Entity Relationship Approach to the Generation of Sentences for Database Queries

Emmanouil J. Marakakis

A Major Technical Report

in

The Department

of

Computer Science

Presented in Partial Fulfilment of the Requirements
for the Degree of Master of Computer Science at
Concordia University
Montréal, Québec, Canada

June 1984

ABSTRACT

Entity Relationship Approach to the Generation of Sentences for Database Queries

Emmanouil Marakakis

This research work presents the design of a text generator system. The system is capable of generating multisentence text as response to queries stated in a subset of the well known SEQUEL query language intended for relational databases. A relational database designed for an university environment has been used as an example to present the various algorithms of the text generator. As part of the research work, a SEQUEL query language processor has been implemented for the relational database managment system RISS. A data dictionary which contains data about the data in the database and a lexicon which has linguistic information are also created as part of the design process. Data dictionary and the lexicon are stored as relational tables. The third knowledge source used for sentence generation is the semantic information. In this work semantic information is represented as entity networks and relationship networks following the lines of Entity - Relationship model of P.Chen.

A Transformational grammar is used to generate a proper deep structure of the sentence(s) to be generated. It requires firing of the appropriate phrase structure rules and insertion of the proper lexical items. In our system a set of algorithms, based on the categorization of the input queries, are used to generate the deep structure of the sentences to be generated. In order to decide upon the structure of the sentence to be generated, the algorithms take into consideration the structure of the query. Additionaly, they make use of the entity and relationship networks and the data dictionary. Moreover for the generation of the deep structure of sentences the database and the lexicon are used. Next the transformational rules are applied in order and the deep structure is transformed into a surface structure representation. At the end, the results from the database search are inserted at proper places and words are arranged suitably in the surface structure to obtain the final form of sentences.

DEDICATION

I dedicate this research work to the memories of my father John, my sister Helen, my godmother Irene and my first cousin Zaharia. They died while I was student in Canada.

ACNOWLEDGEMENTS

I would like to express my sincere thanks to my supervisor Dr. T.Radhakrishnan. His guidance, advice and encouragement throughout this research work were invaluable. His financial support through the FCAC grant provided by the Government of Quebec allowed me to surpass the deterrent of differential fees which I had to pay as a foreign student.

I also would like to thank the families of my sister Olga, my brother Zaharia and my sister Maria. Their financial and moral support during my studies made my stay in Canada enjoyable.

Last, but not least, I would like to thank my mother for her devotion to her children.

TABLE OF CONTENTS

LIST OF FIGURES

I. INTRODUCTION

Originally data bases were accessed by users through programs written in a programming language; later English like query languages such as SEQUEL, QUERY BY EXAMPLE and ALFA were developed. The query languages are easy to learn and consequently more people can access a data base. Yet, casual users of computers find it difficult to learn a formal language. For these people the best way to communicate with a computer is to use a natural language. As a result several systems [16, 23, 24, 27, 33, 34, 43, 45] have been developed to provide a natural language interface to a data base. Some of these systems [16, 27, 33, 34, 45] are highly dependent on the data base and have been developed for specific applications. Such systems have achieved a high level of performance in their area of expertise. There are systems [23, 24, 43] which are data base independent and they are portable. They can be transferred from one application to another without major changes. The domain of discourse for these question - answering systems is the content of the data base. Most of these systems provide adequate syntatic as well semantic coverage of English in their restricted domain of discourse. Yet, none of them provides a 100% natural language interface to a data base. They also obey the conversational principles observed by native speakers of the language. Such principles are:

a) The assumption of certain common knowledge between the two participants.

b) The assumption of a common natural language.

c) Control passes back and forth between the participants, both making statements and asking questions. Some utterances are complete sentences and others are just phrases or words.

d) The dialogue takes place in context. This aids in the interpretation of words and phrases that would be meaningless or ambiguous if uttered in isolation.

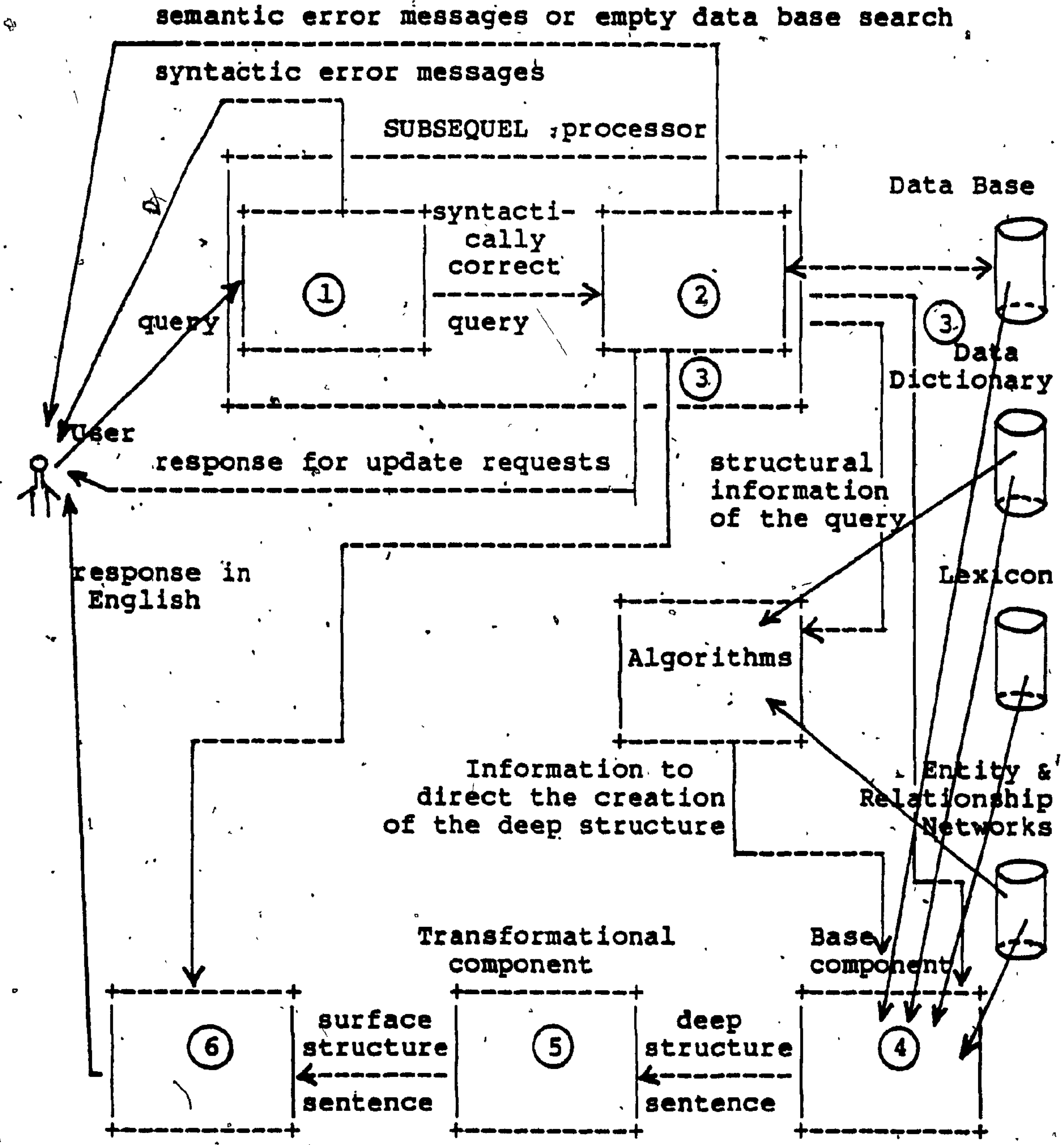
The natural language has its own disadvantages to serve as a language to interact with computers as it is vague and ambiguous. Application of unrestricted natural language is infeasible and only subsets of natural language can be used. In the area of data bases, applications of natural language processing is given special emphasis due to its commercial value.

Generally speaking a system which supports a natural language interface to a data base consists of two modules.

a) The analyser of the statements and questions made by the user.

b) The generator of the statements and questions made by the system.

In this thesis we consider just the second module. Two kinds of practical text generation techniques are in general use today [29]. First the simplest and the most common way


semantic error messages or empty data base search
syntactic error messages
SUBSEQUEL processor
syntactically correct query
1
2
3
query
Data Base
Data Dictionary
User
response for update requests
structural information of the query
response in English
Lexicon
Algorithms
Information to direct the creation of the deep structure
Entity & Relationship Networks
Transformational component
Base component
surface structure sentence
deep structure sentence
6
5
4

1 - Syntactic analysis of the query.
2 - Semantic checks and processing of a correct query.
3 - Database results from the processing of the query.
  - Values of the database which exist in the predicates.
4 - Deep structure generation of the sentence by firing phrase structure rules.
  - Lexical insertion.
5 - Application of the cyclic and post-cyclic transformational rules.
6 - Insertion of the values in the response.

Fig. 1.1 Schematic representation of the generator

is to decide in advance what type of English sentence output is required and then store it as a text string. The computer simply displays the text that has been stored. For this type of text generation all questions and answers must be anticipated in advance. The other approach is to generate text by translating knowledge structures into English. In this case the quality of the generated text depends to a great degree on how the knowledge is structured. Our system is based on the second approach. Two types of problems must be considered in the generation of natural language response by the computer.

a) Determine the content and textual shape of what is to be said.

b) Transform the semantic structures representing what should be said into English.

In our system, algorithms which operate on semantic structures (entity and relationship networks) determine what is to be said. A transformational grammar is used to transform the semantic structures into English sentences.

This thesis is in the area of natural language generation. The research of this thesis involves the design of a system which generates responses in English to requests stated in a subset of the query language SEQUEL. The domain of discourse used as an example is a university environment. The various stages of the generation process are depicted in Fig. 1.1. The user presents a query written in a subset of the query language SEQUEL. The query is interpreted, and any syntactic or semantic error is reported back to the user. In case of an error the processing of the query is stopped, the user is expected to restate it correctly. A correct query is subsequently processed by using the RISS data base management system. RISS which is not shown in Fig. 1.1 accesses the data base and returns the results of the data base search. Structural information of the query is then passed to our algorithms. The algorithms, the networks and the data dictionary altogether guide the generation of the deep structure of the sentences to be generated. The deep structure of a sentence is generated in the base component by firing the phrase structure rules. At that stage the lexical insertion is also performed. Next the transformational component generates the surface structure of the sentence by applying the transformational rules. Finally certain values are inserted into the surface sentence and the response is given back to the user.

This thesis is presented in nine chapters. The first and the last chapters are the introduction and the conclusion respectively. The second chapter covers the subset of the SEQUEL query language which has been implemented as interface to RISS data base managment system. The third chapter studies the design of a database for a university environment based on the entity - relationship model. In chapter four the three basic grammars for natural languages, namely the transformational, augmented transition networks and case grammars, are presented. Generators which have been implemented by other researchers are also presented in this chapter. They are classified into systems which generate single sentences and multiple sentences (text). Chapter five describes the transformational grammar which is used in our system. The phrase structure rules, the transformational rules and structure building operations are presented. Chapter six studies the data dictionary and the lexicon which have been designed to support the generation process. Chapter seven deals with the semantic structures which are used in our system (entity networks and relationship networks). Joining of networks and generation of the deep structure of a sentence are also considered in this chapter. In chapter eight, eight algorithms are presented, one for each type of query. Several sample queries and the response sentences that would be generated by our algorithms are presented throughout this chapter.

II. A SEQUEL INTERFACE TO RISS DBMS

With the development of relational Data Base Management Systems (DBMS), several query languages emerged and SEQUEL is one of them. SEQUEL, also known as SQL (Structured Query Language) [2, 3, 10, 11, 17], is a nonprocedural relational query language which does not make use of quantifiers or other mathematical concepts. It is based on English keywords and operates on normalized relations. It provides the following facilities a) **Data manipulation** which permits insertion, deletion and update of individual or sets of tuples in a relational data base. b) **Data definition** which allows definition of relations and various views of relations. c) **Data** control which enables a user to authorize other users to access his data. SQL can be embedded in a high level host programming language such as PL/I.

2.1 A Subset of SEQUEL (SUBSQL)

I have implemented a subset of the Data Manipulation Language (DML) of SQL. I chose a subset of DML for the following two reasons a) A casual user is interested only for the retrieval of the information and only experienced users may want to update the DB or use other facilities. b) For the scope of this thesis responses dealing with retrieval of the information are appropriate. The responses on requests for the other kinds of facilities are stereotypical and they can be prestored and just printed out when each case occurs.

The following is the BNF syntax for the SQL subset which I have implemented. Expressions in square brackets "[", "]" are optional. Words with capital letters are terminals and must be inserted as they are. The slash "/" means "or".

```
Query         ::= Retrieve /
                  Insert /
                  Delete /
                  Update

Retrieve      ::= simpleretr /
                  nestedretr
Simpleretr    ::= SELECT  sel-list
                  FROM    from-list
                  [WHERE  where-list]
Nestedretr    ::= SELECT  attr-list
                  FROM    table-name
                  [WHERE  attr-list  IN  simpleretr
Sel-list      ::= [table-name.] attr-name /
                  sel-list , [table-name.] attr-name
From-list     ::= table-name /
                  fo+ke
Where-list    ::= [join-term AND] boolean
Join-term     ::= join-factor /
                  join-term  AND  join-factor
Join-factor   ::= table-name.attr-name = table-name.attr-name
Boolean       ::= boolean-term /
                  boolean  OR  boolean-term
Boolean-term  ::= predicate /
                  boolean-term  AND  predicate
Predicate     ::= [table-name.] attr-name comparison value
Attr-list     ::= attr-name /
                  attr-list , attr-name

Insert        ::= INSERT INTO table-name tuples-value
Tuples-value  ::= ( values ) /
                  SELECT  attr-list
```

```
                FROM    table-name
                WHERE   boolean
Values       ::= value /
                values , value

Delete       ::= DELETE table-name [WHERE boolean]

Update       ::= UPDATE  table-name
                SET     pred-list
                WHERE   boolean
Pred-list    ::= pred-value /
                pred-list  pred-val
Pred-val     ::= attr-name  =  value

Attr-name    ::= identifier
Table-name   ::= identifier
Value        ::= " string " /
                integer /
                real
Comparison   ::=  =  /  >  /  <
```

2.1.1 Sample Queries

I will give a typical example for every major type of queries which SUBSQL supports. This will illustrate in an informal way the range of acceptable queries. The queries are in the context of a sample schema of a university database (DB) given below:

DB Schema

STUDENTS(stud-id,stud-name[1],prog-name,status,phone,sex, citiznship)
PROFESSORS(prof-name, office-no, phone, sex, campus)
ADVISOR(prof-name, stud-id)

1. Double underlining denotes a secondary key.

Retrieval Examples

Simple Retrieval: Get all details of all the professors.

```
SELECT prof-name, office-no, phone, sex, campus
FROM   professors
```

There is no predicate in this type of query.

Qualified Retrieval: Get the names of the students and their program who have full-time status.

```
SELECT   stud-name, prog-name
FROM     students
WHERE    status = "F/T"
```

Retrieval from more than one relation: Get the name of professors who advise the student G.Bellos.

```
SELECT   prof-name
FROM     students, advisor
WHERE    students.stud-id = advisor.stud-id
AND      stud-name = "G.Bellos"
```

Retrieval using IN: Get the name and the status of students who are advised by professor J.Mylopoulos.

```
SELECT     stud-name, status
FROM       students
WHERE      stud-id  IN
        SELECT  stud-id
        FROM    advisor
        WHERE   prof-name  =  "J.Mylopoulos"
```

In this type of queries the operator IN is interpreted as follows. Let us assume that processing of the inner block results in the new relation RNEW(stud-id) which has the extension (33777, 77333). The condition

```
            :
WHERE    stud-id  IN
              SELECT  stud-id
                  :
```

evaluates to TRUE if at least one value of **stud-id** in STUDENTS is equal to at least one value of the RNEW relation's extension. This is equivalent to **equijoin** with double entries eliminated (**natural join**).

Update Examples

Single/Multiple tuple updates: Change the status to full time and program name to masters of the student G.Bellos.

```
UPDATE   students
SET      status = "F/T"
         prog-name = "M.Sc"
WHERE    stud-name = "G.Bellos"
```

Single tuple insertion: Add the professor E.Codd with office number 962, phone 77336, sex male and campus Sir George William into the relation PROFESSORS.

```
INSERT INTO professors
         ("E.Codd", 962, 77336, "M", "SGW")
```

Multiple tuple insertion: Enter into the relation TEMP the student names of the students who are in the Master program with full-time status.

```
INSERT    INTO    temp
          SELECT  stud-name
          FROM    students
          WHERE   prog-name = "M.Sc."
          AND     status = "F/T"
```

It is assumed that TEMP is a defined relation with **intention**

TEMP(stud-name).

Deletion of tuples: Delete the professor with office number 927 on Sir George William campus.

```
DELETE   professors
WHERE    office-no = 927
AND      campus = "SGW"
```

Deletion of a relation: Delete all the professors.

```
DELETE   professors
```

2.1.2 SQL and SUBSQL Differences

The subset of SQL that I have developed does not have all the features of the full SQL. My intention is not to develop the full SQL language but only that part which is needed in this research. Some retrieval queries can not be implemented because of restrictions imposed by RISS which are explained in section 2.3. The following types of retrieval queries are supported by SQL but not by SUBSQL: retrieval with ordering, (the user can specify an ordering to the resulting tuples); retrieval involving a join of a table with itself at the same nested level; nested level of the IN type queries being more than one; an attribute from a relation of the outer query appearing in the inner one; retrieval using ALL, NOT EXIST, UNION, etc. SQL provides a number of built-in functions to enchance its retrieval power, none of which have been implemented in SUBSQL. There are also the following syntactic differences in the two implementations. a) The nested queries are not enclosed in parentheses in SUBSQL. b) SQL does not eliminate duplicates from the result of a SELECT operation unless the user explicitly requests it via the keyword UNIQUE; SUBSQL always eliminates duplicates.

2.2 Implementation of SUBSQL

The implementation of SUBSQL has two distinct modules which perform the following tasks: a) Syntactic analysis of the query. b) Semantic checks, and the processing of the query. The second module uses the routines of the Application Level Interface and the Naive User Interface of RISS DBMS with some modifications [26, 32]. The **Application-User Interface** level of RISS consists of 24 primitive routines which can access and modify the DB. Any application program must access its relations through this interface. The **Naive-User Interface** is an application of the application user interface. It has 10 commands which allows a user to access and modify the DB. From these commands only the algebraic operators, SELECT, JOIN and PROJECT and the report facility REPORT are used in our system.

a) Syntactic analysis: This module has a set of routines for each category of query such as retrieval queries, insertion queries etc, to check its syntax. The correspondence of the category of the query and the syntactic routines is shown in Fig. 2.1.

