Centrum voor Wiskunde en Informatica
Centre for Mathematics and Computer Science

M.L. Kersten, H. Weigand, F. Dignum, J. Boom

A conceptual modeling expert system

Department of Computer Science/Algorithmics & Architecture Report CS-R8518 September

Bibliotheek
Centrum voor Wiskunde en Informatica

The Centre for Mathematics and Computer Science is a research institute of the Stichting Mathematisch Centrum, which was founded on February 11, 1946, as a nonprofit institution aiming at the promotion of mathematics, computer science, and their applications. It is sponsored by the Dutch Government through the Netherlands Organization for the Advancement of Pure Research (Z.W.O.).

Copyright © Stichting Mathematisch Centrum, Amsterdam

A Conceptual Modeling Expert System

M.L. Kersten

Centre for Mathematics & Computer Science
Kruislaan 413
1098 SJ Amsterdam

H. Weigand, F. Dignum, J. Boom

Dept. of Mathematics and Computer Science
Vrije Universiteit
de Boelelaan 1081
1081 HV Amsterdam.

ABSTRACT

This paper describes the architecture of an interactive conceptual modeling expert system, called ACME. The input to ACME is a natural language description of the application domain, which is decomposed by a parser into so-called predications. From these predications a preliminary EAR model can be extracted readily, which is subsequently improved by the EAR modeling expert using structural and semantic rules stored in a knowledge base. In an iterative process the user resolves the inconsistency and ambiguity problems discovered by ACME and enhances the conceptual model.
ACM Categories and Subject Classifications: H.2.1 [**Database Management**] Logical design; I.2.1 [**Artificial Intelligence**] Expert Systems; I.2.7 [**Artificial Intelligence**] Natural Language Parsing.

69 H21
69 K11
69 K17

1. Introduction

The design of a conceptual model is the most crucial part in the database life-cycle. Since an incorrect conceptual model often leads to incorrect database schema or requires many expensive changes when the database has been build. The prime causes underlying an incorrect conceptual scheme can be classified as follows:

"Lack of domain knowledge." In many cases database designers lack domain knowledge needed to perform the task. Traditionally this shortcoming is resolved by interviews.

"Complex application domain." Most real-world databases are characterized by many entities, relationships, and user views. Structuring this information forms a major problem even for experienced model builders.

"Lack of modelling expertise." The users which have domain knowledge are in general poor modelers.

These problems suggest that there is a growing need for practical automated tools supporting the process of conceptual model design, to handle complex domains for experts, and to guide inexperienced designers in this task. The ACME system described in this report is such an expert system for conceptual modelling. It uses natural language for man-machine interaction and thereby aims at supporting the design by end-users. In effect, ACME processes a raw natural language

Report CS-R8518
Centre for Mathematics and Computer Science
P.O. Box 4079, 1009 AB Amsterdam, The Netherlands

document describing the relevant Universe of Discourse. By means of linguistic analysis ACME synthesizes a draft conceptual model in the form of an extended EAR description [Chen]. The result is handed to the conceptual model builder for analysis and refinement. It uses a knowledge base to resolve with the aid of the user inconsistencies , ambiguities, and incompleteness in the preliminary model to arrive at a satisfactory conceptual model. The use of formalized data modelling rules aids the expert users in keeping track of the intricacies of the application domain being modelled. The linguistic basis makes it easy to paraphrase the model in natural language again, thereby greatly simplifying the documentation aspects.

ACME is not a panacea for understanding and modelling ill-defined application domains, nor uses common knowledge to aid in the modeling process. It is aimed primarily at the construction of an acceptable conceptual data model for common environments. However, we anticipate that the system can be extended with domain specific knowledge for classes of systems. For example, it is conceivable that the concept of an employee can be described to a great extent in a knowledge base.

The design of ACME is influenced, among others, by the conceptual model design methodologies ISDOS, DATAID-1 and NIAM. The original aim of ISDOS [Teichrow] was to provide a system that could bridge the gap from logical design to information system. Currently, the ISDOS system is mostly used as a documentation tool during the initial system development stages. It has its own specification language (PSL) and a statement analyzer (PSA), which includes various report facilities. The University of Namur has developed a similar software package, called IDA [FNDP], which, besides supporting system documentation, can simulate proposed systems and can generate prototype systems from the specification.

NIAM ([Nijssen], [Griet]) is actually a design methodology. NIAM pays special attention to the initial steps of the design process, with emphasis on the role of natural language processing. NIAM is supported by a software package, called ISDIS [VeBe]. ISDIS is a documentation system, which checks the specification on correctness and consistency, and which can generate database schemata.

