation.

Here follows Hobbs' coherence relations:

Occasion relations	
occasion	a weak causal relation, a coherence between events in the world.
cause	special case of the occasion relation, the normal causal relation (keyword *if ..... then..*).
enablement	special case of the occasion relation, the first assertion enables the second assertion.
Evaluation relations	
evaluation	a meta comment, (keyword e.g. *Do you understand so far... This is good news*)
Ground-figure and explanation relations	
ground-figure	also called background, it is old information, background information, often time related and related to new information.
explanation	is an inverted cause, i.e. a proposition is caused by something. (keyword *because*)
Expansion relations	
elaboration	describes an object or event more in detail, (keyword *i.e. that is)*
exemplification	gives an exemplification of an type of event or object (key word *for example*).
generalization	a proposition is generalized, (keyword *it is well known that...*) it is the same as exemplification, but the order is switched.
parallel	two or more sequential propositions at the same level describing the same object or event level.
violated-expectation	two different assertions gives two different results a proposition is true but... (keyword *but*).
contrast	two similar assertions gives two completely different results.

3.2. Text generation technology

We will take a look at the state of art in the text generation technology to investigate what is possible to achieve:

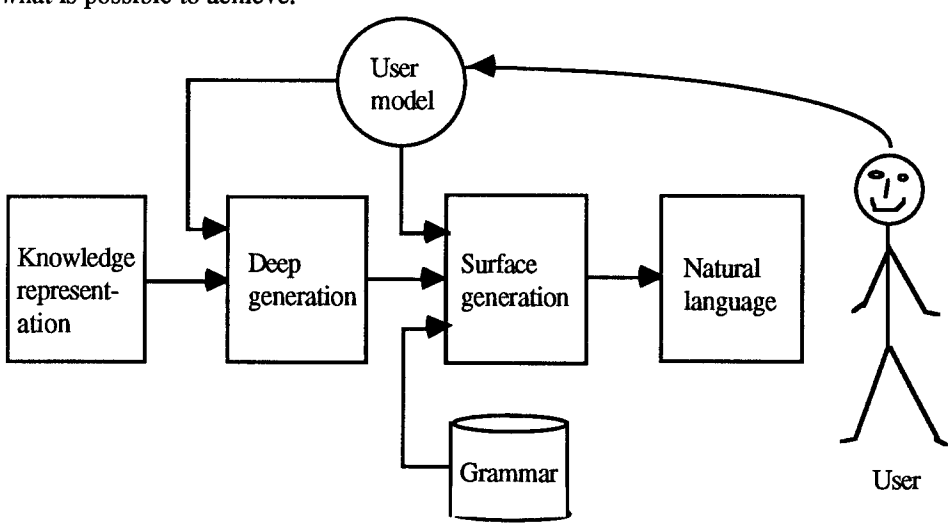


Fig 1. An "average" text generation system

3.2.1. Deep and surface generation

Many researchers consider the task of natural language generation from a computer to consist of two sub tasks, namely:

1) Deep generation
2) Surface generation

In the first sub task, the deep generation, it is decided what to say from the abundant knowledge base. The planning and organization of the information content is determined. Next it is concluded in what form it should be presented according to a specific user model. In which order should the sentences be generated to make the text coherent. The deep generation in a computer must make similar steps as when a human generates text. During the second sub task, the surface generation, it has to be decided how to say it, i.e. the realization of the syntactic structures. Moreover a selection of the lexical items appropriate to express the content has to be made. This paper concerns the first subtask the deep generation component of the natural language generation system.

3.2.2. Deep generation

No one has yet enumerated the kind of tasks a text planner should be able to do, but some of the problems are known. One problem is the content determination, i.e. to select what to say of the abundant information in a knowledge base. A partial solution to this problem is dependent on the question and the knowledge level of the user.

There is various approaches for solving these problems, for example **discourse strategies**. They are usually schema based or have a ready text plan to be used for generation. An example on this in the system **TEXT** by McKeown, [McKeo85a,85b].

To build the system TEXT McKeown had to analyze great deal of text written for the three purposes of *defining, comparing* and *describing* different objects. In the texts she found four rhetorical predicates and with these four rhetorical primitives McKeown defined four different schemas. These schemas can be used for answering three types of meta-questions about the contents in a data base.These three types of questions or commands are:

1) How is an object defined ?
2) What is the difference between two objects ?
3) Describe available information !