Query Category	Main Syntactic Routine
Insert	Readinsert
Delete	Readdelete
Update	Readupdate
Select	Processblock

Fig. 2.1 Syntactic routines and the corresponding query category

PROCESSBLOCK calls a procedure for each clause of a retrieval type query. The procedures it calls are READSELECT, READFROM and INJOINSEL. They check the syntax of the corresponding clauses. When there is a syntactic error control returns to the command mode after it has printed an error message.

b) Semantic checks and processing of the query: When the query is syntactically correct it is passed to other routines for semantic analysis and further processing. Semantic analysis makes checks such as if a column has the required strategy. **Strategy** in RISS's terminology is one of the integer numbers 1, 3, 6, 9 assigned to each column of a relation. The numbers correspond to the following types of values: single character, integer, real and string of characters respectively. The number assigned to each column indicates the type of the value set for that column; if a column belongs to the relation it has been assigned; if the corresponding attributes of a join are compatible, etc. If there is not any semantic error the query is processed by the same routine. For each category of query there is a set of routines which performs these steps. The mapping of a query based on its category to the main routine which

Query Category	Main Semantic and Processing Routines
Insert	Insert <Tuplinsert, Relinsert>
Update	Updatetuples
Delete	Delete
Select	Readquery

Fig. 2.2 Correspondence of the query category with the semantic and processing routines.

performs these tasks is shown in Fig. 2.2. INSERT calls the procedures TUPLINSERT and RELINSERT to perform the semantic checks and processing of the two different types of insert queries, single tuple and multiple tuple queries respectivelly. The procedure RELINSERT calls the algebraic operators to select certain tuples based on the given predicates. The procedure READQUERY calls the procedures SELECT, PROJECT and JOIN which are the implementations of the algebraic operators. The algebraic operators are also called from any query which has the block structure SELECT - FROM - WHERE. The order in which the operators will be applied and which operators will be used depends on the form of the query. The block SELECT - FROM - WHERE in SQL represents the **mapping** operation [10, 17]. In algebraic terms it can be considered either as a SELECT followed by a PROJECT if only one relation is involved, or as a JOIN followed by a SELECT followed by a PROJECT if two relations are in the FROM clause. It is illustrated in the following two queries.

Query 1: Find the names of the students who are full time in the doctoral program.

```
SELECT  stud-name
FROM    students
WHERE   prog-name = "Ph.D."
AND     status = "F/T"
```

The required algebraic operators in the order of application are:

a. Select
b. Project

Query 2: Find the phone number of the advisor of the student G.Bellos.

```
SELECT  phone
FROM    professors
WHERE   prof-name IN
        SELECT  prof-name
        FROM    advisor, students
        WHERE   students.stud-id = advisor.stud-id
        AND     students.stud-name = "G.Bellos"
```

The required algebraic operators in the order of application are:

```
a. Join    ]
b. Select  } Inner query
c. Project ]
d. Join    ]
e. Project ]  Outer query
```

Before the application of each algebraic operation certain semantic checks are made. The operator is applied only if there is no semantic error.

2.3 Restrictions Imposed by RISS on the Queries

RISS [26, 32] is a relational DBMS which consists of the Application User Interface and the Naive User Interface.

Three types of operations can be performed on relations using RISS's Naive User Interface: a) Operations which allow the definition and deletion of relations b) Operations which allow data in a relation to be inserted, deleted, inspected and updated such as rows or specific columns in a row. c) Operations which allow relational algebraic manipulations of relations. A number of constraints are imposed on the form of SUBSQL queries. Some exist because of the way the above operations have been implemented and some depend on the global design features of RISS.

A relation in RISS can have a maximum of 10 attributes. Relations with more than 10 attributes must be split into two relations. At any time a maximum of 3 relations can be resident in the memory. The JOIN routine in RISS allows only natural join, and because of that in a query which requires the join of two relations only the "=" relational operator is allowed to be used to indicate the type of join operation to be performed. Retrievals using the operators <=ANY, >=ANY and ¬=ANY are not allowed in SUBSQL because of the above restriction. The =ANY can be rewritten using IN. The predicates following WHERE may include the following comparison operators =, <, and >. The operators >=, <= and ¬= are not allowed. The number of predicate must be less than or equal to 5.

RISS accepts up to 10 characters for the name of a relation and up to 20 characters for the name of a column. The latter can be increased by changing the value of COLSIZE. The range of acceptable values are: from -125,023 to +125,023 for integer. Real numbers can have exponent between -62 and +62 and the representation of mantissa is accurate to 8 decimal places.

III. ENTITY RELATIONSHIP CONCEPTS

3.1 The Entity - Relationship Model

The application of the Entity-Relationship (E-R) model to database design was proposed in 1976 by P.Chen [13]. Since then E-R modeling has become very popular. It is used to model the real world at a very high level (conceptual level). This model incorporates some of the important semantic information about the real world. Many semantic data models have been developed based on the notions of Entity and Relationship such as Mclead's [31]. The E-R model derives its formal basis from set theory and relation theory. In database design applications, the E-R model is used as a tool for the intermediate stage in the design process. From the E-R diagram a DB designer can derive a DB schema for any of the three popular data models [12, 13].

In the entity relationship model a real world is represented by means of entities and relationships between such entities. Each entity is characterized by a set of attributes. Corresponding to each attribute there is a value set from which that attribute may take its values. A value set may be common to more than one attribute.

An **entity** in the E-R model is an object such as a specific student, course, professor etc. **Relationship** is an association among entities such as STUDENT-COURSE which is a relationship between the entity student and the entity course etc. Entities are grouped into **entity sets** such as all the specific students make the entity STUDENT. A **relationship set** is a relation among n entities each taken from the entity set. All the instantiations of the relationship STUDENT-COURSE make its relationship set. The function of an entity in a relationship is specified by its **role.** Mostly, the identifications of roles are same as the name of the entities and because of that they are not shown explicitly in the formalism. Another important concept in the E-R model is the attribute. An **attribute** is a function that maps an entity set or relationship set into a value set or a Cartesian product of value sets. Certain attributes which identify entities are used as **entity keys.** In a relational schema the entities and relationships are translated into **entity** relations and **relationship relations** respectively. In these relations the entities are identified by their keys. The key of a relationship is formed from the key of the constituent entities. On the other hand there are entities whose existance depends on other entities. For example a "section" depends on a "course". It can not stand alone in the world of university. This kind of entity is mapped into a relation called **weak entity relation.** Relationships which consist of weak entity relations are called **weak relationship relations,** the other type of entities and relationships are called **regular.** This distinction is useful to maintain data integrity.

Flavin M. [19] has augmented the E-R model to enhance its power in order to solve certain problems faced by system analysts in the use of E-R model. The augmented E-R model involves the new concepts of operations and regulations. An **operation** is an action that changes the state of the system being modeled. For example the signing of a contract between two companies can be represented by an operation. A **regulation** is a rule that governs the content, structure, integrity and operational activity of the model.

The E-R model has been chosen for this thesis for two reasons: a) By modeling real world problems using the E-R model I can get a DB design which is suitable for the selected DBMS, namely RISS. b) This model embodies semantic information (meaning of the data) that will be used in the sentence generation module of my system (refer to chapters VII and VIII).

3.2 Database Design for a University Environment

3.2.1 E-R Diagram

The E-R diagram in fig. 3.1 represents the analysis of information in a typical university. The diagram shows the semantics of the database such as:

a) The weak entity SECTIONS depends on the entity COURSES so it can not stand alone.

b) The roles of the entities e.g. students, professors, classes etc, in the relationships are stated by the name of the corresponding relations.

c) It distinguishes between 1:n and m:n mappings, there is no 1:1 mapping in this case. The relationship set DEPT-STUD is a 1-n mapping, it means that a student can be registered in only one department.

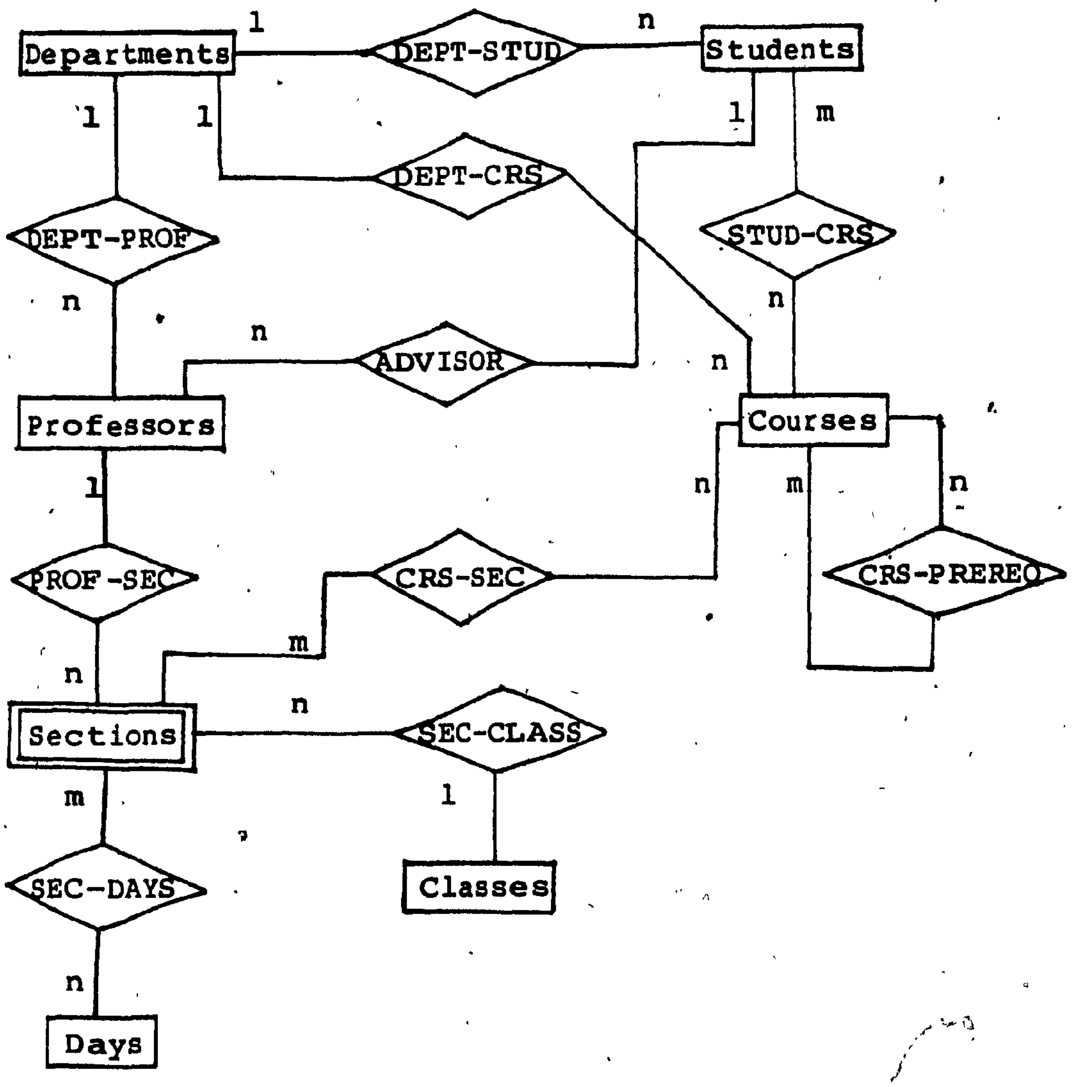


Fig. 3.1 The E-R diagram for analysis of the information of a typical university.

3.2.2 Value Sets Definitions

The next step in the design process of the DB is to define the value sets. A value set defines the type of an attribute and the maximum number of acceptable characters/digits for alphabetic/numeric types etc. This is indicated by the numbers in parentheses. The range of acceptable values or the values themselves, is shown in the third column of fig. 3.2. CHARACTER means alphabetic characters.

VALUE-SETS	REPRESENTATION	ALLOWABLE VALUES
Dept-Name	Character(15)	All
Course-ident	Integer(4)	All
Room-no	Integer(3)	0 - 999
Students	" (4)	5 - 1000
Name	Character(20)	All
Student-ident	Integer(5)	0 - 99999
Program-Name	Character(4)	B.Sc, M.Sc, Qual, Ph.D
Status	Character(3)	F/T, P/T
Phone	Integer	10000 - 99999
Sex	Character(1)	F, M
Citizenship	" (10)	All
Course-Title	" (30)	All
Credits	Real(2.2)	0.00 - 30.00
Capacity	Integer(2)	5 - 99
Grade	Character(7)	A, B, C, F, F/A Audit, In prog, Disc
Campus	Character(3)	SGW, LOY, VAN
Department-ident	" (4)	All
Day	" (3)	Mon, Tue, Wed, Thr, Fri
Time	Real(2.2)	8.00 - 23.00
Semester-Offered	Character(2)	F, W, FW
Section	Alphanumeric Character(2)	All
Stud-registered	Integer(2)	5 - 30

Fig. 3.2 Definition of Value Sets

3.2.3 Entity - Relationship Relations

The next step in the design of the DB is to define the attributes for each entity relation and relationship relation and to arrive at the value sets to which they are mapped. Finally the keys must be defined for each relation. The key in an entity relation stands for the entity, while the key attributes in a relationship relation stand for the constituent entities.

Regular Entity Relation **DEPARTMENTS**

Attribute/Value Set

Dept-id / Department-ident
Dept-name / Dept-name
F/T-stud / Students
P/T-stud / Students
Chairman / Name

Primary Key (PK)
Dept-id

Alternative Key(s) (AK)
Dept-name

Regular Entity Relation **STUDENTS**

Attribute/Value Set

Stud-id / Student-ident
Stud-name / Name
Prog-name / Program-name
Status / Status
Phone / Phone
Sex / Sex
Citiznship / Citizenship

Primary Key
Stud-id

Alternative key(s)
Stud-name

Regular Entity Relation **COURSES**

Attribute/Value Set

Course-no / Course-ident
Crs-title / Course-title
Credits / Credits
Sem-offerd / Semester-Offered

Primary Key
Course-no

Alternative Key(s)
Crs-title

Regular Entity Regular **CLASSES**

Attribute/Value Set	
Class-no	Room-no
Campus	Campus
Capacity	Capacity

Primary Key
(Class-no, Campus)

Regular Entity Relation **PROFESSORS**

Attribute/Value Set	
Prof-name	Name
Office-no	Room-no
Phone	Phone
Sex	Sex
Campus	Campus

Primary Key
Prof-name

Regular Relationship Relation **DEPT-STUD**

Attribute/Value Set	
Dept-id	Department-ident
Stud-id	Student-ident

Primary Key
(Dept-id, Stud-id)

Regular Relationship Relation **DEPT-CRS**

Attribute/Value Set	
Dept-id	Department-ident
Course-no	Course-ident

Primary Key
(Dept-id, Course-no)

Regular Relationship Relation **DEPT-PROF**
Attribute/Value Set
Dept-id / Department-ident
Prof-name / Name

Primary Key(s)
(Dept-id, Prof-name)

Regular Relationship Relation **ADVISORS**
Attribute/Value Set
Stud-id / Student-ident
Prof-name / Name

Primary Key
(Stud-id, Prof-name)

Weak Relationship Relation **SEC-CLASS**
Attribute/Value Set
Course-no / Course-ident
Section / Section
Class-no / Room-no
Campus / Campus

Existance of SECTIONS Depends on Existance of COURSES

Primary Key
(Course-no, Section, Class-no, Campus)

Regular Relationship Relation **CRS-PREREQ**
Attribute/Value Set
Course-no / Course-ident
Preregsite / Course-ident

Primary Key
(Course-no, Prereqsite)

Regular Relationship Relation **STUD-CRS**
Attribute/Value Set
Stud-id / Student-ident
Course-no / Course-ident
Grade / Grade
Instructor / Name

Primary Key
(Stud-id, Course-no)

Weak Entity Relation **SECTIONS**
Attribute/Value Set
Section / Section

Primary Key
Section
Course's Primary Key Through **CRS-SEC**

Weak Relationship Relation **CRS-SEC**
Attribute/Value Set
Course-no / Course-ident
Section / Section
Stud-Reg / Stud-registered

Existance of **SECTIONS** Depends on
Existance of **COURSES**

Primary Key
(Course-no, Section)

Weak Relationship Relation **PROF-SEC**
Attribute/Value Set
Instructor / Name
Course-no / Course-ident
Section / Section

Existance of **SECTIONS** Depends on
Existance of **COURSES**

Primary Key
(Instructor, Course-no, Section)

Regular Entity Relation **DAYS**
Attribute/Value Set
Day / Day
Class-hrs / Time

Primary key
Day

Weak Relationship Relation SEC-DAYS
Attribute/Value Set
Course-no / Course-ident
Section / Section
Day / Day
Time / Time

Existance of **SECTIONS** Depends on
Existance of **COURSES**

Primary Key
(Course-no, Section, Day)

3.2.4 Derivation of Relational Model from the E-R Model

The final step in the design of a DB is the mapping of the E-R relations into relations in the relational model. Both models use the concept of the relation (table form). **Value sets** in E-R model correspond to **domains** in the relational model; **Attributes** in the E-R model convey semantic meaning while **attributes** in the relational model are used to distinguish domains with the same name in the same relation. Chen [13] shows that entity and relationship relations of the E-R model are similar to 3rd Normal Form relations but with clearer semantics. From the above discussion we see that the translation of the entity relations and relationship relations from the E-R model into relations in the relational model is a 1-1 mapping. Fig. 3.3 gives a description of the above schema ignoring details such as value sets, weak entities and relationships.

This will help the reader to have an overall view and a global understanding of the schema. Attributes underlined with double lines indicate secondary keys.

DEPARTMENTS (Dept-id,Dept-name,F/T-stud,P/T-stud,Chairman)

STUDENTS (Stud-id,Stud-name,Prog-name,Status,Phone,Sex,
Citiznship)

COURSES (Course-no,Crs-title,Credits,Sem-offerd)

CLASSES (Class-no,Campus,Capacity)

PROFESSORS (Prof-name,Office-no,Phone,Sex,Campus)

DEPT-STUD (Dept-id,Stud-id)

DEPT-CRS (Dept-id,Course-no)

DEPT-PROF (Dept-id,Prof-name)

ADVISOR (Prof-name,Stud-id)

STUD-CRS (Stud-id,Course-no,Grade,Instructor)

CRS-PREREQ (Course-no,Prereqsite)

CRS-SEC (Course-no,Section,Stud-reg)

PROF-SEC (Instructor,Course-no,Section)

SEC-CLASS (Course-no,Section,Class-no,Campus)

SEC-DAYS (Course-no,Section,Day,Time)

SECTIONS (Section)

DAYS (Day,Class-hrs)

Fig. 3.3 A Compact Form of the DB Schema

IV. SENTENCE AND TEXT GENERATION

4.1 Introduction

The purpose of this chapter is to introduce the grammars for natural languages. It also gives examples of the research done by other researchers in the area of language generation. At the begining of the chapter the three basic models of grammars for natural languages are illustrated. We consider the classification due to Winograd [46]. There are many other grammars not covered here; some of them are modifications of the three basic ones discussed here and some have been developed independently from other models.

Text generation is complementary to text analysis. It is considered easier to generate text than to analyse. Most systems incorporate both input sentence / text analysis and response sentence / text generation. All question answering systems involve both modules [8, 21, 23, 24, 27, 33, 34, 38, 40, 43, 45]. Much of the effort on these systems is concentrated in the analysis module. The development of analysis module makes the design of the generation module easier because the latter will use the same semantic structures and almost the same grammar with a few changes. For instance, in the case of an ATN grammar the tests and actions must be different for analysis and generation. Of late, researchers have started to consider text generation independently. The development of a text generation system requires the same effort as that for analysis. The same problems as for text analysis arise if we are interested in the generation of good quality text that looks natural and has anaphoric references and such.

The early systems for text generation did not use any knowledge representation structure. They generate sentences randomly. The generated sentences are syntacticaly correct but most of the time they are meaningless. These systems do not have much research interest. They have been developed just to test the descriptive adequacy of the test grammar. A system of this type is due to Friedman [21, 22]. Its design goals are to test the effectiveness of transformational grammar. Later more sophisticated programs were developed generating sentences from semantic structures [7, 8, 30, 38, 41, 47]. These systems differ mainly on the form of the semantic structures they use for the representation of the knowledge, and in the types of the grammars they are based upon.