Our approach to automating the process of designing conceptual models can best be compared with the DATAID project [Ceri]. DATAID-1 [DeAn] is an elaborate database design methodology including automated tools to support the information system designer. In the first stage of the development process the designer collects functional descriptions of the future system in the form of natural language text fragments. These fragments are filtered and rewritten (manually) into a restricted language satisfying suitable conventions. In the next stage, these descriptions are used to derive the data, operation, and event glossary. The various glossaries are combined and transformed into a global schema, or conceptual schema, which forms the basis for the logical database design. The last step is the derivation of the physical database design from the conceptual schema, e.g. a relational schema.

ACME differs from DATAID in three important aspects. First, in our system we have implemented a new approach towards natural language parsing. Second, the sentences are analysed automatically, i.e. the information for conceptual modelling is classified by means of heuristic rules. Third, emphasis is placed on deriving a good view from each user. This objective is achieved by applying many of the view integration techniques as early as possible within the design process and allow the user to iteratively enhance (by means of English prose) its description.

In this paper, we present an overview of the ACME approach. In section 2, the overall architecture of ACME is explained. Section 3 describes the linguistic part of the system. Section 4 presents the heuristics used to obtain the preliminary model, whereas the conceptual model builder expertise is discussed in section 5. Finally, in section 6, we discuss some findings, suggest advantages and limitations of our approach and indicate directions for further research.

2. Architecture of ACME

The task of ACME is interactively develop a conceptual model for a database using English text as input. This task is divided into three subtasks: text analysis, model synthesis and model refinement. An architectural overview of ACME is given in Figure 1.

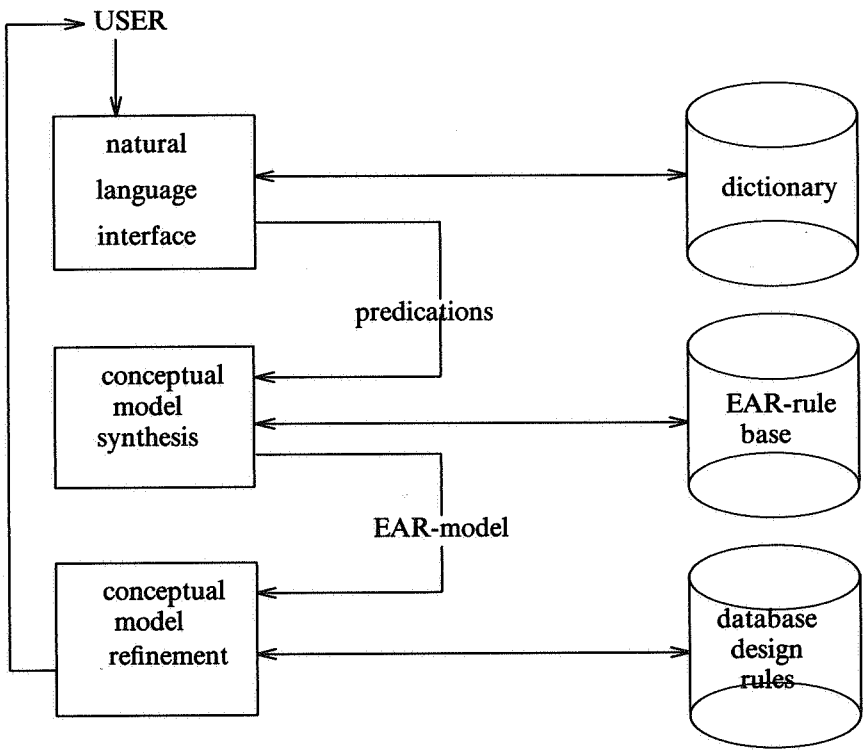


Figure 1 The architecture of ACME

The natural language parsing process analyses an English text taking into account the dependencies between sentences and generates a set of so called *predications*. A predication is a surface-independent representation of the syntactic and semantic content of a sentence. This term is borrowed from the theory of functional grammars. The structure and use of predications is described shortly.

The model synthesis task interprets and combines the predications to obtain a preliminary extended EAR model of the database. The following extensions to the basic EAR model are used in the XEAR model of ACME:

(1) Entities are arranged in an *entity-type hierarchy* [Smith]. In particular, two generic entity types are recognized *person* (animate) and *object* (inanimate).

(2) Relationship which represent a change in the universe of discourse are recognized as *events*. Each event is characterized by a set of participant roles, event name, trigger relations, and pre- and post- conditions on the database state.

(3) Characteristic value sets are recognized and replaced by a *classification* attribute.

The model refinement task applies a set of heuristic rules to check for completeness, ambiguity, and structural inconsistency of the conceptual model. The findings are reported to the user, who enters an iterative cycle to enhance the model and to resolve the problems. The outcome of the whole process is a list of entities, (named) relationships, and events with pre- and post-conditions. Each task is subsequently described in more detail.