A second approach to constrain the knowledge base is **planning and reasoning**, which is concerned with manipulating a knowledge representation with a set of rules to achieve a goal. A possibility is to have a cooperative dialogue with a user which adds knowledge to the system and constrains the knowledge base, an example on this is the system KAMP, Knowledge And Modality Planner, by Appelt, [Appelt85].

The third approach in constraining the knowledge base is by utilizing a **user model.** One problem in natural language generation is to know on what level the user is and what type and organization of the text is needed for understanding the text message. One method is to have different user models of different users and generate a text which is adapted to that model. Paris has in [Paris85,88] described different strategies of generation depending on the knowledge level of the user. Paris studied two different types of encyclopedias written for adults and for children. She discovered that texts made for the adults describe all the parts of the objects, while in the texts written for the children the function, i.e procedural information, of the objects is explained. Moreover in the children encyclopedias the complete chain of inferences is described and there is more redundancies

than in the text for the adults. Many times an expert can be a novice in one part of the domain, or vice versa, a novice can be expert in some other part of the domain. Therefore it is necessary to adapt to this type of users. Various methods for acquiring a user model and applying it to a text generation system for conceptual modeling are described in [Dali91].

3.3. Problem and hypothesis

A conceptual model is a passive and non-redundant and non-ambiguous representation of the real world which can partly be drawn in graphics, but there are parts like the SDD-rules and the fact base which can not be drawn. There is also a set of complicated dependencies between different parts of the conceptual model. The problem for many users of the conceptual models is to have an overview and understanding the represented concepts, therefore it seems obvious to translate the model into natural language. Since the conceptual model is heavily dependent on all its parts it seems appropriate to have a natural language discourse generation. This can be achieved by connecting each part of the conceptual model with the coherence relations of Hobbs [Hobbs85,90]. This will make the natural language generation produced from the system coherent and easily read.

A technique for natural language generation is available and the problem is to find what parts of the conceptual model corresponds to which coherence relations. The sentence level generation has already been carried out by the author and described in [Dali89,90].

4. System overview

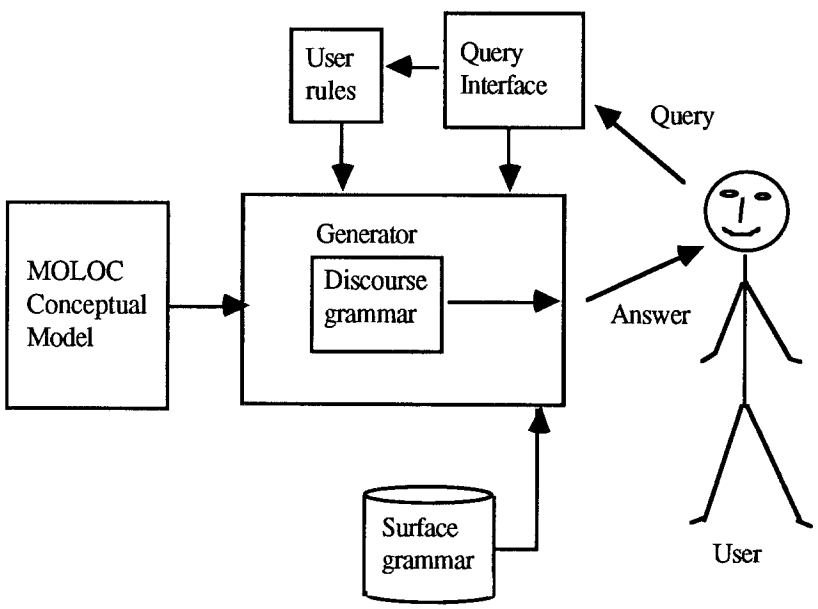


Fig 2. Overview of the proposed natural language generation system

The system is built around three modules: *the user rules, discourse* and *surface grammar,* with a query interface which processes the question input from the user. The query interface passes the processed input both to the user module and to the generator. The user module will find the intentional goals of the user and at what level the user is by the way

module will find the intentional goals of the user and at what level the user is by the way s/he is asking questions, this is described in [Dali91]. This information together with the query from the user will also answer to the user's first intention: to reply to the user's question by making a selection of information from the knowledge base.

The generation is carried out in three steps:

1) The user rules: Find out what the user knows and builds a dynamic user model which helps to select the correct information from the conceptual model.
2) The discourse grammar: Builds a discourse structure from the selected information. The discourse structure should fulfil the intentions and goals of the user.
3) The surface grammar is not described here. It is at a syntactic level and belongs to the surface generation.