The design of a natural language generation system can be broken down into the following three steps.

a. Design the structures for the representation of the knowledge. This will be required to generate meaningful text. The knowledge which is represented by the semantic structure will be mapped to the generated text.

b. Design a grammar to generate syntactically correct sentences. Transformational and ATN grammar are two examples. The grammar will use the semantic structures. It will apply transformations on the deep structure of the response to generate surface level sentences.

c. Apply rules of text and dialogue to generate sentences that flow together making the text dialogue "natural".

4.2 Grammars for Natural Languages

The structure of a natural language can be adequately expressed by a grammmar which has the power of the Turing machine. Chomsky classified grammars based upon their power. His classification divided the grammars into the following 4 types [5].

TYPE	NAME	RESTRICTIONS ON RULES
0	Unrestricted	-None (It has Turing machine power).
1	Context Sensitive	-The right side must be at least as long as the left side.
2	Context free	-Left side must be a single nonterminal.
3	Regular	-Left side must be a single nonterminal and right side must be a single terminal or a single terminal followed by a single nonterminal.

Type 0 is the most powerful; as the type number increases the grammar becomes more restricted.

4.2.1 Transformational Grammar (TG)

Some sentences which have the same structure can give different meaning, and on the other hand some sentences with different structure can give the same meaning. For example the following two sentences have different structures but convey the same meaning.

The student G.Bellos is advised by Professor J.Mylopoulos.

Professor J.Mylopoulos advises the student G.Bellos.

Sentences are considered to have two levels of structure. **The deep structure** which conveys the meaning of the sentence and **the surface structure** which gives the form of the sentence [25].

Chomsky has proposed the transformational model as a model for the natural language [15]. It has two components: (a) The **base component** which is a context free grammar that generates deep structures, and (b) the **transformational component** which consists of a set of transformation rules that operate on phrase-structure trees sometimes called

P(hrase)-markers [46]. The transformation rules start with the deep structure produced by the base component and they result in the surface structure. A transformation rule consists of a **structural description (SD)** which is a pattern to be matched against the phrase-structure tree and a **structural change (SC)** which specifies the operations to be made to the tree if the SD of the rule applies. Some of the transformations are **optional** which means that if they are not applied (when their SD matches), the meaning of the sentence will not change and the final sentence will still be correct. If they are applied, only the form of the sentence will be changed. On the other hand, there are **obligatory** transformations which if they are not applied, when their SD matches, will result in incorrect sentences. The applicability of some transformation rules depends on some additional **conditions.** The transformations are applied only if the condition is true no matter whether the SD matches a P-marker or not. The transformational rule in Fig. 4.1 is an example of the features defined above. More details will be given on chapter V.

AG Agreement

```
          /  (PRE)   [(DET)][(aSg)] X]      PRES   Y    /
                                            PAST

SD:       1    2             3               4     5    6
SC:       1    2             3             4<aSg   5    6

CONDITION: 4<Ø
```

Fig. 4.1 A transformational rule

Many researchers have proposed extensions and refinements to the basic transformational grammar model. A feature which is worth mentioning is the **rule ordering** [1]. The particular ordering of transformational rules affects the final generated sentence. There are derivations in which rule A can be applied, followed by rule B but if rule B is applied before rule A, A is no longer applicable.

In the transformational grammar model, the tranformational rules have augmented the context free rules of the basic component making the grammar more powerful. Also, these rules add abilities to copy, delete and move fragments of the sentence. The interaction between its two components is illustrated in Fig. 4.2.

This model is oriented for generation of sentences rather than for their analysis. It can be used for analysis but the resulting algorithms have been shown to be too time consuming especially when there is a large number of sentences [48]. Algorithms to analyse sentences are based on applying the transformations in reverse [34]. This may

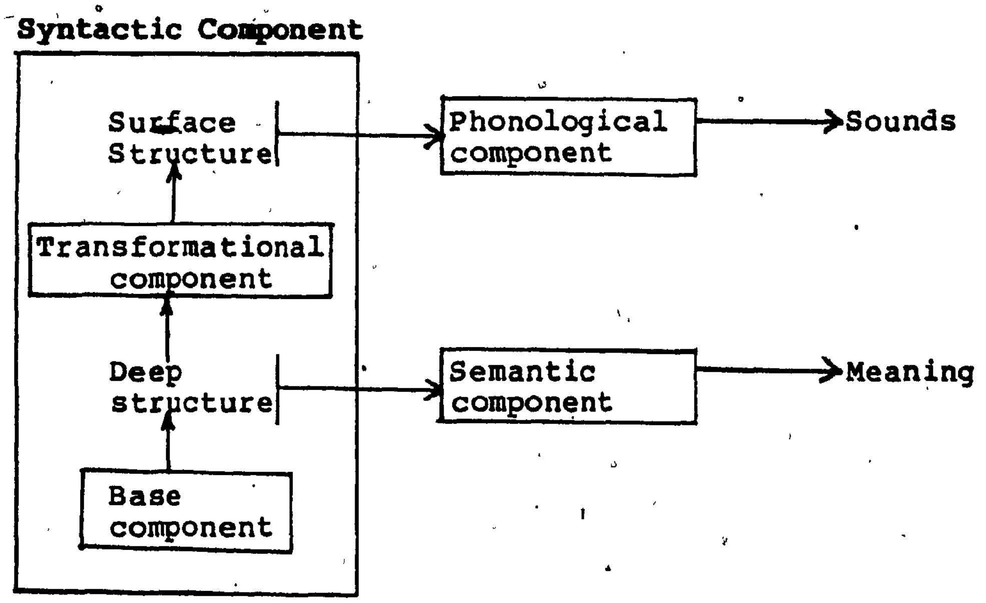


Fig. 4.2 The transformational model

some times result in incorrect deep structures. Another problem is the combinatorial explosion of the number of possible inverse transformation sequences that can be applied to a given surface structure tree; because most of the inverse transformational rules are optional. The tranformational grammar provides a syntactic description of a natural language which is far superior to that obtained by other grammars.

4.2.2 Augmented Transition Network Grammars (ATNG)

In the development of ATN grammars, researchers have progressed from the **Finite State Transition Graphs (FSTG)** to the **Recursive Transition Networks (RTN)** and finally to

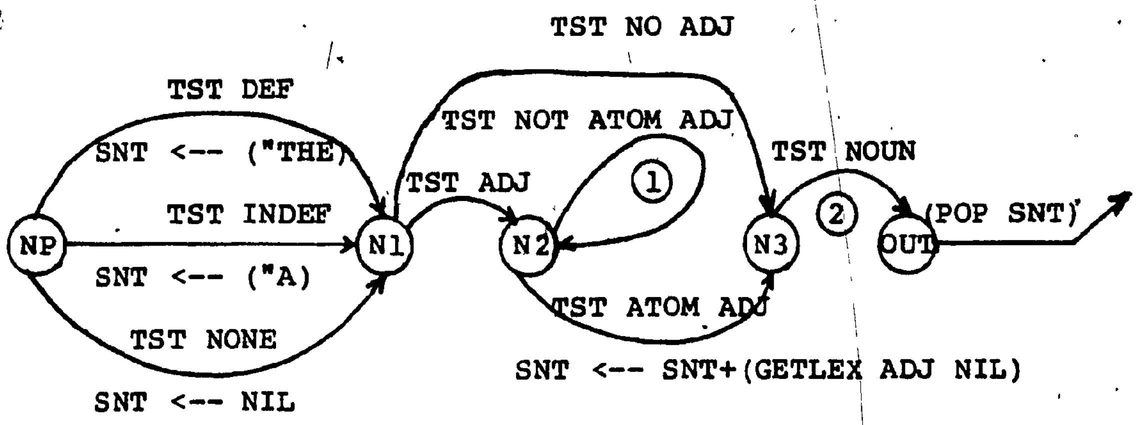
Augmented Transition Networks (ATN). A RTN is a FSTG except that the labels on the arcs may be a nonterminal or a terminal symbol, thus allowing recursion. The RTNs have the power of a context free grammar. To make them more powerful, facilities equivalent to the transformational component of TG have been augmented. The resulting grammar has a **test** associated with each **arc** of the transition network which must be satisfied in order to follow that arc. If the test is satisfied a set of **actions** are executed as the arc is traversed. The actions construct pieces of structure (tree structure, case structure etc) and keep them in certain registers. As the input is processed on the subsequent arcs the content of the registers may be changed, copied or deleted. This formalism is called the **ATN grammar** and it has been shown to have the power of a Turing machine [6, 48].

An ATN grammar has many advantages as a model for natural language and two of the most important advantages are the following:

a) It provides the power of the transformational grammars and the perspicuousness of the context free grammar model.
b) It is a model that can be used both for analysis and generation of sentences.

A simple ATN from Simmons's system [40] which generates

Noun Phrases (NPs) is shown in Fig. 4.3. Above each arc there is the required test, below the arc are the actions which must be carried out when the test succeds. The SNT, ADJ, NOUN, DEF are registers, ATOM, CAR and CDR are LISP functions. GETLEX is a function in that system which takes a structure name (e.g. TOK, ADJ etc) and a morphological attribute (e.g. NBM, TENCE etc) and returns the word form.



(1) SNT <-- SNT+(GETLEX (CAR ADJ) NIL)
ADJ <-- (CDR ADJ)

(2) SNT <-- SNT+(GETLEX TOK NBR)

Fig. 4.3 Noun phrase generation net

NIL is a LISP atom. TST and POP are arc names.

Systems which generate sentences based on ATN grammars include Simmons' [40, 41, 42], Wong's [47] and Shapiro's [38, 39]. These systems generate sentences from deep semantic structures using ATN grammars. The tests and the actions must be changed in an ATN grammar which is used for analysis of the sentences to be used as well as for the generation. Shapiro [39] has proposed changes to the basic model of the ATN grammar so that both the parser and the generator can use the same grammar.

4.2.3 Case Grammar

Case grammars provide a different approach to the problem of how syntactic and semantic interpretation of natural languages can be combined. Grammar rules are written to describe syntactic rather than semantic relationships. The structures produced by the rules (e.g. phrase-structure trees produced by context free rules) have also semantic relations than strictly syntactic ones. A case grammmar is a formalism which can represent semantic relations in these structures [36]. For example the syntactic structures of the next two sentences are almost identical.

1. Mother baked for three hours.
2. The pie baked for three hours.

In the sentences the relationship between **mother** and **baking** is different from that between the **pie** and **baking**. The following case grammar analysis of the above sentences distinguishes between the semantic roles of **mother** and **the pie.**

sentence no.	semantic structure
1.	(baked (Agent mother) (Time three_hours))
2.	(baked (Object the_pie) (Time three_hours))

The case grammar was proposed by Fillmore [18, 37] as a modification to the theory of Transformational Grammar. According to him, a sentence in its basic structure consists of a Proposition and a Modality.

Modality :	Includes modalities on the sentence such as negation, tense, mood and aspect.
Proposition :	Is a set of relationships involving verbs and nouns (and embedded sentences, if there are any). This constituent is expanded as a verb and one or more case categories. Each case occurs only once.

It can be expressed using context free rules as follows

S ---> M + P where S(entence), M(odality) and P(roposition)

P ---> V + C_1 + C_2 +...+C_n (At least one of the C_i i=1,..,n must be chosen)

C_i ---> K + NP   i = 1,..,n, K can be a preposition or empty (Ø).

The set of cases to be used is: A(gent), Counter_Agent (CA), (O)bject, R(esult), I(nstrument), G(oal) and E(xperiencer). This set is not unique and every application can have its own set of cases. In the following sentences **John** is Agent (A)

3. John opened the door.
   (case frame: [O(bject)+A(gent)])

4. The door was opened by John.

while the **key** in 5 and 6 sentences is Instrument (I).

5. The key opened the door.

6. John opened the door with the key.

7. The door opened.

The insertion of the verb depends on the **case frame** of the sentence. The case frames for the verb **open** in the above sentences are: [__O+A] in 3 and 4; [__O+I] in 5; [__O+I+A] in 6; and [__O] in 7 which contains only an O(bject). The case O(bject) is obligatory for the verb open. Sentences for open which do not have it are ungrammatical. The combined case frame for **open** in a more compact form is [__O(I)(A)] wherein parentheses indicate optional elements.

The choice of subjects for generation of a sentence follows the following rule

- **If** there is an A(gent), it becomes the subject; **otherwise** if there is an I(nstrument), it becomes the subject; **otherwise** the subject is the O(bject).

Let us take a simple example to see how a sentence is generated from its deep structure. Suppose the deep

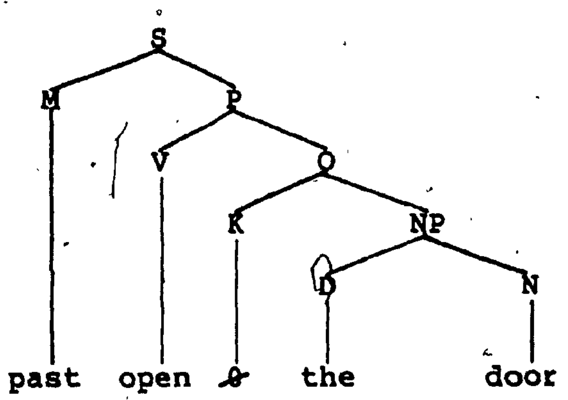


Fig. 4.4 Deep structure of a sentence

structure of a sentence is as in Fig. 4.4. Since the sentence contains only one case category namely O(bject), it is obligatorily moved to the front to become its subject.

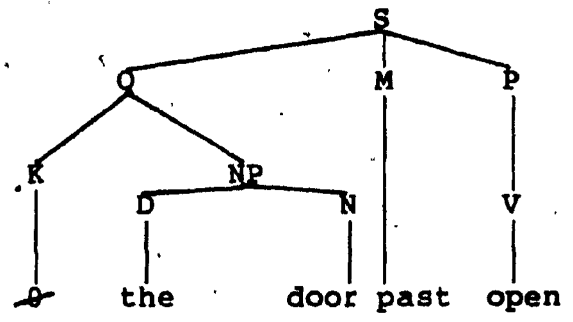


Fig. 4.5 The structure of the sentence after the first transformation

The result is shown in Fig. 4.5.

The final surface form after the incorporation of the tense into the verb is in Fig. 4.7.

After the application of
Subject-preposition deletion
transformation, we get

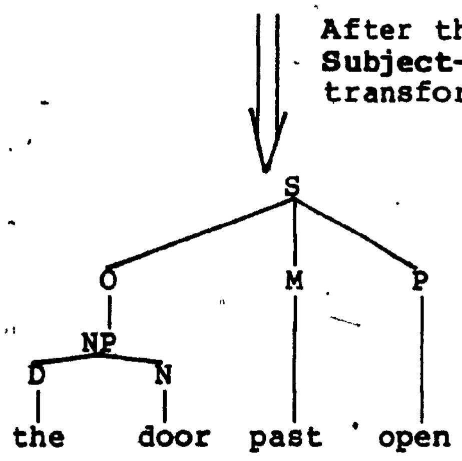


Fig. 4.6 The structure of the sentence after subject-preposition deletion transformation

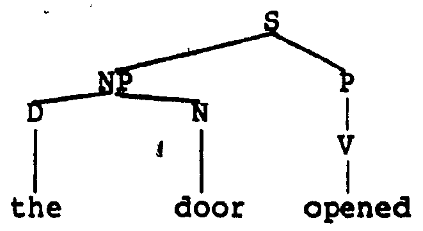


Fig. 4.7 Surface structure of the sentence

Case grammar deals with semantics of a language and in order to take care of the syntax it must be embedded within another formalism.

After Fillmore's proposal many systems have been designed based on case structures [40, 47]. Each one of them has its own peculiarities. The tendecy in these systems is to use case grammmar as a representation formalism for the semantic relatioships among objects.

4.3 Work Done by Other Researchers

4.3.1 Systems Focused Mainly on Generation of Single Sentences

The presentation of the systems in this section is focused on the generation of single sentences. Some of these systems like Simmons's [41] can generate multisentence text but that is not considered in detail. The examples in this section show specific applications of the three basic models of grammars, their advantages, disadvantages and their interactions.

4.3.1.1 Sentence Generators Based on TGs

Bate's & Ingria's Controlled Sentence Generator

Bate & Ingria [7] have developed a sentence generator to provide examples of particular constructions to some tutorial lessons. They will be useful to formulate exercises for people with language handicaps such as deafness. Its difference from other generators is that its focus is not on what to express but on how to express it in acceptable English. They have found TG as a good model for the generation of sentences that are syntactically related (e.g. the active and passive forms). By examining the derivation trees the system gets syntactic information about the generated sentence. This knowledge is given to the user as directions, hints to help him solving the exercise. Their generator consists of tree parts:

a) The **base component** which creates the deep structure tree in the X-bar framework using context free rules. Its input is a set of constraints that determine the type of the base structure which will be produced. The choice of words to be inserted in the base structure tree is controlled by a dictionary and a semantic network.

b) The **transformational component** which applies transformational rules to the trees to derive a surface tree. Its difference from the traditional transformational theory is that its transformational rules are not marked as optional or obligatory. In the standard theory if an obligatory rule does not apply, the result is ruled out as ungrammatical. Here **obligatory transformations** are applied when a structural description matches and, **optional transformations** are applied at random. In this system the surface structure form of a given deep structure is determined by external processes and not by the transformational rules. Because of this property the acceptance or rejection of the sentence does not depend on the transformational rules. The operations that are performed by the rules are the classic transformational operations such as substitution, adjuction etc. Operations which are usually performed by post-transformational processes in the base component such as morphing of inflected forms are performed in this component. Transformations are applied cyclically and within each cycle they are ordered. After all the cyclic syntactic transformations have been applied a set of post-cyclic transformations are applied whose domain of operation ranges over the entire tree. They supply the correct morphological forms of all lexical and grammatical items (e.g. plural form for nouns, the proper form for pronouns etc).

C) The third part is a mechanism to control the first two. A set of constraints directs the operation of the base component and indicates which constraints are to be applied.

According to the authors it is the most syntactically powerful generator, and it is fast, efficient and easy to modify and maintain.

Friedman's Random Sentence Generator

Friedman [21, 22] has developed a generator based on the transformational grammar model as presented by N.Chomsky in [15]. In this system the user has the facillity - to describe the syntax of the deep structure through what is called a basic skeleton. A basic skeleton is an incomplete phrase structure tree which will be filled out by the program. The skeleton has constraints which must be satisfied by the generated deep structure tree. The expansion of each node must satisfy the constraints imposed on it. These constraints protect the system from indefinite expansions for the case of recursive productions. There are three types of constraints, dominance (DOM), nondominance (NDOM) and equility (EQ). Details of this are found in [21] and [22]. If there are more than one valid productions which can be used for the expansion of a specific node of the skeleton then one is chosen randomly. The algorithm terminates when there is no more context free rule to be applied. The process is recycled if and only if there are unexpanded sentence symbols in the current string. The tree produced by the generation program forms an input to the lexical insertion program.

The **lexical component** has the prelexicon which has the definition of certain features such as category of the word and contextual features, and the lexical entries. The lexical insertion algorithm selects lexical items from the lexicon and inserts them into the tree. A word is inserted if the tree satisfies the appropriate contextual feature descriptions.

After the lexical insertion the deep structure tree is fed to the transformational component. The order by which the transformations will be applied is defined by the user with the help of a specially designed control language.

4.3.1.2 Generators Using Case Stractures and ATN Grammars

Most of the systems which make use of case analysis use it as a representation formalism, describing the relationships between a verb and the other objects such as nouns. A case structure can stand either for the answer to a question in a question-answering system or for the semantic representation of an input sentence or text. That semantic structure will be used by the generator to interpret it into a surface sentence. Many researchers have found ATN grammars a good model for the interpretation of case structures into surface sentences. Grammars in many generation systems and the ones which are described briefly in this section are designed on that concepts.