3. The natural language analysis process

Much of the work done on natural language processing aims at full understanding of the utterances [Win80]. Various representation schemes have been suggested for this purpose, such as frames, schemata, and semantic networks. Full understanding presupposes a (universal) model of the application domain and extensive common world knowledge. This assumption does not hold for ACME. It should support the construction of models for many application domains. On the other hand, early text understanding systems such as PARRY or ELIZA [Weiz] do a rather superficial job by scanning sentences for keywords or selected patterns occurrences. Obviously, such an approach is too restrictive for the task ahead.

Our approach towards text understanding can be characterized as *semiotic*. The semiotic paradigm suggests that the meaning of the individual signs is determined primarily by their role in the text structure. It can be simplified to the following principle:

Semantics deals with the way word signs are interrelated. The interrelating text structure reflects the authors' view of the universe of discourse.

We hypothesize that this kind of analysis is most appropriate for our purpose, for it avoids the need to "deeply understand" the text. It abstracts from the way information is syntactically presented. Our research shows that a limited set of text structures suffices to capture the essence of communicating conceptual modelling information. The immediate practical consequence is that the natural language processor's dictionary contains linguistic categories of words only (as in [Barn]). Moreover, the language syntax is devoid of semantic interpretation directives.

The actual implementation of the language parser is based on the combined use of ATN's and Functional Grammar expression rules. Functional Grammars ([Dik], [Win83]) is a rather young linguistic theory. One of its attractive features is the claimed functional dependence between syntax, semantics and pragmatics. Functional grammars (FG) offer a framework to describe these three levels in a clear formalism. A combined approach is more efficient than working with a 'pure' FG approach, because the structure of English heavily relies on word order. Therefore the FG expression rules for English can be represented by means of ATN's or context-free grammars to a large extent. For other languages this may not be the case.

The natural language parser is split into four modules, as shown in Figure 2. Basically each module represents one stage of the text understanding process. Yet for complex sentences iteration may be required. Moreover, the current implementation handles declarative sentences only, imperative or interrogative sentences are omitted for the time being.

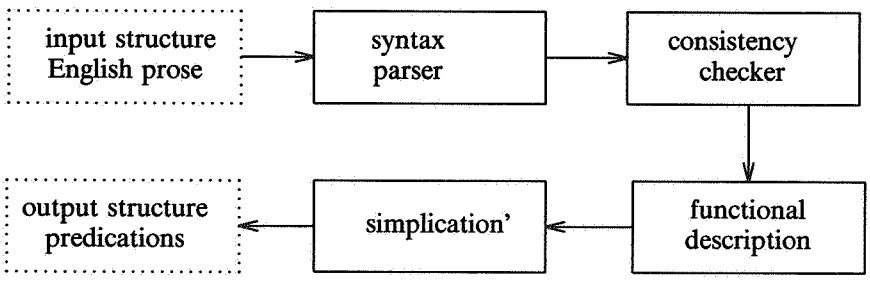


Figure 2 The natural language parser

The first module is driven by the ATN and identifies the clauses, such as noun groups, verb phrases, and preposition groups. Subsequently phrases are combined into higher level clauses according to a syntactic frame. The output of the module is a clause tree. The leaves represent the lexical entities obtained from the lexicon. The categories of words not found in the lexicon are guessed based on their position in the sentence. An exception is made for unknown verbs, since they play a crucial role in determining the function structure of the sentence. We expect that the user is able to provide a known synonym or that he extends the dictionary with the attributes for the new verb.

The second module checks the consistency of the clause tree. In the prototype implementation of ACME these checks are limited to verb agreement, transitivity agreement, and number agreement. Verb agreement checks whether the verb person and number agree with the subject. The transitivity property of a verb is found in the lexicon. It is either non-transitive ("I walk"), transitive ("I read a book"), or bi-transitive("I give the book to the girl"). The transitivity check assures that the required subject and object components have been recognized. Number agreement tests for compliance of the quantifiers (singular and plural) with the stem of the corresponding noun form.

The third module models the Functional Grammar approach where semantic or functional roles are assigned to the clauses. Our approach is similar to the one taken in [Win83][Dik]. The distinguishable sentence categories or process types are:

- Material Process (e.g., "he wrote a book")
- Mental Process (e.g., "he saw the book")
- Verbal Process (e.g., "she told him what happened")
- Equative Process (e.g., "John is Hamlet")
- Attributive Process (e.g., "the library contains 2000 works")

The process type is fully determined by the verb phrase, homonyms not taken into account, and characterized by a set of semantic roles. This information is obtained from the dictionary. For example, a Material Process has an *agent* and if the verb is transitive a *goal*. In the sentence "he wrote a book", "he" is an *agent* and "book" is a *goal*. The result of this stage is a predication where each term is assigned a syntactic and semantic role. The predication framework is shown in figure 3 using a BNF notation. Component values not spelled out completely can be determined from the lexicon.