4.1. Extracting discourse grammar rules.

An example on how the discourse grammar rules are defined from one of the previous proposed examples.

What is a person ?

1) Some persons are owners
2) A person can have exactly one address and
3) (A person can have) exactly one driving licence
4) Carl is a person.
5) He has a driving licence 121
(6) and (he has) no address)
7) Lisa is a person
8) She has an address 211
(9) and (She has)no driving licence)

Gives following discourse tree:

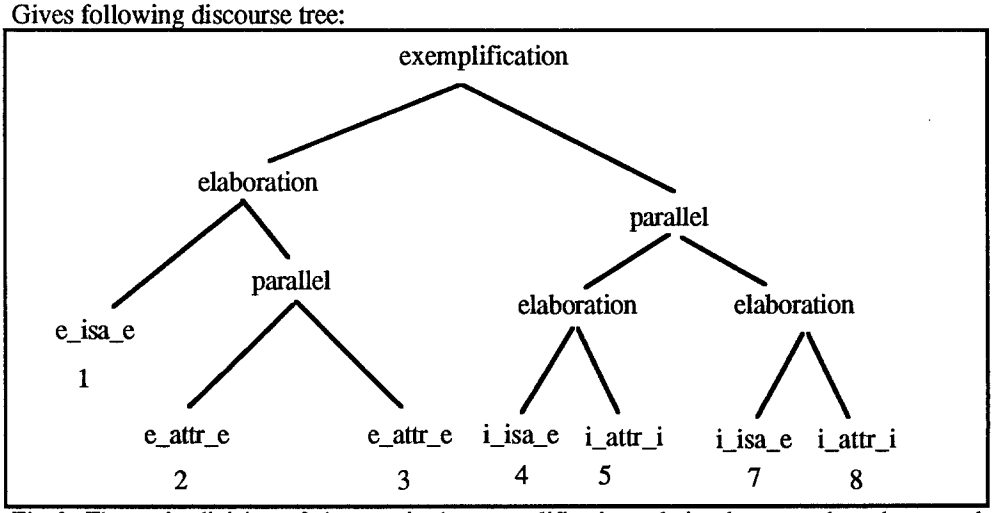


Fig 3. The main division of the text is the exemplification relation between the schema and instance level. The schema leaf is divided by the elaboration relation which elaborates the ISA relation into two equivalent attribute statements related by a parallel relation. In the instance leaf we find a parallel relation which describes two pieces of a discourse at the same level. Each discourse is described by an elaboration relation which elaborates an ISA-statement into an

attribute statement at instance level. (The numbers in the discourse tree corresponds to each sentence above).

Discourse grammar

The following discourse grammar will transform a piece of a conceptual model to a discourse structure which then easily can be transformed to a text by a surface generator. The discourse grammar below is defined according to the example and method above, from the previous proposed examples and from some other examples as well.

The representation below in Backus-Naur form or a Context Free Grammar.
{ } means that something is optional. | means or ( )* means none, one or many times.

Top level

```
DISCOURSE        ::== EXEMPLIFICATION
DISCOURSE        ::== {GENERALIZATION}        (optional)
DISCOURSE        ::== ELABORATION {GENERALIZATION}
DISCOURSE        ::== CAUSE
DISCOURSE        ::== EXPLANATION
```

Rest

```
ELABORATION      ::== E_ISA_E PARALLEL
ELABORATION      ::== I_ISA_E PARALLEL
ELABORATION      ::== I_ISA_E I_ATTR_I
ELABORATION      ::== E_ISA_E E_ATTR_E
ELABORATION      ::== E_ATTR_E ELABORATION
ELABORATION      ::== EVENT_NAME CAUSE
ELABORATION      ::==PARALLEL EXPLANATION
ELABORATION      ::==PARALLEL PARALLEL

EXEMPLIFICATION  ::==E_ISA_E PARALLEL
EXEMPLIFICATION  ::==E_ATTR_E I_ATTR_E
EXEMPLIFICATION  ::==ELABORATION ELABORATION
EXEMPLIFICATION  ::==ELABORATION PARALLEL
EXEMPLIFICATION  ::==CAUSE PARALLEL

EXPLANATION      ::== CAUSE ELABORATION
EXPLANATION      ::== PARALLEL E_ISA_E
EXPLANATION      ::== CAUSE I_ATTR_I
EXPLANATION      ::== SUCCEED_FAIL S_RULE

GENERALIZATION   ::==ELABORATION PARALLEL
GENERALIZATION   ::==ELABORATION ELABORATION

PARALLEL         ::== ELABORATION ELABORATION
PARALLEL         ::== I_ISA_E I_ISA_E
PARALLEL         ::== E_ATTR_E E_ATTR_E (E_ATTR_E)*
PARALLEL         ::== E_ATTR_E E_ATTR_E (CAUSE)*
PARALLEL         ::== I_ATTR_I I_ATTR_I (I_ATTR_I)*
PARALLEL         ::== S_RULE S_RULE
PARALLEL         ::== PRECONDITION PRECONDITION

CAUSE            ::==E_ISA_E S_RULE
```