Simmons' Generator Based on Semantic Networks

Simmons [40, 41, 42] has developed a system that transforms English sentences into semantic structures using a grammar and a lexicon. A parser built from an ATN grammar creates semantic networks while it parses a sentence. Using a different ATN grammar than that of the sentence analyser, English sentences are generated from the semantic network.

Semantic Networks (SN): Semantic networks are a convenient formalism for representing such ideas as **deep structure, underlying semantic structure, etc.** They reflect the linguistic theory of deep case structures originated by Fillmore [18]. The semantic analysis of a sentence transforms it into an unambiguous representation. This representation consists of **concept nodes** interconnected by means of **semantic relations.**

Each sentence has an underlying structure of a verb, its modality (tence, aspect, mood etc) and its nominal arguments (any structure that functions as noun). The **deep case analysis** for the sentence, **John broke the window with a hammer** is in Fig. 4.8.

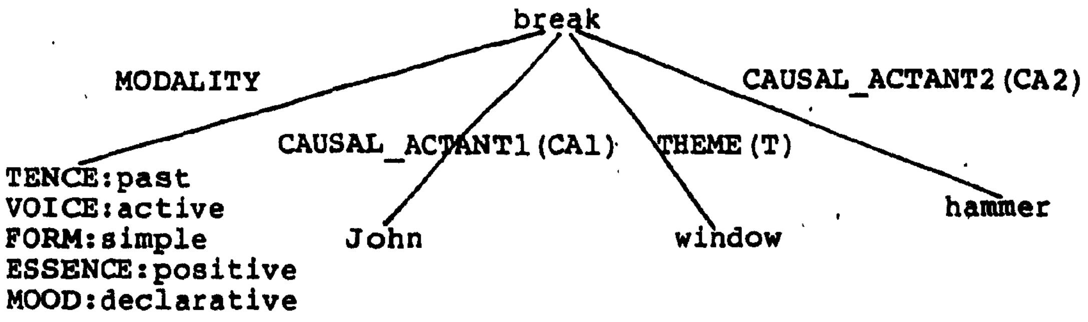


Fig. 4.8 Deep case analysis of the sentence "John broke the window with a hammer"

A concept node is a term such as C_i (contextual meaning of a word) or L_i (a meaning in the lexicon). **Semantic relations** are relations as the following

1. Connectives (OR, NOT, BUT etc)
2. Deep case relations (e.g. T(heme), C(ausal)-A(ctant),

G(oal) etc).

3. Modality relations (e.g. MOOD, ASPECT, VOICE etc).
4. Token substitution (TOK)en
5. Quantitative relations (e.g. NBM (number), COUNT, DET(erminer) etc)
6. Attributive relations (e.g. POSSESSIVE, SIZE, HASPART etc)
7. Set relations (e.g. SUP(erlative), PARTOF, EQ etc)

They can be viewed as a set of triples (A R B) where A and B are nodes and R a relation. The sentence **John broke the window with a hammer** has the semantic network structure in Fig. 4.9.

C1	TOK	break	C2	TOK	John	C3	TOK	window	C4	TOK	hammer
	CA1	C2		DET	Def		DET	Def		DET	Indef
	THEME	C3		NBR	S(ingular)		NBR	S		NBR	S
	CA2	C4								PREP	with

Fig. 4.9 Semantic network structure for "John broke the window with a hammer"

It can be represented in a graphical form as in Fig. 4.10

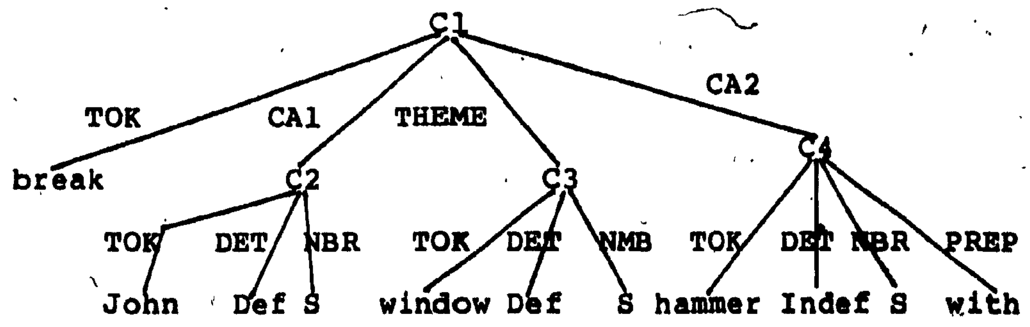


Fig. 4.10 A SN in graphical form

Sentence Generation from Semantic Networks (SN): The generation algorithm is based on an ATN grammar and a SN. The SN drives the grammar to generate meaningful sentences. If we generate sentences using only an ATN grammar and a lexicon the sentences can be syntactically correct but semantically meaningless. The use of the SN in the generation process make them to be meaningful. The **conditions** on the arcs of the ATN are tests for syntactic or semantic agreement, or the presence or absence of lexical features characterizing the form. The **operations** are transformations. In a semantically controlled generator the **conditions** and **transformations** associated with an arc in the ATN are applied to semantic nodes of the SN, transforming and generating sentences from them.

Using SN, multiple sentences can be generated as well. The generator function takes a list of nodes and a grammar as its arguments, e.g. GEN((E1,E16),GR). In the generation process new nodes may be added in the list, and nodes are removed as their generation is completed. The discourse generator stands above the sentence generator. In order to select portions of the SN from which sentences are to be generated, to exert thematic control via choice of subjects and the ordering of sentences.

Wong's Sentence Generator in TORUS

Knowledge Representation using Case Grammar: Wong [47] uses case grammar as a knowledge representation formalism, to encode general knowledge for the domain of discourse. With each event (verb), he has associated one (or more) case frames. For each case he finds a concept (noun, adjective) that can fit the role of that case and connects the event and the concept with an edge labelled under the name of the case. In the sentence

8. John broke the window with a bat.

the event is break whose case frame is (A,O,I). This case

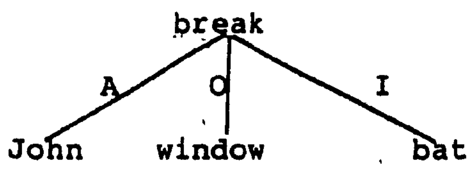


Fig. 4.11 Case frame representation for "John broke the window with a bat"

frame representation is shown in Fig. 4.11. Wong has extended Fillmore's set of cases in order to describe all the actions in his domain of discourse.

SELECTOR is an important routine in Wong's sentence generator which accepts a deep case representation of an answer to a user's question. The graph in Fig. 4.12 which is input to the SELECTOR represents the answer to the user's question

9. What is John Smith's address

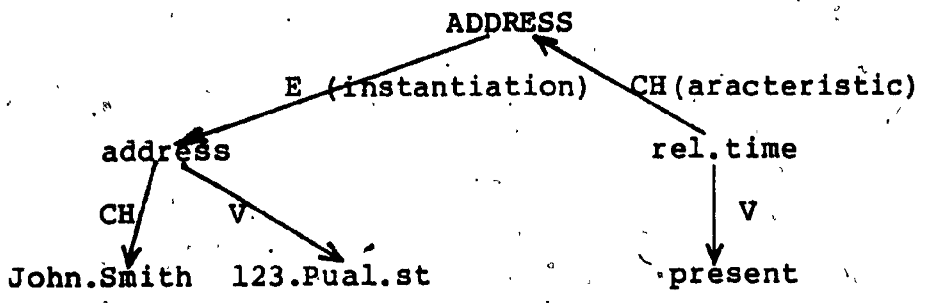
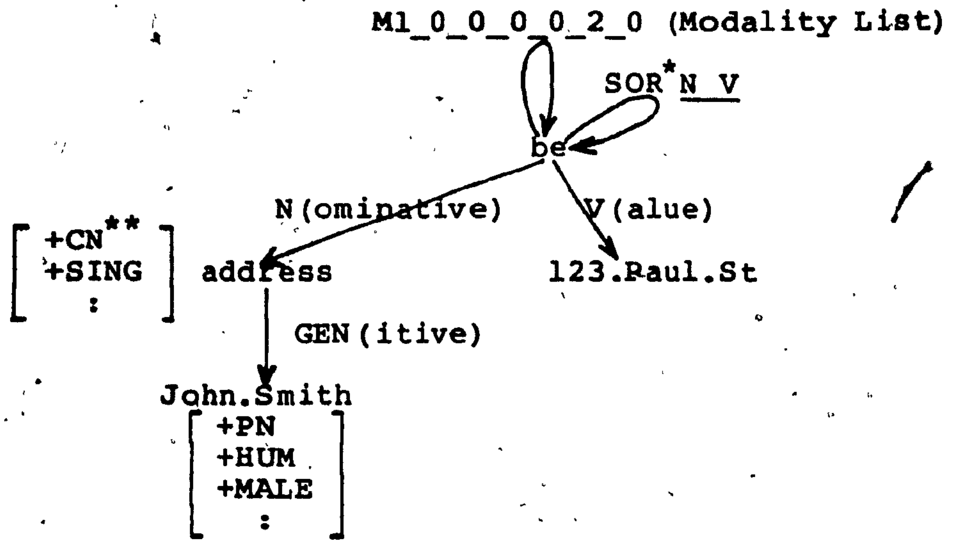


Fig. 4.12 Input graph to the selector

The SELECTOR builds a graph for the OUTPUT ROUTINE (another important routine in Wong's generator) to work with. The SELECTOR's output graph has a few additional cases than the input one. Its difference from the SELECTOR's input graph is that it is more "surface - structure" oriented. It contains syntactic structures, prepositions, ordinal, quantifiers, adjectives etc.

The SELECTOR's output graph for the above example (created from the graph of the Fig. 4.12) is shown in Fig. 4.13. Clearly Wong has used case grammar to represent the relationships among the objects in the domain of discourse namely the relationships between events (verbs) concepts (nouns, adjectives) and characteristics. His list of cases will change if the domain of discourse changes.

To arrive at the graph in Fig. 4.13 various rules and transformations have been used. Case placement is a set of



* ~~SOR:~~Surface Ordering Rule
** CN:Common Noun

Fig. 4.13 Output graph from selector

transformations which map a deep case structure onto a structure which is the same as the "deep structure" of the standard theory of transformational grammar. Basically, case placement transformations have to pick a case as the subject, one to be the object (if the verb is transitive) and the other cases become either complements or prepositional phrases. Two such transformations from Wong [47] are 9 and 10, given below:

9. If there is an Agent, it would become Subject

10. If the case Object were not the Subject, it would become the direct object.

Similar to case placement rules can be found in Fillmore [18]. Case placement transformations also find the appropriate prepositions to be associated with the cases and two of them are in Fig. 4.14. Their reason is: (a) The preposition for **agent** is **by.** (b) The preposition for the **instrument** in by if there is not agent, otherwise it is

	CASE	PREPOSITION
a.	Agent	by
b.	Instrument	with, by
	:	:

Fig. 4.14 Case - Prepositions Association

with. These are used by Wong mainly to construct the Surface Ordering Rule (SOR)-case frame, which plays an important role in the generation of a sentence. We assume that we have the event **break** with the following cases (A,O,I,T). According to case placement transformations, A(gent) is the Subject, O(bject) is the direct object, I(nstrument) and T(ime) are adverbials (prepositional phrases). Augmenting the SOR with the prepositions, **with** for I and at for T we get the final form for the SOR A_O_with, I_at, and T which corresponds to the sentence

11. John broke the window with a bat at noon

<u>ATN Sentence Generator for a Case Grammar System</u>: Wong [47] has implemented an ATN grammar which generates sentences from a deep case network. In ATN the transitions from state to state involve testing the deep structure and the registers being analyzed. If the tests were positive, some operations are performed on the deep structure and relevant information is stored in a set of registers.

Transformations are carried out from state to state. ATN is an efficient tool for realizing many of them. For embebbed structures such as complements and relative clauses, the grammar invokes other parts of the grammar to transform the embedded structures into surface sentences. The edges between two nodes are ordered 1,2,3,..etc. A test attached to each edge is evaluated based on that ordering starting from the first node and so on until a TRUE outcome is obtained to jump to the next state. There are three special action functions in the grammar namely PUSH, POP and JUMP and three preactions before every PUSH. An interpreter carries out operations of the grammar. It has three main functions, the MONITOR, STEP and ACTIVATE.

The MONITOR takes two arguments. One is a pointer to a state of the ATN, another to a node in the input graph (output graph of the SELECTOR). Initially the two arguments point to the starting state of the ATN and the event node of the input graph. The MONITOR then calls function STEP to make one transition in the network. When STEP returns, the MONITOR concatenates the string returned by STEP to the register SENTENCE. If the state returned by STEP were the final state of the grammar and if control were now in the highest level of computation, the MONITOR would return with the sentence in register SENTENCE. If either one of the above conditions is not satisfied, STEP is called with a new state of the grammar and a new pointer to the input graph.

ACTIVATE is called from STEP. It takes a label of a set of statements as argument and branches to it. After the statements are executed, it returns to the place where it is called. The following set of primitive functions (actions and forms) are used by ACTIVATE, (SETR, SENDR, GETF, SETF) which perform the same tasks as they have been defined by Wood in [48].

The grammar has three main components: Sentence, Verb String and Noun Phrase.

1. **Sentence network** is a top level network which controls the generation of complete sentences.
2. **Verb String Network** is a direct realization of Chomsky's treatment of auxiliaries. Basically the network generates the necessary verb string based upon a set of modality registers such as VOICE, ASPECT, MODAL etc.
3. **Noun Phrase Network** generates complements, conjoined noun phrases, complex noun phrases and personal pronouns.

Shapiro's Sentence Generator from Semantic Nets

Shapiro's [38] generator generates surface strings from semantic nodes of a semantic network. The generator accepts as input a node from the semantic network and optionally an ATN grammar. Calls to the generator with different nodes results in different sentences. For example the calls c_1

and c_2 to the same network with different nodes (M0026, M0023) result in the responses r_1 and r_2 respectively, Fig. 4.15.

c_1 (SNEG M0026)
r_1 (Charlie is believing that a dog kissed sweet young Lucy)

c_2 (SNEG M0023)
r_2 (A dog kissed sweet young Lucy)

Fig. 4.15 Calls to Shapiro's generator and the corresponding responses.

Actually the generator expresses the concept represented by the input node (e.g. M0023, M0026, etc) in a natural language surface string. SNEG is the top level generator function, M0026 and M0023 are nodes in the SN. The nodes in the network represent concepts which may be reasoned about. The edges represent semantic binary relations between nodes. A difference in Shapiro's SNs as compared to Simmons's is that Shapiro's SN representation does not include information considered to be features of the surface string such as tense and voice.

The generator uses an ATN grammar which differs from that of Wood [48] on the types of actions, arcs and forms. The form of the generated surface string (sentence, NP, etc) depends on the ATN node which optionally is passed as parameter to the generator. If the grammar node is not given, generation always begins from a specific node of the ATN. Each arc of the ATN nodes has functions which examines the SN connected with the given semantic node to 'decide' for the surface form of the sentence which will describe that node. Different ATN nodes may generate different surface forms for the description of the same node.

In a more recent paper [39] Shapiro lists the changes required in the ATN grammar as it has been proposed by Wood to be used both as a model for analysis and for generation of sentences. He proposes new types of arcs, actions and forms. The benefits from such changes is that parsers and generators can be written using the same grammar. A single interpreter can be written for both the parser and the generator. He also proposes a set of conditions under which the same lexicon can be used from the parser and the generator.

4.3.2 Systems Focused on Multiple Sentence Generation

Most research in the area of text generation has concentrated on sentence generation which is a very challenging research topic. There are cases such as in Computer - Assisted Instruction (CAI) where single sentences are not adequate. It is necessary to have multiple sentences or even multiple paragraphs to communicate complex information.

There are several systems which generate multisentence text. They can be divided into two groups according to the technique they apply to generate the text [29].

a. The simplest case is by having prestored information in frames. The material in the frames usually includes paragraphs of text to be presented. In the case of CAI they can have specific questions with their expected correct answers and some expected incorrect answers. This type of generation does not have reasoning. This type of systems are not interesting from research point of view.

b. Text can be generated from knowledge structures. This technique involves the translation of the knowledge into sentences. The quality of the text depends on how knowledge is structured. This kind of systems are very important. I will present two such systems: Mckeown's TEXT [30] and Carbonell's SCHOLAR [8].

<u>Carbonell's SCHOLAR</u>

SCHOLAR [8, 9] is a mixed-initiative instruction system which generates multiple sentences. Its database is about the geography of the South America. A segment from a sample session with a user is shown in Fig. 4.16.

```
         :                     :
Student  *Tell me something about Peru
Scholar  Peru is a country.
         It is in South America.
         The population is approximately 7000000
         people.The capital is Lima.

Student  *What is Cordoba ?*
Scholar  Cordoba is a city.
         It is located in Argetina.
         The population is 7000000 people.
         :                     :
```

Fig. 4.16 A fragment from a conversation of a student with SCHOLAR

In SCHOLAR the knowledge for the domain of discourse is represented in the form of semantic networks. Units are the basic components of the semantic network. Each unit is associated with a name such as Argentina, latitute, South America etc. It has semantic information about that object in the form of properties. Units form complex networks of facts, concepts and procedures. SCHOLAR uses its semantic network to generate the responses, by making calculations and inferences. The strategy adopted for the generation of sentences is that of using short sentences with no embedded clauses and a limited repertoire of verbs.

Mckeown's TEXT System

TEXT [30] is a system which generates text of paragraph lenght from a database (DB). It can anwser questions of the following three classes: a) questions about information available in the DB b) requests for definitions; and c)

questions about the differences between DB entities. The DB it uses is a portion of the Office of Naval Research (ONR) which contains information about vehicles and destructive devices. Questions must be phrased in a special notation. For example, the questions of the first class must have the form.

(definition <e>) where <e> represents an entity in the DB

TEXT's response to the question

What kind of data do you have ?

is

> All entity in the ONR database have DB attributes REMARKS. There are 2 types of entities in the ONR database: destructive devices and vehicles. The vehicles has DB attributes that provide information on SPEED-INDICES and TRAVEL-MEANS. The destructive device has DB attributes that provide information on LETHAL-INDICES.

To answer questions about the structure of the DB, the **knowledge base** is accessed. It has a high-level description of the classes of the objects in the DB. A subset of the knowledge base is selected which has information relevant to the question. It is called the **relevent knowledge pool. Schemas**, which encode aspects of discourse structure are used to guide the generation process, a schema is shown in Fig. 4.17.

```
attribute/identification (entity)
constituency (entity)
{attributive/identification
              (sub-class1, sub-class2,..)
     {evidence (sub-class1,sub-class2,..)}}+
{attributive/analogy (entity)}
```

Fig. 4.17 The **constituency** schema

There are four types of schemas each associated with different type of questions. For example the **constituency schema** described above is used in situations where an object or concept can be described in terms of its subparts or subclasses.

Once a question has been posed to TEXT, a schema is selected for the response structure based on the type of question. The mapping of schemas to question types is shown in Fig. 4.18.

Schema name	Type of Question
1. Identification	- requests for definitions
2. Attributive	- requests for available information
3. Constituency	- requests for definitions
	- requests for available information
4. Compare & contrast	- Requests about the differences between objects.

Fig. 4.18 Schema - question type mapping

For example in response to the question: **What is a projectile?**, the constituency schema is selected because more information is available about sub-classes of the guided projectile than the guided projectile itself.

<u>Schema</u> <u>Selected</u>: Constituency

<u>Predicates</u>

1. Identification
2. Constituency
3. Identification
4. Identification
5. Evidence
6. Evidence
7. Attributive

Fig. 4.19 Schema selected for the request, "What is a projectile?"