The last module simplifies the predications. In particular, pronouns are replaced by their corresponding noun groups on the basis of simple heuristic rules, conjunctive sentences are split up into two simple sentences, and embedded clauses are put into separate predications (leaving a pointer behind). As a result, all predications are one-level and contain only one main verb.

```
()          is used to group components
[]          indicates an optional component
[]*         this component can occur zero or more times
[]+          this component can occur one or more times
a|b|c       this component is either a, b or c

Predication ::= Modifier Subject Verbphrase Object Modifier.

Verbphrase  ::= [Modal] Verb Tense (active|passive)
                    Trans Process (positive|negative).

Subject     ::= Noungroup Role.

Object      ::= Noungroup Role.

Attributes  ::= [Sort_adjective]* (definite|indefinite)
                    Quantifiers (singular|plural) (first|second|third)
                    (animate|inanimate) Relations.

Quantifiers ::= [Predets] | [Ordinals] | [Cardinals].

Relation    ::= [ISA Noun] [poss (Noungroup|Pronoun)] [Sort]
                [restrict (Lexicon|Prepgroups)] [or Noungroup]*.

Noungroup   ::= ([Adjective]* [Noun]+ Attributes) | Pronoun.

Sort        ::= Sort_adj (equate|compare|super)
                    (positive|negative) (Noungroup|Lexicon).

Modifier    ::= (Lexicon Role) | ([Sort] [Prep (Noungroup|Pronoun) Role]+).

Prepgroups  ::= [Prep Noungroup]+.

Tense       ::= present | infinite | past | pastpart | prespart.

Trans       ::= intransitive | transitive | bitransitive.

Process     ::= material | verbal | attribute | equative | mental.

Role        ::= agent | medium | beneficiary |
                    sayer | verbalization | addressee |
                    carrier | attribute |
                    identifier | identified |
                    cognizant | phenomenon.

Lexicon     ::= /* The actual words in the sentence */
```

Figure 3 The predication frame

The construction of a predication is illustrated by a text fragment taken from the example in the appendix. For example, consider the sentence:

> *"... The adherent can borrow or return one or several works by applying to one of the library wickets. ..."*

The linguistic analysis of ACME comes up with the following predications (in PROLOG list notation):

```
[
/*modifier */    [],
/*verbphrase */ [[can,borrow], infinite,active,bitransitive,material,positive],
/*subject */    [[adherent], [],definite,[],singular,third,animate,[],agent],
/*object */     [[work], [],indefinite,[one],plural,third,inanimate,
                [[or,[[work],[], quantified,[several],plural,third,inanimate,[]]
                  ], medium],
/*modifier */   [[by,applying,to,one,of,the,library,wickets], sentmod]
]

[
/*modifier */   [],
/*verbphrase */ [[can,return], infinite,active,bitransitive,material,positive],
/*subject   */ [[adherent], [],definite,[],singular,third,animate,[],agent],
/*object */     [[work], [],indefinite,[one],plural,third,inanimate,
              [[or,[[work],[], quantified,[several],plural,third,inanimate,[]]
                  ], medium],
/*modifier */   [[by,applying,to,one,of,the,library,wickets], sentmod],
]
```

4. The conceptual model synthesis process

In the second phase ACME extracts a preliminary EAR model from the predications. Although these predications provide useful information for the EAR model synthesis, the material is still fuzzy and incomplete. Therefore we choose a knowledge based approach, looking for a set of simple heuristic rules that work in many situations. Other approaches, such as NIAM or DATAID-1, incorporate such heuristics in their cookbook or methodology as well. Our heuristics can be regarded as a formalization of this kind of "designer's wisdom" under a linguistic (FG) interpretation.

As explained before the EAR model has been extended to capture more information and to simplify the transformation of prose into the model. In particular, we use objects, entity-type hierarchies, type-attributes, and events. *Objects* represent lexical objects which have not been classified as either entities, attributes, relations, or events earlier in the synthesis process. They are recognized as important linguistic tokens for the model under construction only. The *entity-type* hierarchy models ISA relationships. *Type-attributes* are used to label 'semi-identical' entities. For example, the entities NOVELS, SF, and DETECTIVE can be described as subtypes of the entity book, or can be represented as a single type-attribute of the entity book with the value set (NOVELS,SF,DETECTIVE,BOOK). The *events* represent recognizable actions possibly constrained by pre- or post-conditions. The EAR concepts can be summarized in the hierarchy shown in Figure 4. The arrows indicate the partial relationship between the concepts which serves the model refiner process in resolving ambiguities.

As the task of the model synthesizer is to generate a preliminary model using information in the predications only. This means that little contextual information is used and that ambiguous information may be generated. It is the task of the model refiner to resolve the problems.