CAUSE	::== E_ATTR_E S_RULE
CAUSE	::== E_ATTR_E E_ISA_E
CAUSE	::== EVENT EXPLANATION
CAUSE	::== PARALLEL S_RULE
CAUSE	::== ELABORATION S_RULE

Terminals are sentences or parts of the CM

E_ISA_E	::== ENTITY TYPE1 ISA ENTITY TYPE2 (schema)
I_ISA_E	::== INSTANCE ISA ENTITY TYPE (mixed)
E_ATTR_E	::== ENTITY TYPE1 ATTR ENTITY TYPE2
I_ATTR_E	::== INSTANCE ATTR ENTITY TYPE
I_ATTR_I	::== INSTANCE ATTR ENTITY TYPE
EVENT	::== event name or type of event
PRECONDITION	::== precondition in an event
(SUCCEED_FAIL)	::== (the rule will succeed)|(the rule will not hold)
(SUCCEED_FAIL)	::== (the attributes will hold)|(the attributes will not hold)
S_RULE	::== (STATIC | DYNAMIC) rule
D-RULE	::== DEDUCTION rule

The above discourse grammar describes the connection between a conceptual model and a discourse form. This means that a question of the user together with the available conceptual model will create a discourse according to Hobbs classification of discourse structure.

The discourse grammar is almost executable as it stands in Prolog, only some minor syntactic changes are needed. The problem is then that the grammar will overgenerate and that it does not have a control mechanism. This control mechanism will be created by using the same method which was used for analysing the discourses and by implementing features and control predicates.

4.2. The implementation language is Prolog

The implementation language for the different already programmed parts is Prolog, which is well-suited both for parsing and for generation of natural language. The reason for this is that Prolog was developed for doing natural language research. Prolog is also well suited for fast prototyping specially for natural language processing due to its modularity. Today a large number of compatible Edinburgh syntax Prologs are available, e.g. SICStus, Quintus, Arity and AAIS Prolog for UNIX, MS-DOS, OS/2 and for the Macintosh operating system. Prologs which can be both interpreted and compiled and are fast and efficient and which can call or be called from other programming languages such as C or Pascal.

4.3. The discourse grammar and the control of the generation

A draft implementation using a subset of the discourse grammar has been carried out. The discourse grammar is a so called Definite Clause Grammar, DCG, grammar, [Pereir80, Clock84], and is executed and controlled by the Prolog interpreter. The execution of the grammar is performed backwards. The terminals consists of the selected information for each question. Each terminal consists of a single sentence at either schema, instance or mixed level. Further more various *features* and *predicates* of the grammar are implemented for controlling the generation.

The *features* with the values i,e,r,_ for describing entity types and instances of them.

i stands for instance level

e for entity type or schema level
r for rule
_ for the anonymous variable or irrelevant.

The coherence relation *exemplification* extended with features is an example of the above control of the grammar. Exemplification is a divider between explanation at the schema level, and explanation at the instance level i.e.features entity type e or instance i. An other example is the *elaboration* relation which can be performed both at schema and instance level, but it has to be kept either of the ways. This both cases can be seen in the extract of the discourse grammar below.

Predicates

The predicate, *not_occur/2*, checks whether any part of the terminals are used more than once for generation, i.e. none of the terminals occur more than once in the discourse tree. The predicate *same/2*. is used to check if two clauses has any connection to each other at all, for example in elaboration. For example: to talk about same entities.
When a piece of discourse tree is generated it is checked for not violating the generation rules.