After a schema has been selected, it is filled by matching the predicates it contains against the relevant knowledge pool. The semantics of each predicate define the type of information it can match in the knowledge pool. The following response is generated from the schema shown in Fig. 4.19.

1) A guided projectile is a projectile that is self-propelled. 2) There are 2 types of guided projectiles in the ONR database: torpedoes and missiles. 3) The missile has a target location in the air or on the earth's surface. 4) The torpedo has an underwater target location. 5) The missile's target location is indicated by the DB attribute DESCRIPTION and the missile's flight capabilities are provided by the DB attribute ALTITUTE. 6) The torpedo's underwater capabilities are provided by the DB attributes under DEPTH (for example, MAXIMUM OPERATING DEPTH). 7) The guided projectile has DB attributes TIME_TO_TARGET__UNITS, HORZ_RANGE__UNITS and NAME.

The sentences are numbered to show that there is 1-1 mapping of the predicates with the generated sentences.

The schemas are implemented using a formalism similar to an ATN. A **focusing mechanism** is used to provide further constraints on the selection of information to be talked about next. The new sentence will be tied in with the previous discourse in an appropriate way. When the response has been constructed, the system passes the response representation to a component which uses a functional grammar to translate it into English.

V. A TRANSFORMATIONAL GRAMMAR FOR SENTENCE GENERATION

The requirements on the choice of a grammar are the following.

a) To be able to generate responses for each type of valid query expressed in SUBSQL.

b) The underlying grammar should express clearly the syntactic relationships of the constituents of the sentence such as deep structure subject, deep structure object etc. This requirement is needed because of the way I have designed the semantic structures and because of the way the generator creates the deep structure of the sentences.

c) The model of the grammar must be oriented and designed to be usedful for sentence generation as opposed analysis.

The transformational model of grammar satisfies the above requirements and our design of the generator is based on it. The first requirement will be satisfied depending on how big the grammar is. The other two are mainly inherent characteristics of the transformational model. Many researchers use the ATN model. Although an ATN grammar could be designed to generate sentences, it can not satisfy the second requirement.

5.1 Base Component

The base component also called phrase structure component generates the deep structure of the sentences and inserts the lexical entries into this structure. The lexical insertion will not be discussed here, it is done in the next chapter (VI).

5.1.1 Phrase Structure Rules (PS-Rules)

The base component contains a set of **Phrase Structure Rules** (or rewrite rules) as shown in Fig. 5.1. Braces indicate alternative expansions of the same symbol. For example T can be expanded either as T ---> PRES or as T ---> PAST. Symbols in parentheses are optional. How the derivation of the deep structure of a sentence proceeds is explained in chapter VII, here only the PS-rules are presented.

$$S \dashrightarrow (PRE)\ NP\ AUX\ VP$$

$$PRE \dashrightarrow (NEG)\ (Q)$$

$$AUX \dashrightarrow T\ (M)$$

$$T \longrightarrow \begin{Bmatrix} PRES \\ PAST \end{Bmatrix}$$

$$VP \dashrightarrow (Have\ en)(be\ ing)\begin{Bmatrix} V\ (NP) \begin{Bmatrix} (PP)^{*} \\ (S) \end{Bmatrix} (MAN) \\ be\ (ADJ) \end{Bmatrix}$$

$$PP \dashrightarrow \left(\begin{Bmatrix} EITHER \\ AND \\ OR \end{Bmatrix}\right) PREP\ NP$$

$$MAN \dashrightarrow PREP\ P$$

$$NP \dashrightarrow \left(\begin{Bmatrix} MORE\ THAN \\ LESS\ THAN \end{Bmatrix}\right) (DET)\ N\ (PP)^{*}\ (S)$$

$$DET \dashrightarrow ART\ (S)$$

$$ART \dashrightarrow (WH) \begin{Bmatrix} DEF \\ INDEF \end{Bmatrix}$$

Fig. 5.1 Phrase structure rules

The following abbreviations are used throughout this chapter and the thesis. S(entence), PRE(sentence), Noun Phrase (NP), AUX(iliary), Verb Phrase (VP), Prepositional Phrase (PP), T(ense), M(odal), NEG(ative), Q(uestion), V(erb), PRES(ent), PREP(osition), N(oun), ADJ(ective), DEF(inite), INDEF(inite), P(assive), MAN(ner) - agentive phrase, DET(erminer), ART(icle), $ (sentence boundary). The aspectuals (have en) and (be ing) are features of the VP in other TGs as in [1] they are treated as constituents of the AUX. The symbolism SYMB* means that the SYMB can be repeated one or more times.

5.2 Transformational Component

The transformational component converts the deep structure into a surface structure by applying successive transformational rules (T-rules). The transformational rules are applied on P-markers.

5.2.1 Structure-Building Operations

Each TG has some basic operations which are applied on the P-markers to transform them. The basic operations are shown in the SD of the T-rules. The operations required by this transformational component are the following:

a) Substitution: This operation substitutes one subtree by

SD:	A	B	C
	1	2	3
SC:	3	2	∅

another. The above T-rule when it is applied on the P-marker I gives the P-marker II, as shown in Fig. 5.2. It substitutes constituent A by constituent C.

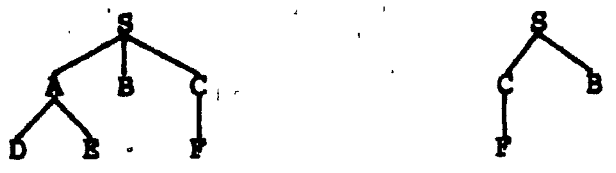


Fig. 5.2 Illustration of the substitution operation

b) __Deletion__: This operation deletes a subtree. For example

SD:	A	E	F
	1	2	3
SC:	~~0~~	2	3

the above T-rule when it is applied on P-marker I gives the P-marker II, as shown in Fig. 5.3. It deletes constituent A.

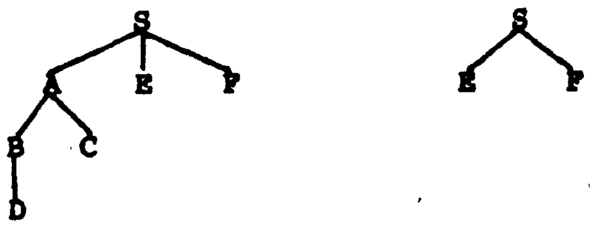


Fig. 5.3 Case I: Illustration of deletion operation

In case that some constituent X is the only daughter of some constituent Y and the deletion of X is defined, then Y is deleted as well. This is shown in Fig. 5.4 by applying again the previous T-rule. The T-rule requires the deletion of A but B is also deleted.

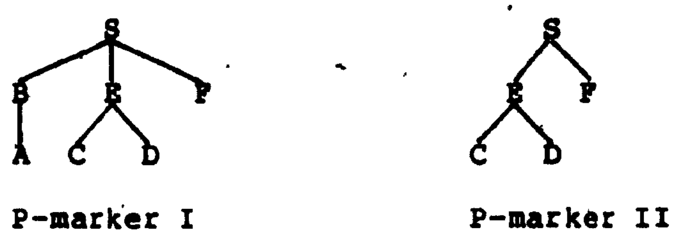


Fig. 5.4 Case II: Illustration of deletion operation

c) <u>Sister</u> <u>Adjunction</u>: This operation introduces a subtree under the immediate domination of some constituent which has

```
SD:   A     E    G
      1     2    3
SC:   1+3   2    ∅   "+" means sister adjunction
```

at least one daughter. The above T-rule when it is applied on the P-marker I in Fig. 5.5 gives the P-marker II. G is adjoined as sister to A.

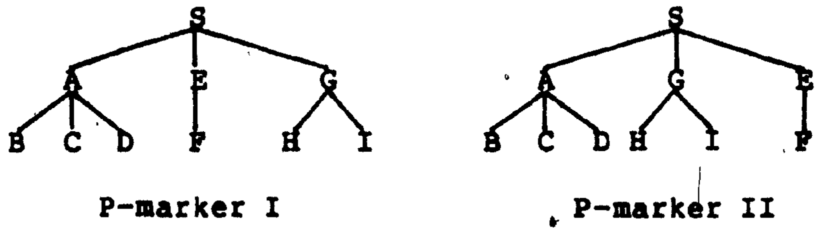


Fig. 5.5 Case I: Illustration of sister adjunction

If a constituent X is the only daughter of a constituent Y, and the sister adjunction of Z to X is defined then Z does not adjoin Y. The above rule when it is applied to P-marker I Fig. 5.6 gives the P-marker II. The rule requires G to be adjoined as sister to A, so it is adjoined as sister.

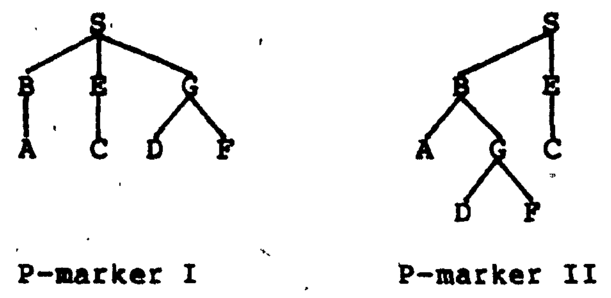


Fig. 5.6 Case II: Illustration of sister adjunction

If a constituent X is the only daughter of a constituent Z, X is to be adjoined as a sister to Y, then Z is adjoined to Y. The above rule when it is applied to the P-marker I in Fig. 5.7 gives the P-marker II. The rule requires G to be adjoined as sister to A, but F is adjoined as sister to it.

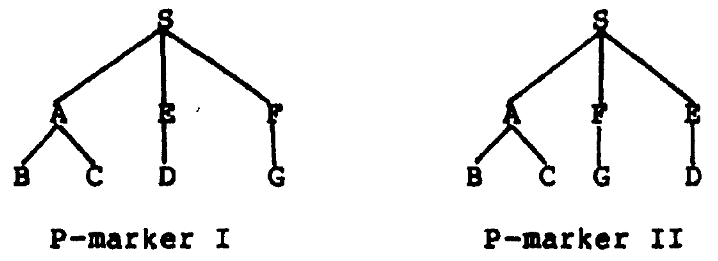
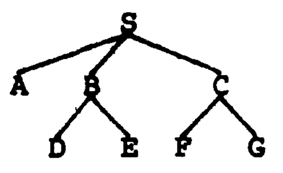


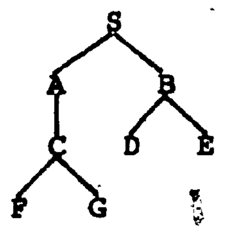
Fig. 5.7 Case III: Illustration of sister adjunction

d) <u>Daughter</u> <u>adjunction</u>: This operation introduces a subtree under the immediate domination of some constituent which does not dominate any other constituent. For example the T-rule,

```
SD:   A     B     C
      1     2     3
SC:   1<3   2     ~~0~~   "<" means immediately dominates
```

if it is applied on the P-marker I in Fig. 5.8 results in P-marker II. The T-rule requires C to be immediately dominated by A.

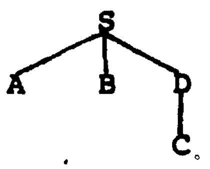


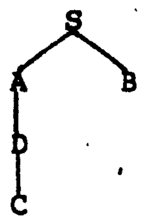


P-marker I P-marker II

Fig. 5.8 Case I: Illustration of daughter adjunction

If a constituent X is the only daughter of a constituent Z, X is to be adjoined as a daughter to some constituent Y which dominates no other constituent, then Z is adjoined as a daughter to Y. This is shown in Fig. 5.9 by applying the above rule. D is adjoined as daughter to A instead of C.





P-marker I P-marker II

Fig. 5.9 Case II: Illustration of daughter adjunction

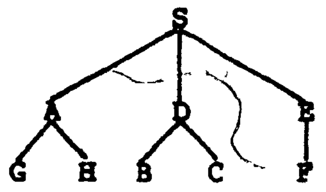
5.2.2 Structural Features of the Transformational Rules

Before presenting of the transformational rules some features of their structure are explained.

a) A label bracketing means that a subtree either must have a certain analysis or it must dominate some particular

SD:	A	[B	$C]_D$	E
	1	2	3	4
SC:	1	∅	3	4

subtree. For example the above T-rule matches in the next



tree. Its SD requires among other things that it must consist of B and C. The following T-rule is also applied to the above tree, its SD requires that the same subtree D must dominate B followed by C.

SD:	A	$[B\ C]_D$	E
	1	2	3
SC:	1	∅	3

b) A subtree must be identical or not identical to some other subtree, this is not expressed in the SD but in the condition part of the rule. For example the rule

SD:	A	B	C	A
	1	2	3	4
SC:	1	2	∅	4

Condition: 1 = 4

cannot be applied on the P-marker I but it can be on the P-marker II of the Fig. 5.10.

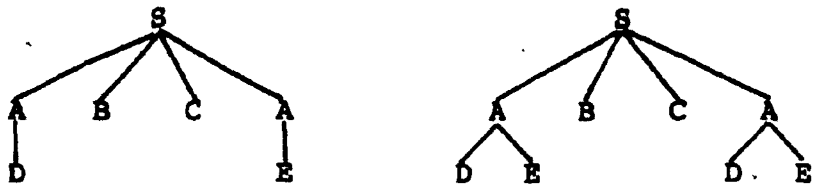
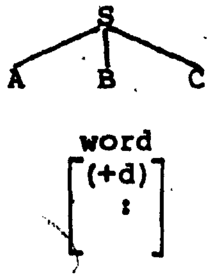


Fig. 5.10. Illustration of the use of the condition in a T-rule.

c) A trasformational rule may include syntactic features of a **complex symbol** (syntactic feature vector). The next T-rule in order to match a P-marker, among other things, it

```
SD:   A    [(+d)]B    C
      1       2       3
SC:   1       0       3
```

requires that the symbol B must dominate a complex symbol with the syntactic feature (+d). This T-rule matches the above P-marker.



d) Transformations of the following form match a terminal

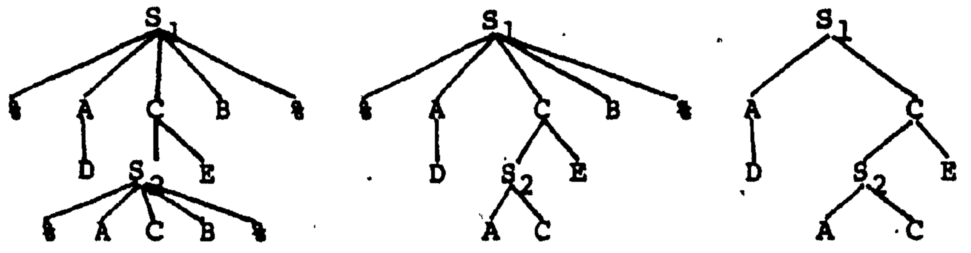
```
SD:   A    (+c)    B
      1     2      3
SC:   1     2      0
```

string, if it can be segmented into three substrings of which the first is an A, the second is a complex symbol with the feature (+c) and the third is a B.

There are two ordered sets of transformational rules, the cyclic and post-cyclic. The cyclic rules apply in sequence first to the lowest sentence. The sentences are seperated by the sentence boundary symbol (%). There is a T-rule which removes the boundary symbol. It is the finally applied T-rule. It makes the sentence at the next level lowest. For example if we apply the following two cyclic T-rules on the P-marker I of the Fig. 5.11, we get the P-marker II after their application in the innermost

```
T-rule 1:   SD:   %  A  X  B  %
                  1  2  3  4  5
            SC:   1  2  3  Ø  5

T-rule 2:   SD:   %  X  %
                  1  2  3
            SC:   Ø  2  Ø
```



P-marker I P-marker II P-marker III

Fig. 5.11 Illustration of the application of the cyclic T-rules.

sentence (S_2). Now sentence S_1 is the lowest sentence, the application of the T-rules on it results in the P-marker III.

After the application of the cyclic T-rules the resulting P-marker is the input to the post-cyclic T-rules. These rules are also ordered. For the example if we apply

```
SD:   A   C
      1   2
SC:   ∅   2
```

the above post-cyclic T-rule to the P-marker III of the

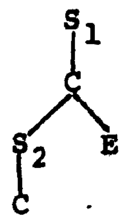


Fig. 5.12 Illustration of the application of the post-cyclic T-rules

Fig. 5.11 we get the P-marker in Fig. 5.12. The T-rules, both the cyclic and post-cyclic are always ordered. The symbols N1, N2 stand for specific nouns. Any other symbol which has no corresponding abbreviation as the ones given above, such as ADJ(ective), is assumed as variable. A variable e.g. X, X1, X2, X3, Z, etc can match anything. The symbol "a" is used to denote the alternative where both "+" and "-" are applicable, e.g. aHuman.

5.2.3 Transformational Rules

A subset of T-rules are presented in an order in this section. Readers interested for more T-rules they can refer [28].

Cyclic Rules

PASSIVE Passive OB(ligatory)

SD:	%	(PRE)	NP_1	AUX	V	(PREP)	NP_2	X	PREP	P	Y	%
	1	2	3	4	5	6	7	8	9	10	11	12
SC:	1	2	7	4	be+en+5	6	∅	8	9	3	11	12

Condition: $3 \neq 7$

RELPLACE Relative Placement OB

SD:	%	X	ART	S	N	Y	%
	1	2	3	4	5	6	7
SC:	1	2	3	∅	5+4	6	7

AUXFILL Auxiliary Filler OB

SD:	%	X	T	$\begin{Bmatrix}\text{be}\\ \text{have}\end{Bmatrix}$	Y	$\begin{Bmatrix}\text{V}\\ \text{ADJ}\end{Bmatrix}$	Z	%
	1	2	3	4	5	6	7	8
SC:	1	2	3+4	∅	5	6	7	8

AG Agreement OB

SD:	%	(PRE)	$[(\text{DET})[(\alpha\text{Sg})]_N\ X]_{NP}$	$\begin{Bmatrix}\text{PRES}\\ \text{PAST}\end{Bmatrix}$	Y	%
	1	2	3	4	5	6
SC:	1	2	3	4<aSg	5	6

Condition: 4<∅

WHAG WH-Agreement OB

SD:	%	X	WH	$\begin{Bmatrix}\text{INDEF}\\ \text{DEF}\end{Bmatrix}$	(ever)	$[(\alpha\text{Human})]_N$	Y	%
	1	2	3	4	5	6	7	8
SC:	1	2	3	4<aHuman	5	6	7	8

Condition: 4<∅

PROGDEL Progressive Deletion OB

SD:	#	X	N	(PREP)+WH+DEF	N	Y	#
	1	2	3	4	5	6	7
SC:	1	2	3	4	∅	6	7

Condition: 3=5

AF Affix OB

SD:	#	X	$\begin{Bmatrix} \text{T} \\ \text{-C} \\ \text{ing} \\ \text{en} \end{Bmatrix}$	$\begin{Bmatrix} \text{NP} \\ \text{(+V)} \\ \text{(+M)} \\ \text{have} \\ \text{be} \end{Bmatrix}$	Y	#
	1	2	3	4	5	6
SC:	1	2	∅	4+3	5	6

Condition: 5 = a terminal string s_1, s_2, ...s_n such that $s_1 \neq$ -C, ing, en, or T and 2 = a terminal string t_1, t_2, ..., t_n such that $t_n \neq$ (+V) or (+M).