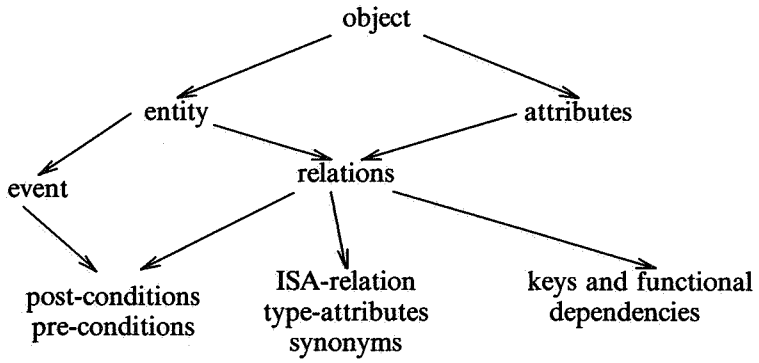


Figure 4 XEAR concept hierarchy

The first step in the synthesis process is to identify the objects which play a role in the conceptual model and to classify them as either entities or as attributes. A simple heuristic for finding objects is to consider each noun in the predications as such. Because they denote a real-world concept.
When an adjective is applied to a noun we have to deal with two objects, i.e. the noun and the adjective+noun, because the latter can be considered a refinement of the former. Therefore, the adjective+noun is considered an entity as well and an ISA relation is defined (ISA(noun+adjective,noun)). The model refinement process may turn this adjective into an attribute of an entity.

The heuristic to find relations and events is to use the process type of the sentence and the verb classification. Figure 5 shows the table to derive them. Whenever the process type is material or verbal we consider the sentence an event specification. Because in that case the sentence expresses a (dynamic) change of the world. The Agent of an event is always an entity (e.g. a person). All other verbs give rise to static relations qualified by two entities or an entity and an attribute.

	transitivity		
process	intransitive	transitive	bitransitive
attributive	EA-rel	EA-rel	EA-rel or EE-rel
material	event	event	event
verbal	event	event	event
equative	*	EE-rel	*
mental	-	-	-

EA-rel := entity - attribute relation
EE-rel := entity - entity relation
* := this combination appears seldomly
\- := ignored

Figure 5 Finding relations and events

In general, attributive process verbs indicate named relationships between the Subject and the Object. An exception is made for the verbs "to be" and "to have". In most cases these words describe a primitive relation, such as possession ("the book is in the library") and the generalization hierarchy ISA ("an adherent is a person"). In all other cases a relationship is suggested that is not concretely specified. For example, although "the library has 750 adherents" can be interpreted as an integrity constraint on the number of elements in a set, it does not indicate the role of an adherent in relationship with the library(it is not a possessive). Therefore, the synthesis process generates a 'nameless' relation. The model refiner obtains the missing information from the user later on.

Relation cardinalities are obtained from the quantifiers or the singular/plural attribute in the predications. The default rule is to declare all transitive event relations to be 1-n (single agent, multiple goal). Otherwise the relation cardinality is n-m.

A formal representation of the pre- and post-conditions of events is more involved. Preconditions may be expressed in many different ways, such as *if-clauses*, *relative-clauses*, and *deontic sentences*. Example are "the adherent has the possibility to reserve a work if none of its copies are available" and "the adherent can not renew the loan of a work that has been reserved". Deontic sentences, such as "he can not dispose of more than 3 library works at the same time", must be analyzed deeper before they can be mapped into a formal pre-condition. In this example we should find out what makes someone to dispose of more than three books. The answer must be the action borrow, repeated more than three times without returning any book. So the rule can be reformulated as a precondition of the action "borrow". However, the text is not sufficient to come up with this automatically. We expect that even if a more detailed dictionary is available, user assistance on this matter remains necessary.
In the current implementation of ACME, we apply the following heuristic for conditions; if a clause represents a material process and is modalized (with "can", "has possibility", or "may"), then *if-clauses* or *relative-clauses* are interpreted as pre-conditions and stored as phrases in the event relation. This ensures that all information is kept together. Although its use will be limited to documentation of the initial design.

Construction of the type hierarchy, the ISA-relations, is supported by two heuristic rules. First, if the verbphrase is "be", the sentence is active, and the subject role is *identifier* than we conclude that there exists an ISA relation between the subject noun and the object noun (e.g. "a detective is a book"). In that case, the subject is a specialization of the object. Second, the other ISA relations are obtained from the predications directly, because they are recognized by the parser as such already. In particular, if a noun stands in agent position of a material or verbal process then it is considered a subtype of the basic type *person*. Similar, a noun that is only used in goal position and is not animate -which can be determined from the dictionary- is considered a subtype of *object*.