Extract from the discourse grammar which generates the discourse below

```
discourse(exemplification(E))
        --> exemplification(_,E).
exemplification(_,(elaboration(E) & parallel(P)))
        --> elaboration(e,E), parallel(i,P),{not_occur(E,P),!}
elaboration(i,i_isa_e(I1) & i_attr_i(I2))
        --> [I1,I2],{i_isa_e(I1), i_attr_i(I2),not_occur(I1,I2),same(I1,I2)}.
elaboration(i,i_isa_e(I) & parallel(P))
        --> [I],{i_isa_e(I) , parallel(i,P),not_occur(I,P)} .
parallel(e,(e_attr_e(E1) & e_attr_e(E2)))
        --> [E1,E2],{e_attr_e(E1), e_attr_e(E2),not_occur(E1,E2),same(E1,E2)}.
parallel(i,elaboration(I1) & elaboration(I2))
        --> elaboration(i,I1), elaboration(i,I2),{not_occur(I1,I2),!}.
```

Example on a generation

```
Question: What is a person ?

?- list_db.                                          selected sentences
[some,persons,are,owners]
[carl,is,a,person]
[lisa,is,a,person]
[carl,has,an,driving_licence,121]
[lisa,has,an,address,211]
[a,person,can,have,exactly,one,address]
[a,person,can,have,exactly,one,driving_licence]
   yes

?- discourse(TREE,NL).
   TREE = exemplification(elaboration(e_isa_e(
                                [some,persons,are,owners])
                               &
                              parallel(
```

```
                                [a,person,can,have,
                                 exactly,one,address])
                               &
                               e_attr_e(
                                 [a,person,can,have,
                                  exactly,one,
                                  driving_licence])))
                    &
                   parallel(elaboration(
                            i_isa_e([carl, (is),a,
                                     person]) &
                             i_attr_i(
                               [carl,has,a,
                               driving_licence,121]))
                           &
                           elaboration(
                             i_isa_e([lisa, (is),a,
                                      person]) &
                              i_attr_i(
                                [lisa,has,an,address,
                                 211]))))),
discourse 1)
        NL =    [[some,persons,are,owners],
                [a,person,can,have,exactly,one,address],
                [a,person,can,have,exactly,one,driving_licence],
                [carl, (is),a,person],[carl,has,a,driving_licence,121],
                [lisa, (is),a,person],[lisa,has,an,address,211]]
```

The example above gives an idea how it would technically be possible to achieve the above proposal and how the discourse tree would look like.

5.Conclusions

Support tools for conceptual modeling lack natural language generation functions. In this paper we have argued for the need of natural language generation as a support tool for conceptual modeling.

The paper proposes a set of appropriate questions which could be posed by the user and a set of suitable natural language discourses to answer these questions. From the proposed discourses a discourse grammar is generalized. The discourse grammar connects a conceptual model to a discourse structure. A natural language generation system built on this grammar is suggested. The goal of the system is to improve the validation of a conceptual model.

The purpose of a natural language discourse is to answer a question to a user in a more satisfying and contextual sensitive way than a single sentence or a set of unordered sentences would. We know that natural language lowers the conceptual barrier for the user and that a natural language discourse gives a better comprehension, since the receiver of the discourse when reading the linear text will try to identify the higher order structure of the text, according to [Ander85], and consequently the conceptual model.

Future research will be to implement the proposed system and to extend the grammar for more cases and then test the system to determine which discourse structures the users require.

I would conclude with: Interpreted data gives information, reasoning about information gives knowledge and knowledge expressed in natural language gives understanding !

Acknowledgements

I would like to thank my advisor Carl Gustaf Jansson and my thesis committee: Janis Bubenko, Carl Brown and Östen Dahl for generously contributing of their knowledge in their fields and for their valuable comments. I would also like to thank Paul Johannesson and Rolf Wohed and others in the SYSLAB research group for interesting discussions which contributed to this paper and also, thank you, Stewart Kowalski for commenting on the English.

References

Ander85 J.R. Anderson: Cognitive Psychology and Its Implications, Carnegie-Mellon University, W.H. Freeman and Company 1985.

Appelt85 D.E. Appelt: Planning English Sentences, Cambridge University Press 1985.

Black87 W.J.Black: Acquisition of Conceptual Data Models from Natural Language Descriptions, In The Proceedings of The Third Conference of the European Chapter of Computational Linguistics , Copenhagen, Denmark 1987.

Boman91 M. Boman et al: Conceptual Modeling, Department of Computer and Systems Sciences, Stockholm University Oct 1991.