AGDEL Agent Deletion OP(tional)

SD:	#	X	$[\text{PREP+INDEF+}[(\text{+PRO})]_N\,]_{MAN}$	Y	#
	1	2	3	4	5
SC:	1	2	∅	4	5

DETDEL Determiner Deletion OB

SD:	#	X	[DET	$[(\text{-DET})]_N$	$(\text{Y})]_{NP}$	(Z)	#
	1	2	3	4	5	6	7
SC:	1	2	∅	4	5	6	7

ISD Identical Subject Deletion OB

SD:	#	(DET)	$[N_1]_N$	(X)	(DET)	[(Y)	N_2	$(\text{Z})]_N$	(W)	#
	1	2	3	4	5	6	7	8	9	10
SC:	1	2	3	4	5	6	∅	8	9	10

Condition: 3 = 7

IND1 Identical Noun Deletion 1 OB

SD:	%	(Y)	$[X_1$	(DET)	$[X_2$	N_1	$(X3)]_N$	(X4)	(DET)	[X5	N_2	$(X6)]_N\,]_{NP}$	(W)	%
	1	2	3	4	5	6	7	8	9	10	11	12	13	14
SC:	1	2	3	4	5	∅	7	8	9	10	11	12	13	14

Condition: 6 = 11

IND2 Identical Noun Deletion 2 OB

SD:	%	X	[X1	(DET)	$[N_1]_N$	X2	(DET)	[(X3)	N_2	$(X4)]_N]_{NP}$	(X5)	%
	1	2	3	4	5	6	7	8	9	10	11	12
SC:	1	2	3	4	5	6	7	8	∅	10	11	12

Condition: 5 = 9

PREPM Preposition Marking OB

SD:	%	X	$\left\{\begin{array}{l}\text{[PREPDEL}_{PREP}\\ \text{[(EITHER)WITH}_{PREP}\end{array}\right.$	$\left.\begin{array}{l}[Y]_{NP}]_{PP}\\ [Y]_{NP}]_{PP}\end{array}\right\}$	[WITH_{PREP}	$[Z]_{NP}]_{PP}$	[WITH_{PREP}	$[W]_{PP}]$	U	%
	1	2	3	4	5	6	7	8	9	1
SC:	1	2	3	4	PREPDEL	6	7	8	9	1

ANDINS AND Insertion OB

SD:	%	X	$\left\{\begin{array}{l}\text{[PREPDEL}_{PREP}\\ \text{[(EITHER)WITH}_{PREP}\end{array}\right.$	$\left.\begin{array}{l}[Y]_{NP}]_{PP}\\ [Y]_{NP}]_{PP}\end{array}\right\}$	[WITH_{PREP}	$[W]_{NP}]_{PP}$	U	%
	1	2	3	4	5	6	7	8
SC:	1	2	3	4	AND	6	7	8

PRDEL Preposition Deletion OB

SD:	%	X	[PREPDEL_{PREP}	$[Y]_{NP}]_{PP}$	Y	%
	1	2	3	4	5	6
SC:	1	2	∅	4	5	6

ERASE Boundary Erasure OB

SD:	#	X	#
	1	2	3
SC:	∅	2	∅

Post-Cyclic Rules

NUM Number

SD:	X	$[(-Sg)]_N$	Y
	1	2	3
SC:	1	2+s	3

BE1 Be 1

SD:	X	be	$[+Sg]_{PRES}$	T
	1	2	3	4
SC:	1	∅	is	4

BE2 Be 2

SD:	X	be	$[-Sg]_{PRES}$	Y
	1	2	3	4
SC:	1	∅	are	4

BE3 Be 3

SD:	X	be	$[+Sg]_{PAST}$	Y
	1	2	3	4
SC:	1	∅	was	4

BE4 Be 4

SD:	X	be	$[-Sg]_{PAST}$	Y
	1	2	3	4
SC:	1	∅	were	4

HAVE1 Have 1

SD:	X	Have	$[+Sg]_{PRES}$	Y
	1	2	3	4
SC:	1	∅	has	4

HAVE2 Have 2

SD:	X	Have	$[-Sg]_{PRES}$	Y
	1	2	3	4
SC:	1	∅	Have	4

HAVE3 Have 3

SD:	X	Have	PAST	Y
	1	2	3	4
SC:	1	∅	had	4

WH1 WH 1

SD:	X	[WH	$[(+human)]_{\left\{ DEF \atop UNDEF \right\}}]_{NP}$	Y
	1	2	3	4
SC:	1	∅	who	4

WH2 WH 2

SD:	X	WH+DEF	Y
	1	2	3
SC:	1	which	3

WH3 WH 3

SD:	X	WH + INDEF	Y
	1	2	3
SC:	1	what	3

BY By

SD:	X	[PREP	NP]$_{MAN}$	Y
	1	2	3	4
SC:	1	by	3	4

INDEF Indefinite

SD:	X	INDEF	Y
	1	2	3
SC:	1	a	3

INDPLUR Indefinite Plural

SD:	X	INDEF	[(-Sg)	Y]$_N$	Z
	1	2	3	4	5
SC:	1	Some	3	4	5

DEF Definite

SD:	X	DEF	Y
	1	2	3
SC:	1	the	3

5.3 Range of the Generated Sentences

The responses which this system generates can be classified into the following four types.

Sentence Type	Example
1. Active	- The course Man-machine communication has the course number 772.
	- The professor with the office number 933 on Sir George William campus is

V.Alagar.

2. Passive - The student G.Bellos is registered in the bachelor program.

- The section with identification AA of the course with the number 643 is given by instructor R.Shinghal.

3. Relative Clauses - The courses which are offered in the Fall semester have the course number 772, 653, and 765.

4. Conjunction - The departments either with 50 full time and 110 part time students or with chairman professor C.Suen are Electrical Engineering and Computer Science.

- The Computer Science department has 50 full time students, 110 part time students and chairman professor C.Suen.

5.4 The Grammar

The grammar I have presented in this chapter is based on the IBM Core Grammar of English presented by D.Liberman in [28]. That grammar has been modified and extended to satisfy the needs of our system. The following changes have been made:

<u>PS-Rules</u>: The PS-rules have been changed to support:

a) Conjunctions such as EITHER-OR etc.

b) Prepositional Phrases in the deep subject NP.

c) The IBM CG suports two PP in the VP, whereas our system can handle more.

d) Two types of adverbs MORE-THAN and LESS-THAN are added.

Lexicon: For the lexicon's design see the next chapter (VI).

T-rules: Some T-rules have been modified and some new rules have been added.

a) The AUXFILL has been changed in order to be able to have the verb BE and HAVE as main verbs in a sentence.

b) The following cyclic T-rules added are: DETDEL, ISD, IND1, IND2, PREPM, ANDINS and PRDEL. They are used to improve the quality of generated sentences, either by removing redundant lexical items from the P-markers or by inserting new lexical items into the P-markers.

c) A new post-cyclic T-rule is added: INDPLUR. The new T-rules have been ordered with the others.

These new T-rules are applied in the sequence indicated in section 5.2.3.

VI. THE LEXICON AND THE DATA DICTIONARY

6.1 The Lexicon

The lexicon contains syntactic properties of the words used for sentence generation. The possession of a particular feature by a lexical item (word) is indicated by the symbol "+". For example, to indicate that the lexical item "department" is a noun we assign to it the feature "+N". Any feature which is not marked positively is assumed to be marked negative. It is implicit that the lexical item "department" has the features (-V), (-ADJ) etc. The lexicon is stored in the DB in a relational form, as a collection of tables. There are three types of syntactic features: 1) Syntactic categorization, 2) Inherent and 3) Strict subcategorization. The DB schema for the lexicon is shown

CATEGORYCL(<u>word</u>, category)

NOUNS(<u>noun</u>, npdeterm, ppdeterm, inherentf)

VERBS(<u>verb</u>, trn-intrn)

above.

For each word the relation CATEGORYCL stores its syntactic category such as (+N), (-ADJ) etc. There are words which can be classified into more than one category. For example the word "name" can be either adjective or noun or verb, so it must have the features (+ADJ), (+N) and (+V). In our system each word belongs to one and only one of the possibly many categories. This restriction does not seem to be a limiting factor of the sentence generator.

The relation NOUNS is a collection of words that could be used as nouns. The columns NPDETERM AND PPDETERM store features about the article of the corresponding lexical item. Certain nouns loose their article when they appear in PPs. The common, countable noun "department", for example, when is used as a constituent of a PP does not require a determinant. This fact is indicated in the attribute PPDETERM of the relation NOUNS by the value -DET. The assignment of features to a noun depends upon if the noun is used in a noun phrase or in a prepositional phrase. The two columns NPDETERM and PPDETERM in the relation NOUNS corresponds to these facts. The column INHERENTF in NOUNS indicates whether a noun is human or not and this is refered to as an inherent feature. The inherent feature is useful in handling certain cases such as anophoric references.

The last relation is VERBS. It stores information about verbs. The column VERB contains the root form of the verb and the column TRN-INTRN has indications of whether the verb is transitive or intransitive. This information is needed for the generation of passive sentences.

The columns NPDETERM, PPDETERM and TRN-INTRN may be marked in some cases to show that they have both features. For example the verb "teach" in the column TRN-INTRN has the feature +TR, it means that it can be in sentences with and without an object NP. In the terminology of [28, 1] these four columns have strict subcategorization features. They impose restrictions on the context of the corresponding noun or verb. As an example the restriction imposed on an intransitive verb is that the verb must not be followed by a NP. This type of features is also called contextual features.

The Fig. 6.1 shows sample extensions of the relations of the lexicon.

```
relation CATEGORYCL

(Department, +N)
(Credit, +N)
(On, +PREP)
(Offer, +V)
(Register, +V)

relation NOUNS

(Department, +DET, -DET, -HUM)
(Credit,+DET, -DET, -HUM)
(Computer Science, +DET, -DET, -HUM)
(Professor, +DET, +DET, +HUM)

relation VERBS

(Offer, +TR)
(Register, +TR)
(Belong_to, +TR)
(Give, ±TR)
```

Fig. 6.1 Sample entries from the lexicon

Consider the following problem, the verb "advice" requires both its subject and object to be human (to have the feature +HUM). Some systems as in [28] take care of this problem by having special features in the lexicon for each verb called selectional subcategorization features. In our system this problem is taken care of by the semantic structures in which the verb 'advice" is the label of an arc which associates nodes (attributes) whose corresponding nouns have the feature +HUM.

6.2 Data Dictionary (DD)

6.2.1 Data Dictionary in a Database Environment

A tool that enables one to control and manage the information about the data in ~~the~~ design, implementation, operation and expansion phases of a data base is called a **data dictionary**. The DD stores information about data such as its origin, description, relationship to other data, usage, format, and persons responsible for keeping the data up to date [17, 4, 44].

The term data dictionary, the way it is used in the data base environment stands for two things. a) The stored data discription information which is organized as a data base by itself. We call it **data dictionary data base** for clarification purposes. b) The software which creates and manages a data dictionary data base. The DD (software) may be integrated within a data base managment system or it may be a seperate package. Both approaches have advantages and disandvantages [4].

The data dictionary data base supports the conceptual, logical and internal models. For the conceptual model it has information about the data involved in the design process of this model such as entities, data elements representing the entities and relationships between the data.

elements. It also has appropriate labels, textual descriptions, synonyms, versions, memberships etc, which must have the entities, data element and the relationships. For the logical model it has information about the underlying data model, the relationships of the groupings based on the data model, the logical transactions, the programs, etc. For the internal model it has information about the physical store of the data such as lenght (character), mode (character string, floating point etc), precision for numeric elements, access control information etc.

6.2.2 A Data Dictionary for the Generator

The data dictionary in our system is a relational data base. It is used by the sentence generator. The generation process is based also on the information derived from the query and the E-R model. A user's query contains details such as attribute names, values and predicates. These details are considered to be at the "logical" level of a data base according to the three schema architecture of data bases [17]. Details about entities, relationships between entities are considered to be at the conceptual level. The DD contains information about these two levels. There is also linguistic information in the DD which does not belong to the above types. The relation PREP-ATTR associates to each attribute of the data base some prepositions. From the above discussions we see that the overall role of the DD, in our case, is somewhat extended.

The DD in our system is different in many points from a DD used in a DB environment. a) Our DD does not involves any special software. The DD is created using the naive-user interface level of RISS DBMS. b) It does not have any information about the internal model. c) Its use is different, it is not used in the development process of a DB. The fact that our dictionary is different from the standard DDs used in DB systems does not mean that it is based on different design concepts. This dictionary as any DD is data about data contained in the data base.

The schema of each relation of the DD is presented next with some sample extensions for each relation and a brief description about the content of that relation.

Schema: RELATIONS(<u>rel</u>-<u>name</u>, rel-type)

Extensions: (department, E-R)
            (advisor, R-R)

For each relation, the relation RELATIONS has its type, whether it is entity relation or relationship relation.

Schema: COLUMNS(rel-name,col-name)

Extensions: (advisor, stud-id)
(advisor, prof-name)

The correspondence of each relation to its attributes is stored in COLUMNS.

Schema: KEYS(identif, rel-name, key-size, key-id, col-name1, col-name2, col-name3, col-name4)

Extensions: (K1, department, 1, PR, dept-id, -, -, -)
(K2, department, 1, ALT, dept-name, -, -, -)

The primary and the secondary keys of each relation are in KEYS.

Schema: SYNONYMS(attribute,synon-attr, common-mng)

Extensions: (dept-id, dept-name, department)
(course-no, prereqsite, course)

SYNONYMS has synonym pair of attributes and their common meaning.

Schema: ABBR-ATTR(value-set, abbr-val, compl-val)

Extensions: (department-ident, COMP, computer science)
(program-name, B.SC, bachelor)

Certain values in the DB are abbreviated forms from words/expressions such as "master" is stored as "M.Sc". The complete meaning of the abbreviated values is stored in ABBR-ATTR.

Schema: ATTR-VSETS(attribute, value-set)

Extensions: (dept-id, department-ident)
(chairman, name)

ATTR-VSETS has the value set corresponding to each attribute.

Schema: ATTR-ENT(<u>attribute</u>, <u>entity</u>)

```
Extensions: (dept-id, department)
            (stud-id, student)
```

The relation ATTR-ENT is a collection of attributes each of which stands for a corresponding entity.

Schema: ENTITY-REL(<u>rel-name</u>,<u>entity-rep</u>)

```
Extensions: (departments, department)
            (students, student)
```

The entity corresponding to each entity relation is stored in ENTITY-REL.

Schema: RELOPERATR(<u>relop</u>, stands-for)

```
Extensions: (>, more than)
            (<, less than)
```

The meaning of each relational operator such as ">" is contained in the relation RELOPERATR.

The last two relations PREP-ATTR and ATTR-DOMNS have linguistic information. Some of our sentence generation algorithms require that certain attributes must build prepositional phrases.

Schema: PREP-ATTR(<u>attribute</u>, subjprep, objprep)

```
Extensions: (dept-id, with, with)
            (prog-name, in, in)
```

PREP-ATTR contains for each attribute the prepositions which can be used to build a PP based on its meaning.

ATTR-DOMNS has the meaning of each attribute and the form it takes when it will be used as constituent (e.g. subject, PP) in subject/object NPs. When the meaning is a composite

```
Schema: ATTR-DOMNS(attribute, full-name, used-mng, subjform,
                   subject, objform, d-mnouns, a-mnouns)

Extensions: (dept-id, department identification, department,
                        D_V, -, D_V, identification, -)
            (prof-name, name, professor, D_V,
                   A_V(-or, -and), A_V, -, -)
```

noun it also indicates the main noun. That information is used by the transformational component. The columns FULL-NAME and USED-MNG have semantic information, the meaning of each attribute. The difference of the two columns is that USED-MNG has shorter description of the meaning than FULL-NAME. That makes the sentences in certain cases to have more natural form. The values in the columns SUBJFORM, SUBJECT and OBJFORM are what I call **forms**. The symbols V, A and D are used to build the forms. V, A and D stand for V(alue), the meaning of the attribute taken from the column USED-MNG and the meaning of the attribute taken from the column FULL-NAME respectively.

VII. DESIGN OF THE GENERATOR

7.1 E-R Model and the English Language

The entity - relationship model is a tool used to model real word. It has been applied also to modelling sentences written in English. Basic constructs of English such as nouns, adjectives, etc, are mapped into objects of the E-R model such as attributes, relations etc. The inverse mapping is also possible. Refer to the relations ENTITY-REL and ATTR-ENT of the DD described in chapter VI. The ENTITY-REL maps an entity in the E-R model to a noun in English. Similarly the ATTR-ENT maps those attributes which stand for certain entities in the real world into nouns in English. The columns FULL-NAME and USED-MNG of the relation ATTR-DOMNS of the DD correspond to either a noun or a composite noun or a noun modified by an adjective. Consequently an **attribute** is mapped into a **noun** which may be modified by some other constituent of the English language.

The following three terms which are used very much in this and in the next chapter are explained here.

a) **Entity attribute** means an attribute which stands for an entity.

b) **Nonentity attribute** means an attribute which does not stand for an entity.

c) **Labeled arc** means an arc which has a **verb** as label. Some arcs have two verbs as label and either of them can be used.

The relationships between the entity attributes in the relationship relations (RR) are expressed by **verbs.** For example the relationship of the entity attributes STUD-ID and COURSE-NO in the relationship network STUD-CRS Fig. 7.2 is depicted by directed arcs and their labels correspond to verbs ("take"). The same is true for the entity and nonentity attributes of a relationship relation. The verb "get" on the arcs joining the entity attribute STUD-ID with the nonentity attribute GRADE represents the relationship between these two attributes. This is depicted in the relationship network of STUD-CRS in Fig. 7.2. In the case of a weak entity, its relationship with the entity on which it depends is expressed by a preposition. In entity relations (ER) there are similar associations between the entity and its attributes. For example, the verb "register" expresses the relationship between the entity "student" and the nonentity attribute PROG-NAME, Fig. 7.1. The entity attribute STUD-NAME is related to the entity "student" by means of the verb "be", Fig. 7.1. In conclusion we can say that objects or relationships among objects of the E-R model correspond to certain categories of words in English language such as nouns, verbs etc.

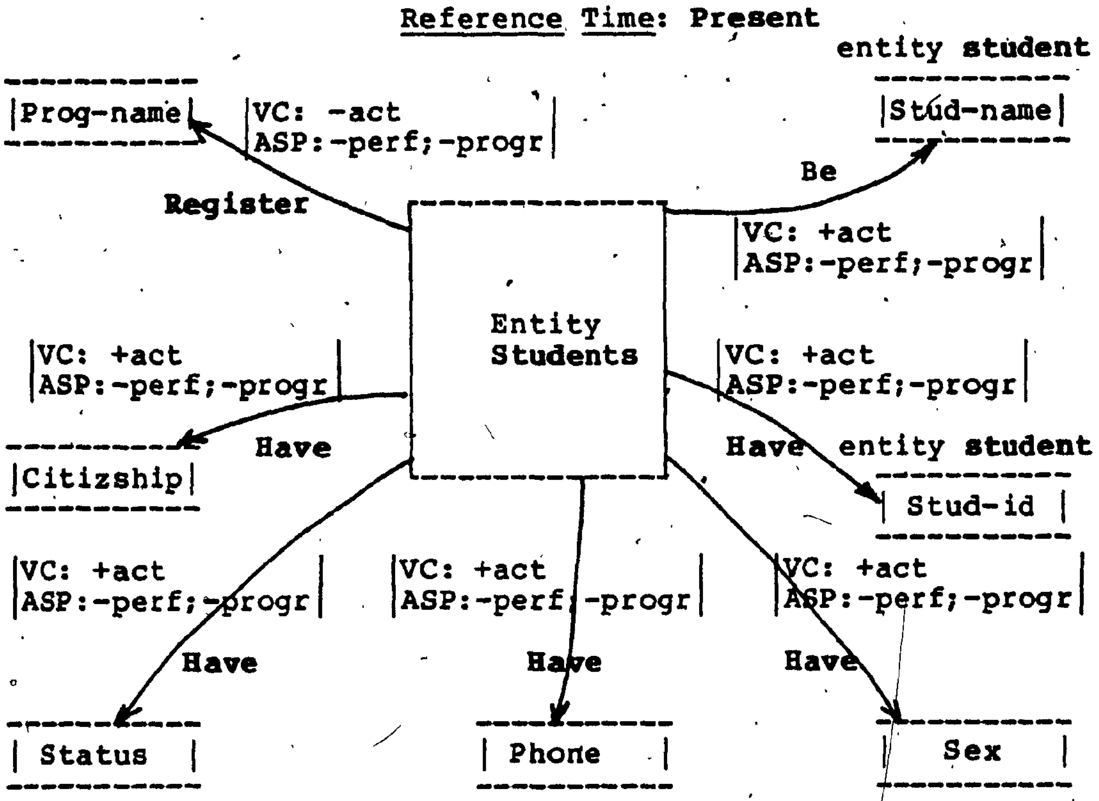
In [14] P.Chen has associated to each **entity** a **noun and** to each **relationship** a **transitive verb.** He makes a distinction between entity attributes and relationship attributes. He illustrates several examples where he shows that **entity attributes** correspond to **adjectives** in English and **relationship attributes** correspond to **adverbs** in English. In this thesis we handle entity and relationship attributes uniformly. The approach of P.Chen works perfectly in the domain he had chosen but does not work in other applications. For example, the relationship attribute GRADE in our domain does not correspond to an adverb in English. A similar objection can be raised for the attributes PHONE, SEX of the entity relation STUDENTS with regard to adjectives. P.Chen proposes some **rules** (guidelines) which can be used to translate English sentences into corresponding E-R diagrams. This thesis does the opposite task, it translates semantic structures into English sentences. The design of the semantic structures is based on the E-R model.