Some additional rules are applied for noise reduction. For example, in mental process sentences the verbs are ignored, because they indicate irrelevant information for the modelling process. Note, however, that they are scanned for entities and objects using the previous heuristics. Another example, irrelevant sentences are determined and skipped. That is, if a predication contains the personal pronoun I/we, then the predication can be omitted. This rule deletes the first predication in the example application in the appendix ("we have a library..").

The model synthesis process currently implemented is by no means complete. As indicated above pre- and post-conditions as well as (arithmetic) integrity constraints are not fully explored yet. For example, "the average number of books" and similar aggregate operations can be abstracted from and recognized easily. Moreover, currently predications are analysed one at a time with little concern about their context or the application domain. Both can be accommodated by extending the knowledge base.

5. The conceptual model refinement process

The last phase in the generation of the XEAR model is the analysis of the intermediate model, to resolve ambiguities, and to optimize it. This process necessarily incorporates the active participation of the user to reach a satisfactory result, because the natural language presentation analysed so far is likely to be imprecise and incomplete. The refinement approach taken is hierarchical, based on the concept tree depicted in figure 4. That is, the ISA-relation hierarchy is not analysed before all entities are known. Similarly, synonyms are located after the ISA-hierarchy is established. The drawback of this approach is a seeming lack of analysis concentration. An alternative would be to apply a focussed approach, where the analyser starts with a single concept and resolves all related problems before moving on to the next. Such an approach, though attractive, hampers from the assumption that to resolve problems the related information should be complete and consistent.

The refinement rules currently implemented in ACME are classified as follows:

- classification rules
- disambiguity rules
- synonym/homonym rules
- structural consistency rules

- optimization rules
- database schema rules

Each of these topics is discussed shortly.

One of the first concerns is to uniquely classify each object and each nameless relation. Most objects are classified as entities or attributes based on the sentence structure already by the model synthesizer. The few leftovers are returned to the user for clarification. Nameless relations, introduced by the synthesizer, are returned as well. The system expects that each can be given a proper verb name. Otherwise, they are dropped.

Ambiguity results from classifying objects in a local context only. The heuristic applied in resolving such problems is based on the order Attribute -> Entity -> Relation name. An ambiguous object is classified as low as possible. To enforce this rule, nameless relationships may have to be introduced. For example, the synthesis process erroneously determines "work" as both an attribute of "library" and as an entity "work". The solution generated is to declare "work" as an entity and define a 1-n relation between "library" and "work". In this particular case the nameless relation generated can be assigned the name "contains". Similar rules can be defined for handling the other cases.

Ambiguity can also result from naming relations. Checks applied by ACME are to recognize duplicate and inconsistent definitions of the same relation. Finally, a graph analysis may turn up transitive use of relation names. For example, the following set of relations contain one superfluous relationship:

a library contains works
a library contains book shelves
a book shelf contains works

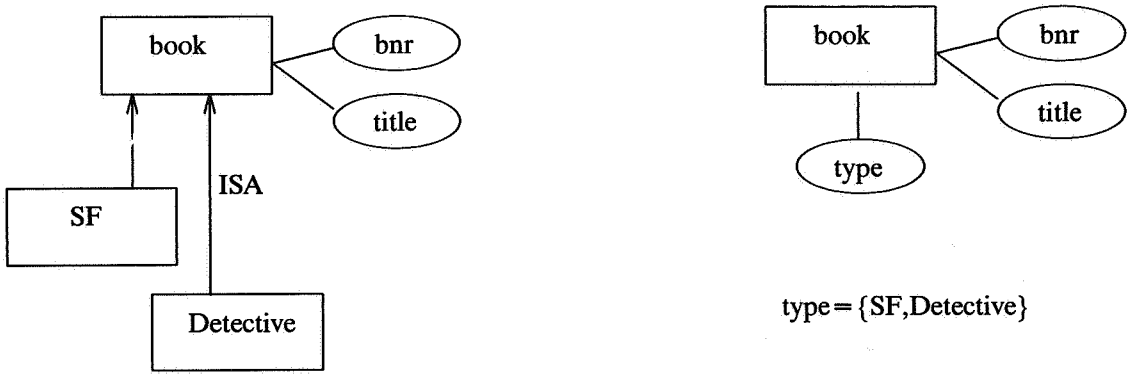
The generalization hierarchy provides a useful concept in modelling information. It allows compact description of inheritable attributes. However, the derivation of ISA relations from text may result in a series of anomalies. For example, a cycle in the ISA relations indicates that either all entities on the cycle are identical, that they are variations of a single entity type, or that there is an error in entity naming. If attributes are associated with only one entity on the cycle then we may safely conclude that they are synonyms. If the attribute sets differ per cycle element then the user should resolve this problem.

Recognition of synonyms is nearly impossible, because it involves the semantics associated with the entities. However, potential synonyms can be indicated by comparing entities and their attribute sets. When two attribute sets have many elements in common then either they constitute synonyms or they can be placed in a ISA relation (which results in a nameless relation).