Buben84 J. Bubenko et al: Konceptuell modelering - Informationsanalys, Studentlitteratur, Lund 1984, (in Swedish)

Buben86 J. Bubenko: Information System Methodologies - A Research View, SYSLAB Report no 40, Department of Computer and Systems Sciences, Stockholm University, Sweden 1986.

Cauv88 C. Cauvet et al: Information Systems Design: An expert system approach, Proceedings of IFIP, Guangzhou China, 1988.

Cauv91 C.Cauvet et al: ALECSI: An expert system for requirements engineering, in Proceedings of Computer Aided Information System Engineering, CAISE-91, Eds. R. Andersen et al,Trondheim , 1991.

Chen83 P. P-S. Chen: English Sentence Structure and Entity Relationship Diagrams, Information Sciences 29, p.p. 127-149.

Clock84 W.F. Clocksin et al: Programming in Prolog, Springer Verlag 1984.

Dali89 H. Dalianis: Generating a Natural Language Description and Deduction from a Conceptual Schema, SYSLAB Working Paper no. 160, Royal Institute of Technology, Nov 1989.

Dali90 H. Dalianis: Deep generation strategies and their application for creating alternative descriptions form conceptual schemas, SYSLAB Working paper no. 177, Royal Institute of Technology, Nov 1990.

Dali91 H.Dalianis: Generating a Deep Structure from a Conceptual Schema with consideration of a User Model, SYSLAB Working paper no. 184, Royal Institute of Technology, Aug 1991.

DeTro88 O. De Troyer et al: RIDL* on the CRIS case: A workbench for NIAM, Computerized Assistance During the Information Systems Life Cycle.T.W Olle, et al. (eds).Elsevier Science Publishers B.V North Holland, 1988.

Grosz87 B.J. Grosz et al: TEAM: An experiment on the design of Transportable Natural Language Interfaces, J. of Artificial Intelligence, pp 173-243, no 32 1987.

Hobbs85 J.R Hobbs: On the Coherence and Structure of Discourse, Report No. CSLI-85-37, October 1985.

Hobbs90 J Hobbs: Literature and Cognition, CSLI Lecture Notes Number 21, Center for the Study of Language and Information, 1990.

ISO82 ISO Technical Report, Concepts and Terminology for the Conceptual Schema and the Information Base, ed J.J. van Griethuysen, ISO/TC97/SC5 - N 695, 1982.

Johan90 P. Johannesson: MOLOC: Using Prolog for conceptual Modeling, Proceedings of the International Conference on Entity-Relationship Approach, North Holland 1991.

Kuntz89 M. Kuntz et al.: Ergonomic Schema Design and Browsing with More Semantics in the Pasta-3 Interface for E-E DBMSs, Proceedings of the 8th International Conference on Entity-Relationship Approach, Ed F.H. Lochovsky, Toronto, Canada, 1989.

Lloyd87 J.W.Lloyd: Foundations of Logic Programming, Springer-Verlag 1987.

Mann84 W. C. Mann: Discourse Structures for Text Generation, Proceedings of the 22nd annual meeting of the Association of Computational Linguistic, Stanford, CA, June 1984.

Mann88 W.C Mann et al: Rhetorical Structure Theory: Towards a Functional Theory of Text Organization, In TEXT Vol 8:3, 1988.

McKeo85a K.R. McKeown: Textgeneration: Using discourse Strategies and focus constraints to generate natural language text, Cambridge University Press 1985.

McKeo85b K.R. McKeown: Discourse Strategies for Generating Natural Language Text, Artificial Intelligence, vol 27 no 1, Sept 1985.

Paris85 C. Paris: Description Strategies for naive and expert users, Proc. of the 23rd Annual Meeting of the Association of Computational Linguistics 1985.

Paris88 C. Paris: Tailoring Object's descriptions to a User´s Level of Expertise, J. of Computational Linguistics, Vol 14, No 3, Sept 1988.

Pereir80 F.C.N Pereira et al: Definite Clause Grammars for Language Analysis - A Survey of the Formalism and a Comparison with Augmented Transition Networks. J. of Artificial Intelligence 13, 1980, pp 231-278.

Tauzo89 B. Tauzovich: An Expert System for Conceptual Data Modeling, Proceeding of the Entity Relationship Approach Toronto, Canada, 1989.

Woh88 R. Wohed: Diagnosis of Conceptual Schemas, SYSLAB report no 56, Department of Computer and Systems Sciences, Royal Institute of Technology and Stockholm University 1988.