7.2 Semantic Structures

Two types of semantic structures are introduced in our system [35], the **Entity Network** and the **Relationship Network**. Their role in the sentence generation process is very important. The generated sentences are considered as translations of these network structures. Furthermore the join operation as defined below can be performed on these structures. This operation enhances the power of the generator resulting in the generation of multiple sentences or texts.

7.2.1 Entity Networks

For each entity in the E-R diagram an Entity Network is created which is represented by a big rectangle surrounded by its attributes which are represented by smaller rectangles. Labeled directed arcs join the entity with its attributes. The label of each arc is a verb accompanied by a **frame.** The frame has values for two features of the verb, the "voice" and the "aspect". The content of each relation either entity or relationship must satisfy some "time constraints". For example the entity relation STUDENTS can not have students who have already graduated. Because of this nature every relation is assigned a "reference time" which is the time interval over which the responses generated from that relation will be valid. Consequently, all the generated sentences are with regard to the reference time. Fig. 7.1 shows the Entity Network corresponding to the entity relation STUDENTS. Most of the arcs in the entity networks have the verb HAVE. It is used in the sence of **possesion,** that ~~is~~ the entity possesses such and such attribute. The association of the attribute PROG-NAME with the entity STUDENTS is expressed by the verb "register". The direction on the arcs has syntactic meaning; it means that the concept at the tail of the arc (entity) must appear in the surface subject - NP while the concept at the head of the arc (attribute) must appear in the surface object - NP with the label on the arc as the main verb. The verb can be



Reference Time: Present
entity student
Prog-name
VC: -act ASP:-perf;-progr
Register
Stud-name
Be
VC: +act ASP:-perf;-progr
Entity Students
VC: +act ASP:-perf;-progr
Have
Citizship
VC: +act ASP:-perf;-progr
Have
entity student
Stud-id
VC: +act ASP:-perf;-progr
Have
VC: +act ASP:-perf;-progr
Have
VC: +act ASP:-perf;-progr
Have
Status
Phone
Sex

Fig. 7.1 The entity network of the entity STUDENTS

used only in the indicated direction.

7.2.2 Relationship Networks

The Relationship Network shows the relationships which exist between two entity attributes or between an entity attribute and a nonentity attribute. There are no relationships between two nonentity attributes. The nonentity attributes have meaning only with respect to the relationships of the entities as it is stated by the relationship network. For example the nonentity attribute GRADE has meaning of existance in a relationship of a student with a course. The relationship network STUD-CRS on which this attribute appears expresses that relationship. In these networks most of the attributes are connected with a pair of arcs pointing in the opposite directions. That means that a noun which stands for an attribute can be either in the surface subject - NP of a sentence generated by using the outgoing arc or in the surface object - NP of a sentence generated by using the ingoing arc. The labels on the arcs may or may not be different. The Fig. 7.2 is the relationship network of the relation STUD-CRS (student-course). Inside the big rectangle are the entity attributes. It is also shown that an attribute which does not stand for an entity is related only to an entity attribute. For example the attributes GRADE and INSTRUCTOR are related to the entity attributes STUD-ID and COURSE-NO respectively. There are two more types of arcs:

a) A broken arc denotes the relationship of a weak entity and the one on which it depends. That dependence is also mapped in the generated sentences. The label of that arc is a preposition which is used to express the dependency in the generated response. The head of the arc points to the independent entity.

b) Unlabel directed arcs. They indicate that a NP/PP built

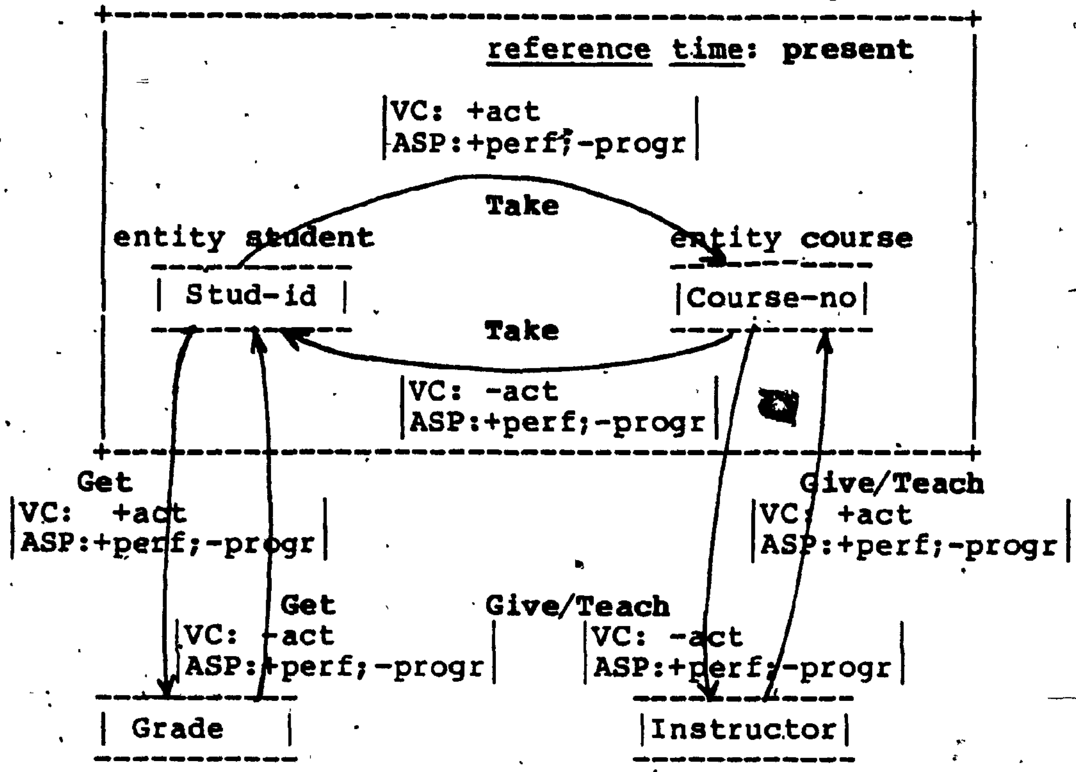


Fig. 7.2 The relationship network of the RR STUD-CRS

from the attribute at the tail of the arc must be followed by a NP/PP built from the attribute at the Head of the arc.

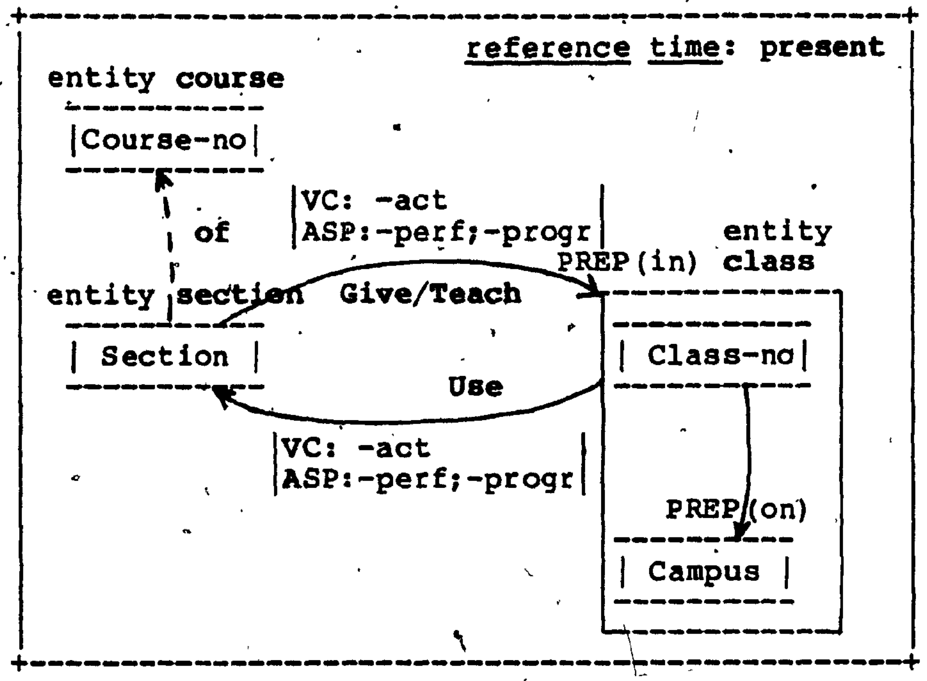


Fig. 7.3 The relationship network of the RR SEC-CLASS

Another symbolism which we introduce is the "PREP(preposition)". It can be either at the tail or at the head of an unbroken arc. It indicates that the attribute in that part of the arc must build a PP with the preposition inside the parentheses of PREP. The above symbolisms are shown in Fig. 7.3. We can see that "section" is a dependent entity on the entity "course". The label on the broken arc is OF. The entity CLASS must be preceded by a preposition when it is in the object NP. A NP built from the attribute of the node CAMPUS must always be preceded by preposition "on". That PP must follow the PP/NP built from the attribute of the node CLASS-NO.

The most important differences between the entity and relationship networks are the following:

1) There is no relationships among the attributes in an entity network.
2) Most of the attributes in a relationship network are connected with a pair of arcs pointing in the opposite directions. It means that the entity and nonentity attributes are handled in the same way.
3) Different types of arcs have been defined in the relationship networks to represent different kinds of relationships which occur among attributes.
4) The entity networks are built around entities of a E-R diagram.

7.2.3 The Join Operation on the Networks

To answer certain queries one needs to access more than one relation. In that case, to generate a response more than one network must be traversed. To make a logical transition from one network to another these multinetworks must be connected in some way. The definition of the join operation allows a logical transition from one network to another.

Let us consider the nodes which stand for entities as **primary nodes**. The ones which do not represent entities are **secondary nodes**. Then the join operation on the networks is defined as follows. Two networks can be joined on a node that satisfy the following requirements.

a) If the node stands for an entity, the node in both relations must represent the same entity.

b) If the node does not stand for an entity both nodes must have the same name.

c) The join can be performed on more than one common nodes. The join operation merges the common nodes leaving the other part of the network as it is.

d) Entity networks are not allowed to be joined with each other.

e) The networks to be joined must have the same reference time.

Reference time: present

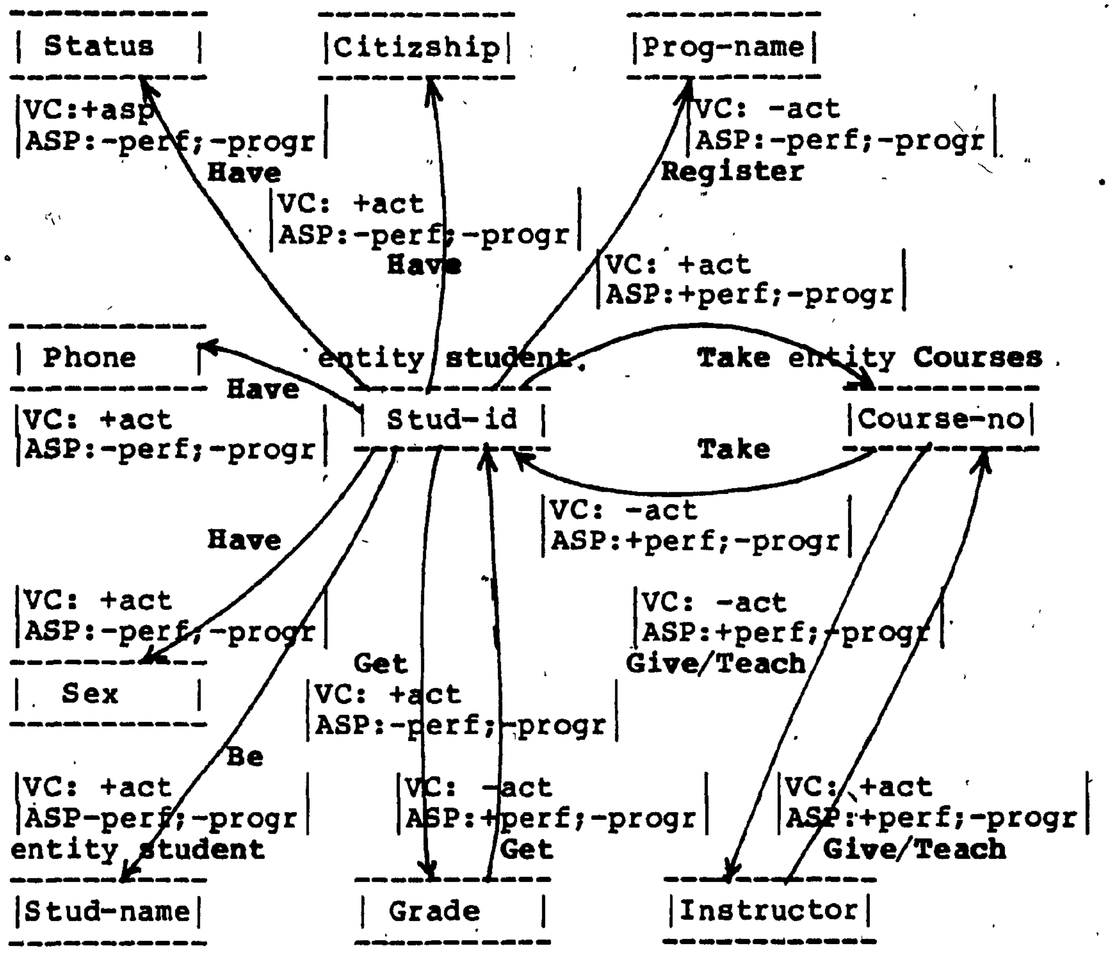


Fig. 7.4 Illustration of the join operation

Fig. 7.4 shows the join of the entity network STUDENTS with the relationship network STUD-CRS. They have been joined on the common primary node STUD-ID which stands for the entity "student". They can also be joined on the primary node STUD-NAME of the entity network STUDENTS with the primary node STUD-ID of the relationship network STUD-CRS, both stand for the same entity. The input query will be used to decide on which nodes the selected networks must be joined. As a rule of thumb the nodes corresponding to the attributes on which the relations are joined are used also to join the networks. However, any other node which stands for the same entity can be used for joining networks.

For the following two different queries the same network is created and the network is shown in Fig. 8.4.

```
SELECT  ..                        SELECT  ..
FROM    professors                FROM    students
WHERE   prof-name  IN             WHERE   stud-id  IN
      SELECT prof-name                  SELECT stud-id
      FROM   advisor,students           FROM   advisor,professors
      WHERE  students.stud-id =         WHERE  advisor.prof-name =
             advisor.stud-id                   professors.prof-name
        :                                  :
```

Fig. 7.5 Different queries which result in the same networks.

7.3 Sentence Generation

The sentence generator is dependent upon the structure of the query and the semantic networks. The lexicon, DD and the DB are sourses of knowledge which do not affect the form of the generated sentence. The only exception is the relation NOUNS of the lexicon. The basic structure of a sentence is: Subject NP followed by Verb followed by Object NP. The network and the algorithm selected based on the query type are used to create that basic structure. Every arc joins just two nodes, one of the nodes at the tail of the arc is used to build the subject NP and the other at the head of the arc is used for the object NP. The label of the arc (verb) connecting them is taken as the main verb of the generated sentence. From which node the generation process will start is determined by the selected algorithm. The generation will stop only when all the attributes in the query have participated in the generation of the response as specified by the selected algorithm. Some algorithms specify the generation of the first sentence only, and they require the network to be traversed further in order to generate the remaining sentences. For these algorithms the traversing of the network is done as follows:

a) Sentences will be generated with reference the attributes in the input query which have not already been used in the generation of other sentences, if any.

b) The nearest node must be found which has participated in the generation process to the node corresponding to the attribute for which we want to generate a response.

c) The network between these two nodes will be traversed generating sentences following the outgoing arc of the node which has been used in the generation process.

d) The node in the tail of the arc will be used to build the subject NP (it will be a proper pronoun) and the node in the head of the arc must be used to build the object NP.

These steps specify how the transitions to the other nodes in the network take place. The generated sentences (the discourse) are meaningful because the networks embed semantic knowledge.

The generation of the sentences proceeds as given below:

STEP 1: a) The PS-rules are fired to generate the deep structure sentence. It is done using the relationship NOUNS of the lexicon, the query, the algorithm and the network corresponding to the query.

b) The lexical insertion is performed in this step. The sourses for the lexical items are: 1) the lexicon; 2) the DD; and 3) the networks.

STEP 2: a) The cyclic transformational rules are applied in order. The optional rules may be ignored.

b) The post-cyclic transformational rules are applied in the given order. In this step the deep structure form of the sentence is transformed into the surface structure sentence.

STEP 3: Every symbol "V" with its values properly arranged either is replaced by the values obtained from the DB search or from the values in the predicates of the query.

7.4 Firing PS-Rules

The firing of the PS-rules and the lexical insertion result in the deep structure sentence. The algorithms presented in chapter VIII direct the structure of the generated sentence. These algorithms are used to fire most of the PS-rules and they also determine, in certain cases, the lexical items to be inserted. The first PS-rule has the initial symbol "S" on its left side. From the set of possible expansions of a rule one is selected. The rules to be fired are determined by using the following sourses of knowledge.

a) The network which corresponds to the query.

b) The structure of the query.

c) The algorithm which corresponds to the query.

d) The relation NOUNS of the lexicon.