The next set of refinement heuristics models structural consistency. That is, they check for desirable properties of the XEAR representation without concern about the semantics. The first rule to check is connectivity of the EAR graph (ignoring the two generic types *person* and *object*). If the graph is not connected than certainly something is amiss. Because then we may assume that two partial descriptions are given of the real world being modelled. Either this is resolved by telling what entities are synonyms or by defining new relations to connect the components.

A variation of the connectivity rule is that all entities should occur in the ISA hierarchy. If an entity does not satisfy this criterium then one does not know whether the entity is an *object* or a *person* either. This can be considered an undesirable situation.

Another major cause of structural inconsistency is incompleteness of the description. For example, each entity should have (key) attributes. When entities after questioning the user remain attributeless and still play a role in a relation then they can be transformed into a type_attribute; probably associated with the relation. Similar rules can be defined for handling ISA hierarchies with loose tails or heads. For example, the following ISA relation hierarchies are probably equivalent.



The rules for event refinement are incomplete, due to the complex nature of their textual occurrence. The minimum requirements checked are; every event predication is fully specified; all argument positions are classified; goal-positions are specified to be either affected or effected goal. Moreover, every entity type has one or several event types in which the entity type is in the effected goal position. This rule assures that every new entity comes into being by means of a foreknown event.

Another class of structural consistency rules is based on complexity metrics. For example, an excessive amount of attributes per entity or many relations between two entities can both be considered signals of bad design. This information is returned to the user for clarification.

The optimization and database schema generation rules, as applied in [SECSI], haven't been implemented yet.

6. Summary and conclusions

Our approach towards automated intelligent database design tools effectively uses functional linguistics to derive Entity-Relationship models for a wide class of applications. The Functional Grammar framework enables us to make the intuitive correspondence of verb usage and relationships in the data model more explicit. Every relationship is placed in the verb process hierarchy described above. The semantic roles associated with each verb frame delimit the set of possible links between relationships and entities. An alternative interpretation of material processes as event structures will integrate the static and dynamic part of the model in an interesting way. In particular, it draws attention to the time attribute of relationships, making a link with the research on historical databases. What we call an event corresponds with the *delta* or difference from [DaLW].
The mapping of nouns to entities shows, among others, that the notion of entity is in no way absolute. The effect of *nominalization*, i.e. transforming a verb into a corresponding noun, is that relationships can be transformed, by means of some semantic mapping, into an entity. It is an interesting point of study whether the ER model (and its implementation !) can support this kind of relativism.
Finally, the ER model seems to give priority to the entities, rather then to the relationships. The importance of verb usage in our analysis seems to suggest an opposite priority.

The advantages of ACME are summarized as follows. ACME gives the designer active and intelligent support during the initial stage of conceptual model development. A full bag of relevant questions and design heuristics are available in a knowledge base. This way the designer is better equipped to analyse complex environments and gets a substantial part of the model documentation as a side effect for free. Users become more involved in the design process, because *their* own words are at the bottom of the design, not the designer's interpretation. By means of a paraphrase program (not described in this paper), the conceptual model can be expressed in natural language, which we expect to improve the communication and cooperation between users and designers.
From a theoretical point of view the linguistic paradigm offers new insights in the field of database semantics. This is not surprising, because natural language happens to be a rather effective data description and communication language for many ages. We are convinced that such a linguistic

approach is a fundamental prerequisite to make the transition to Fifth Generation Database Systems.

The current version ACME supports a limited grammar and dictionary, as is inherent for a prototype system. The static part of the model (entities and relationships) is handled by ACME surprisingly well. This is not the case with the dynamic part (preconditions and triggers), where the shortcoming partly stems from a lack of good intersentential relations in our linguistic framework, making it an area for further investigation.

A prototype version of ACME is implemented in C-PROLOG under UNIX on a VAX 750. As far as the conceptual language is concerned, our ER model should be extended to capture more semantics. In particular, the role of quantifiers, deontic sentences, and time relations. Experience shows that the knowledge base of heuristic rules can be extended by analysis of more large scale examples. Finally, ACME should be incorporated in a package of software tools including the design and analysis of the information flow model, the generation of prototype database schemata, and the automatic development of intelligent user interfaces.

Bibliography

[Barn] Barnwell, K.G.L., Introduction to Semantics & Translation, Summer Institute of Linguistics, High Wycombe, 1974.

[Ceri] S. Ceri (ed.), Methodology and tools for data base design, North-Holland, 1983.

[Chen] Chen, P.P., The Entity Relationship Model, Toward a unified view of data, ACM TODS 1, 1976.

[DaLW] Dadam, P., Lum. V, Werner, H.-D., Integration of Time Versions into a Relational Database System, Proc. of the Tenth Int. Conf. on Very Large Data Bases, Singapore, 1984.

[DeAn] De Antonellis, V., Di Leva, A., DATAID-1: A Database Design Methodology, IEEE Tuturial Notes 3, 1984.

[Dik] Dik, S.C., Functional Grammar, North-Holland, Amsterdam, 1978.

[FNDP] DSL/DSA manual de reference, Institut d'Informatique, FNDP, Namur, 1982.

[Griet] Griethuysen, J.J. van (ed), Concepts and Terminology for the conceptual schema, ISO/TC97/SC5-N695, 1983.

[LeLu] Leonard, M. and B.T. Luong, "Information System Approach, Integrating Data and Transactions", Proc. VLDB, 1981.

[Nijssen] Nijssen, G.M., A framework for advance mass storage applications, Proc. IFIP MEDINFO, North-Holland, 1980.

[Smith] Smith, J., Smith, D., Database Abstractions: Aggregation and Generalization, ACM TODS 2, 1977.

[Teichrow] Teichrow, D., Sayani, H., Automation of system building, Datamation, August 15, 1971.

[VeBe] Verheijen, G.M.A., Bekkum, J. van, NIAM: An Information Analysis Method, in: Information Systems Design Methodologies, editors TW Olle, AA Verrijn-Stuart, North Holland (1982).

[Weiz] Weizenbaum, J., ELIZA - a computer program for the study of natural language communication between man and machine, Comm of the ACM 9, 1966.

[Win80] Winograd, T., What does it mean to understand language?, in: Cognitive Science 4, 1980.

[Win83] Winograd, T., Language as a Cognitive Process (Vol. 1: Syntax), Addison-Wesley, 1983.

[Wood] Woods, W.A., Transition network grammars for natural language analysis, Comm of the ACM 13, 1970.

APPENDIX

The following is a brief example adapted from [LeLu], which can be handled by ACME.

We have a library which contains 2000 works and 750 adherents. Each work is identified by a shelfmark. The average number of copies per work is three.
The adherent can borrow or return one or several works by applying to one of the library wickets. He also has the possibility to reserve a work if none of its copies are available. In that case, his reservation is placed at the end of the other reservations made for the same work. As soon as a copy becomes available, the adherent is informed of it and he has a whole week to come and borrow the work.

The adherent cannot dispose of more than 3 library works at a time and each loan has to be returned at the end of 3 weeks. The adherent cannot reserve more than 3 works at the same time. Moreover, a loan of a work can not be renewed if this work has been reserved.

Due to lack of space the predications generated from this text are not reproduced. The output of the model synthesis process is the following list of observations:

```
object(library).
object(work).                 object(adherent).
object(shelfmark).  object(copy).
object(possibility).  object(reservation).
object(week).                 object(library_work).
object(loan).                 object(to_renew_the_loan_of_a_work).

person(adherent).

attribute(shelfmark).
attribute(possibility).
attribute(week).

entity(library,[],[adherent, work]).
entity(work,[shelfmark, []).
entity(adherent, [] [week, possibility]).
entity(copy, [], []).
entity(reservation, [], []).
entity(library_work, [], []).
entity(loan, [], []).
entity(to_renew_the_loan_of_a_work, [], []).

event(adherent, borrow, work, by_applying_to_one_of_the_library_wickets,nil).
event(adherent, return, work, by_applying_to_one_of_the_library_wickets,nil).
event(adherent, inform, nil, a_copy_becomes_available, nil).
event(adherent, dispose, nil, no_more_than_3, nil).
event(adherent, dispose_of, work, no_more_than-3, nil).
event(adherent, reserve, work, more_than-3, nil).
event(adherent, allow, to_renew_the_loan_of_a_work, if_this_work_has_been_reserved, nil).
event(reservation, place, nil,nil,nil).
event(reservation, make, nil,nil,nil).
event(copy, become, nil, available, nil).
event(nil, return, loan, at_end_of_week, nil).
event(nil, reserve, work, nil,nil).
event(nil, make, reservation, less_than_5, nil).

relation(copy, average_number_of, work, 3,1).
relation(library, contain, work, nil, 2000).
relation(library, nameless_1, adherent, nil, 750).
relation(adherent, borrow, work, nil, nil).
relation(adherent, return, work, nil, nil).
relation(adherent, nameless_2, possibility, nil, nil).
relation(adherent, nameless_3, week, nil, nil).
relation(adherent, reserve, work, nil,  more_than_3).
relation(adherent, allow, to_renew_the_loan_of_a_work, nil, nil).
relation(work, posses, copy nil, nil).
relation(reservation, make_for, work, nil, nil).

isa(library_wicket,wicket).
isa(library_work, work).
```

The model refinement process reports ambiguities and inconsistencies to the user. These problems are interactively resolved. (This tedious communication process is left out here)