The following example illustrates the firing of PS-rules.

```
SELECT  Stud-name
FROM    Students
WHERE   Prog-name = "M.Sc"
AND     Citizship = "Greek"
```

The SUBSQL query is the one given above:

<u>Fired Rule</u>	<u>Comments</u>
S --> % NP AUX VP %	- DB search is nonempty; query well defined. So there is no need for question or interogative sentence.
NP --> DET N PP PP	- The algorithm has specified the structure of the SUBJECT NP. Namely this query is one level query with an entity relation. The corresponding algorithm specifies that the subject noun must be the noun corresponding to the entity "student". The

	predicates must build PP postmodifying the subject noun. There are two predicates so two PPs are built.
	- DET: because from the lexicon the noun "student" has the feature +DET in the column NPDETERM.
N --> Student	- From the relation ENTITY-REL of the DD we get for the entry STUDENTS the corresponding entity which is "student".
DET --> ART	- There is no embedded sentence.
ART --> DEF	- From the relation NOUNS of the lexicon we find, "student" has the feature +DET in the NPDETERM column which stands for definite determiner.
PP --> PREP NP	- This PP corresponds to the predicate which is after the WHERE clause. There is no OR in the query so there is no need for EITHER, AND, OR;
PREP --> in	- From the relation PREP-ATTR of the DD we get "in" from the column SUBJPREP for the entry PROG-NAME.
NP --> DET N	- DET:It is used because we get the feature +DET for the noun

	"program" from the column PPDETERM of the relation NOUNS in the lexicon.
DET --> ART	- ART:Because we are not interested in generating embedded sentence. (For this example we could also generate an embedded sentence but in this case we decided not to generate embedded sentences.)
ART --> DEF	- DEF:The article for "program" is definite.
N --> V_program	- The form for the column PROG-NAME is V_A. That is taken from the column SUBJFORM of the relation ATTR-DOMNS; "A" has been replaced from the corresponding value of the column USED-MNG which is "program".
PP --> PREP NP	- This PP corresponds to the second predicate which is joined with an AND with the previous one; so the conjunctions EITHER, AND, OR are not needed.
PREP --> with	- From the column SUBJPREP of the relation PREP-ATTR of the DD we get from the entry CITIZSHIP this preposition.

NP --> N	- From the relation NOUNS of the lexicon we get from the entry CITIZSHIP of the column PPDETERM the feature -DET.
N --> V_Citizenship	- From the column SUBJFORM of the ATTR-DOMNS we get the form V_A for the entry CITIZSHIP. "A" has been replaced from the corresponding value of the column USED-MNG.
AUX --> T	- The generated sentences do not involve modals
T --> PRES	- The reference time for the relation STUDENTS is "present"; obtained from the Entity-Network.
VP --> V NP	- From the network for the entity student we get the features +ACT, -PERF, -PROGR from the frame corresponding to the node STUD-NAME. There is no embedded sentence in the object NP.
V -->be	- The reference verb for the column STUD-NAME is "be"; that from the network of this query.
NP --> DET N	- The algorithm pertinent to this query type has determined that the OBJECT NP must be the form in the attribute in the SELECT clause of

	this query.
	- From the column NPDETERM of the relation NOUNS of the lexicon we get the feature +DET for the noun "student".
N --> Student_V	- From the column OBJFORM in the the relation ATTR-DOMNS we get the form A_V for the entry STUD-NAME. "A" has been replaced with the value "student" from the column USED-MNG.
DET --> ART	- There is no embedded sentence.
ART --> DEF	- There is no relative clause. From the column NPDETERM of the relation NOUNS we get the feature +DET for the entry "student" which implies definite determiner.

7.5 Pronouns

The pronouns bind the sentences of a text or conversation together. The use of a pronoun in most cases is an act of reference. This means that the pronoun is intented to stand for a particular concept that has come before in the conversation. Consider the following sentences.

- The student with identification number 11000 has part time status. The student with identification number 11000 is registered in the bachelor program.

- The student with identification number 11000 has part time status. He is registered in the bachelor program.

The pronoun he has replaced the subject NP **"The student with identification number 11000"**, so pronouns provide abbreviated reference. The second set of sentences appears more natural. The pronoun does not carry much information about the referent object and so the burden is on the listener to identify the referent candidate that was either mentioned or implied by the preceding text.

In natural language analysis the problem is to identify the referenced items in the text. In natural language generation the problem is the generation of the appropriate pronouns for objects/concepts already mentioned. The solution to this problem in our system is based on the network structure and the fact that the generated responses deal with local discourse. It means that the generated sentences do not make references outside the context of the current query. For every query no matter what is its structure there is a corresponding network which might have been created by joining other networks. A **reference list** (R-List) is built as the generation of the response proceeds. Every time an arc is traversed a check is made in the R-List to decide whether a pronoun is needed to be generated. At the begining of the processing of a query the list is empty. Every entry of the R-list has:

a) The **node name** of the node which has participated in the generation process.

b) The **noun** which has been used for the description of that node. If the node is primary so it stands for an entity the noun corresponding to the entity will be in the list.

c) The **sex** of the noun. It is taken from the database.

d) The **number** (singular, plural) of the noun. The number depends on the number of items in the V(alue) part of the form of the attribute corresponding to that node. For example if an attribute appears in the predicates it is associated with one value, the number of the noun in a NP/PP built from that attribute will be singular. If an attribute appears in the SELECT clause and the DB search retrieves more than one tuple then the number of the noun in a NP/PP built from that attribute will be plural.

Every time an arc is traversed the R-list is checked to see whether that node has occured before. If it occured and the sex and number features agree then a pronoun is generated. If the node is new its description is stored in the R-list.

For example in Fig. 7.6 we have shown the generated sentences and the R-list for the following query.

```
SELECT   Course-no, grade
FROM     Stud-crs
WHERE    Stud-id = 99337
AND      Instructor = "V.Alagar"
```

Response sentence: The student with identification number 99337 has taken the courses with the numbers 793, 653 and 763.

	1st element	2nd element
NODE NAME	Stud-id	Course-no
NOUN	Student	Course
SEX	Male	Asexual
NUMBER	Singular	Plural

Response sentence: They have been given by the instructor V.Alagar

Stud-id	Course-no	Instructor
Student	Course	Instructor
Male	Asexual	Male
Singular	Plural	Singular

Response sentence: He has gotten grades A, B and B.

Stud-id	Course-no	Instructor	Grade
Student	Course	Instructor	Grade
Male	Asexual	Male	Asexual
Singular	Plural	Singular	Plural

Fig. 7.6 Illustration of the use and creation of the R-list

The relationship network for STUD-CRS is given in Fig. 7.2. The algorithm starts from the node STUD-ID of the network to generate the first sentence. The status of the

R-list after the generation of a sentence is shown below the corresponding sentence in Fig. 7.5. The generation of the second sentence makes use of the node COURSE-NO which has already been used, that is found from the R-list because of that it makes an anaphoric reference to that node. In the generation of the third sentence the node STUD-ID is involved which is already in the R-list and an appropriate anaphoric reference is made to it.

VIII. ALGORITHMS FOR GENERATION OF SENTENCES

8.1 The Algorithms

In this chapter several algorithms are presented and they can be classified based on:

a) The type of the SEQUEL query, such as whether it has one or two nested levels.

b) The number of relations in each nested level, whether it has one or two relations.

c) The type of the relations such as whether it pertains to entity or relationship relations.

There are three basic algorithms which are used also by other algorithms: a) The algorithm for one level queries with one entity relation; b) The algorithm for one level queries with one relationship relation; and c) The algorithm for one level queries without WHERE clause.

In order to generate proper responses, the algorithms take into consideration the following:

a) The type of attributes which exist in the query such as whether they are key or nonkey attributes and whether they are entity or nonentity attributes.

b) The place in the query where they appear, in the SELECT or in the WHERE clause.

c) The type of boolean operators with which the predicates are joined.

d) The network corresponding to the query which shows how the various nodes are connected their labels and frames.

e) The number of tuples which are retrieved from the DB search.

Certain nodes are met when the entity and relationship networks are traversed in the generation phase. The nodes which do not correspond to any attribute in the query generate indefinite NPs. For example in the response of the query no.1 in section 8.5 there are the indefinite NPs "a section" and "a class". They are generated because the nodes SECTIONS and CLASSES had to participate in the generation process but their corresponding attributes are not in the query.

8.2 Presentation of the Algorithms

For clarity of presentation we employ the following conventions. When we say that the attribute "A" is connected to the attribute "B" we mean the nodes corresponding to these attributes are connected by an arc in the network of the query. We use attributes instead of nodes because the algorithms operate on the query and at the same time they make reference to the network of the query. For each attribute in the query there is a node in its network. So there is one-to-one correspondence of the nodes in the network and the attributes in the query. We use four different symbols for labelling the tests and the actions involved in our algorithms in the following presentations:

a) Integer numbers. (Outer most level of nesting)

b) Upper case characters. :

c) Lower case characters. :

d) Latin numbers. (Inner most level of nesting)

In some algorithms there is the expression "identifies completely the entity". We mean that some entity attributes have the complete name of an entity e.g. the entity attribute STUD-NAME (student name) identifies completely the entity "student" while the entity attribute STUD-ID (student identification) is just a unique identification of that entity without giving any other information about the entity. Some algorithms have requirements that the nodes on which the networks will be joined must have "identical semantic meaning and syntactic representation". Consider the following two cases where the restriction applies.

a) Suppose we join the node COURSE-NO which stands for the entity "course". The entity "course" may be used to identify the entity "section" in one network and in the other network the node for the entity "course" may not identify the entity "section". In a case like this the two nodes do not have the "same meaning and syntactic representation".

b) All the arcs (ingoing,outgoing) on the node DAY in the relationship network SEC-DAYS have the label PREP(on) which means that a NP built from that node must always be preceded by that preposition. The same does not apply to the entity "day" in the entity network DAY. The two nodes do not have "same meaning and syntactic representation". The node DAY in SEC-DAYS says about an event which happens on a specific day while the entity DAY in the entity network says something about the days. There is semantic difference.

8.3 Insertion of the Tuples from the DB Retrieval into the Generated Response

The tuples which are retrieved from the DB are inserted in the generated response. They replace the "V" part in the forms of the columns which appear in the SELECT clause of the query. In case of nested queries the SELECT clause of the outer subquery is used. The following are applied only when more than one tuple are retrieved from the DB. There are two cases to distinguish.

a) Queries with one attribute in the SELECT clause: After the generation of the sentence the "V" part of the form corresponding to the attribute in the SELECT clause is replaced by the values in the tuples retrieved from the DB. The arrangement of the values must be in an order as it is specified by $\textcircled{1}$, v_i i=1,n stands for the value in the i-th tuple.

$$v_1, v_2, \ldots, v_{n-1} \text{ and } v_n \qquad \textcircled{1}$$

In this case identical entries may be ignored without any problem. For example let's assume that the DB search results in the following tuples (211, 231, 241) for this query.

<u>Query no.1</u>

```
    SELECT  course-no
    FROM    stud-crs
    WHERE   grade = "F"
```

<u>Response</u>: Grade fail has been gotten by some students. They have taken the courses with the numbers 211, 231 and 241.

b) <u>Queries with more than one attributes in the SELECT clause</u>: Again in this case the "V" part in the form of each attribute which is in the SELECT clause will be replaced by the set of values in the corresponding column of the retrieved relation. The following rule must be applied in order to avoid the generation of ambiguous or incorrect sentences.

All the values, no matter whether they appear once or more in the set of values of an attribute must appear in the generated response.

Applying the above rule there is one-to-one correspondence in the values of the columns in the SELECT clause. In this way the listener/reader of the response can understand the correspondence of the values. The arrangement of the values for each column will have the same form as above.

$v_1, v_2, \ldots, v_{n-1}$ and v_n

The following example illustrates the above condition.

Query no.2

```
   SELECT   class-no, course-no
   FROM     sec-class
   WHERE    section = "Al"
   AND      campus = "SGW"
```

Consider that the DB search has retrieved four tuples for the query no.2.

RETRIEVED TUPLES

class-no	course-no
925	211
925	231
803	322
925	322

In this set of tuples each tuple is unique but in each attribute a value appears more than once. If we remove from each column the entries which appear more than once and set the values for each column properly in the "V" part of its form the generated sentence will be ambiguous. It is shown in the following response.

Ambiguous Response

The sections with identification Al of the courses with the numbers 211, 231 and 322 are given in the classes with the numbers 925 and 803 on Sir George William campus.

From the above response it is not clear which course with it's section name Al is given in the class room 925. There is no one-to-one mapping of the courses and the classes. But even if we had one-to-one mapping it would not correspond to the actual results. For example, consider that the last tuple is changed to (816, 322) then there would be one-to-one mapping of the classes and the courses {(925, 803 and 816) - (211, 231 and 322)} but that mapping does not correspond to the actual one e.g. 803 is mapped into 231 which is not correct. The following response is precise but is awkward.

Unambiguous Response

The sections with identification A1 of the courses with the numbers 211, 231, 322 and 322 are given in the classes with the numbers 925, 925, 803 and 925 on Sir George William campus.

8.4 One Level Queries with one Entity Relation

The algorithm for this type of queries makes use of the query and the entity network. All the generated responses involve the entity since all the attributes are related to it. Most sentences have PPs embedded into the subject NP postmodifying the subject of the sentence. A PP is generated using the form of the corresponding attribute from the column SUBJFORM of the relation ATTR-DOMNS. The preposition is taken from the corresponding entry of the column SUBJPREP of the relation PREP-ATTR of the DD.

This research showed that most of the attributes are associated with the preposition with, that can be seen by examining the examples presented in this section. This can be explained because "with" implies a kind of possession denoted by the verb **have** [20]. In the E-R model , an entity possesses certain attributes, that possession is denoted in the generated response. There are a few other prepositions (e.g. in, on,..) which are used to indicate other kinds of relationships of the noun corresponding to the predicate with the subject noun (entity). The fourth example shows that in certain cases relative clauses can be used interchangably with PPs. The relative clause "who are registered in the bachelor program" can be interchanged with the PP, "in the bachelor program". However, the response with the relative clause is preferred because it qualifies the subject noun in a better way.

8.4.1 Algorithm

a. Subject NP

```
1. If there is only one predicate whose attribute is key and
   identifies completely the entity then
   A. SUBJECT will be the form of that attribute taken
      from the appropriate entry of the column SUBJECT
      of the relation ATTR-DOMNS of the DD;
2. else
   A. SUBJECT will be the noun associated to the entity
      relation. It is taken from the relation
      ENTITY-REL of the DD.
```

B. <u>If</u> there is no OR operator in the WHERE clause <u>then</u>

a. The predicates must follow the subject as postmodifiers of it. For each predicate a PP is built and is added at the end of the subject NP.

<u>Optional</u> step

b. <u>If</u> the frame of a verb has the feature -ACT and that verb belongs to a node whose attribute is in the predicates <u>then</u>

i. A relative clause is built instead of a PP. It must follow right after the subject of the sentence. The PPs must come after the relative clause.

C. <u>else</u>

a. For each predicate a PP is built. The PPs are joined using the EITHER_OR* conjunctions as follows. EITHER must precede the PPs built from the predicates which are joined in the query with AND algebraic operators. These predicates must follow right after the WHERE keyword. The first predicate is included which is not preceded by any algebraic operator. For each predicate preceded by an OR operator its PP must be preceded by an OR conjunction;

----

b. Object NP

3. If there is one column in the SELECT clause then
    - A. OBJECT will be the form of the attribute in the SELECT clause taken from the column OBJFORM of the relation ATTR-DOMNS. If the corresponding entry for that column in the column OBJPREP of the relation PREP-ATTR of the DD is nonempty the preposition it has must precede the object form of the attribute to build a PP.
4. Else
    - A. The number of sentences depends on the number of different verbs on which the columns on the SELECT clause are associated;
    - B. If there is a key column in the SELECT clause then
        - a. The object NP of the first sentence must be built from that column;
    - C. If two or more columns in the SELECT clause have the same reference verb then
        - a. Generate a sentence using one of the columns applying 1, 2 and 3 steps of this algorithm. If there is a Key column in a group use it first.
        - b. Add at the end of that sentence the object form (taken from the column OBJFORM of the relation ATTR-DOMNS of the DD) of the other attributes, following the formula $\boxed{2}$. Where n is the number of columns with the same reference verb.

OFORM(i) is the object form of the i-th attribute.

SENTENCE (,OFORM(i))* i=2,..,n (2)

D. The generation of the object NP of the attributes which do not have the same verb with other attributes in the SELECT clause must follow part 3 of this algorithm.

E. The subject NP of the first sentence is based on the previous algorithm. The subject NPs of the other sentences to be generated are pronouns. They refer to the subject of the first sentence.

In the following we present some queries and the generated sentences based on the above algorithm:

<u>Query no.1</u>

```
   SELECT  prog-name, status
   FROM    students
   WHERE   stud-name = "G.Bellos"
```

<u>Response</u>: The student G.Bellos is registered in the bachelor program. He has part-time status.

<u>Query no.2</u>

```
   SELECT  status, phone, stud-name
   FROM    students
   WHERE   prog-name = "M.Sc"
   AND     citizship = "Greek"
```

<u>Response</u>: The students in the Master program with Greek citizenship are S.Koutsianos and D.Katsiapi. They have full time and full time status, phone numbers 33678 and 89679.

Query no.3
```
    SELECT   stud-name
    FROM     students
    WHERE    prog-name = "Ph.d"
    OR       status = "F/T"
    OR       citizship = "Canadian"
```

Response: The students either in the doctoral program or with full time status or with Greek citizeship are L.Winsor and T.Dally.

Query no.4
```
    SELECT   stud-name
    FROM     students
    WHERE    prog-name = "B.Sc"
    AND      status = "P/T"
```

Response 1: The students who are registered in the bachelor program with part time status are P.Rae and V.Wong. (This response is generated if the optional step is followed)

Response 2: The students in the bachelor program with part-time status are P.Rae and V.Wong.

8.5 One Level Queries with one Relationship Relation

The algorithm for relationship relations is based on the query and the relationship network which corresponds to the relation referenced in the query. It checks the place where the key and nonkey attributes appear in the query as well as which attributes are connected and what type of arc joins them and then decides about the sentence to be generated. Non-entity attributes are handled in the same way as the entity attributes. They may be used to build the object NP or the subject NP of a sentence, their form is used as subject NP or object NP. In the case of entity attributes the entity corresponding to the attribute is used to build the subject/object of the sentence as in the case of the entity relations. The noun corresponding to the entity is postmodified by a PP built from the form of that entity-attribute. When there is a dependency link between two entities then after the PP/NP of the dependent attribute the phrase which indicates the dependency must follow, it is built using the next rule.

> The label of the dependency arc which is a preposition must be followed by a NP. That NP is built from the node corresponding to the regular entity. The generated PP must come after the NP built from the node corresponding to the weak entity.

The following two examples illustrate the above. The entity "section" depends on the entity "course". This dependency is expressed in the generated sentence by the prepositional phrase "of the course". The attribute "course-no" in the WHERE clause stands for the entity "course". Because the "course-no" is an entity attribute, the PP "of the course" is followed by another PP "with the (course) number 722". The noun "course" in the second PP is deleted by a T-rule to make the sentence more readable.

Query no.1

```
   SELECT  campus
   FROM    sec-class
   WHERE   course-no = 722
```

RESPONCE: A section of the course with the number 722 is given in a class on Loyola campus.

Query no.2

```
   SELECT  stud-id
   FROM    stud-crs
   WHERE   grade = "F"
```

RESPONCE: Grade fail has been gotten by the students with identification numbers 77660, 32733 and 60037.

The second query illustrates the generation of the subject NP from a nonkey attribute. The subject form SUBFORM from the relation ATTR-DOMNS of the attribute GRADE has been used as subject NP. For the subject form of the attribute GRADE the successive steps which transform it into the final form are: (A_V) --> (grade_V) --> (grade_fail).

8.5.1 Algorithm

```
1. If there is an attribute which stands for an entity in
   the WHERE clause then
   A. SUBJECT: It is built from the attribute which stands
      for the entity;
   B. If there is an attribute which stands for an entity
      in any clause (SELECT, WHERE) and it is connected
      with an arc which is not a depencency arc with the
```

entity of the subject <u>then</u>

a. OBJECT: it is built from that entity.

C. <u>else if</u> there is a nonkey column in any clause (SELECT, WHERE) and it is connected with the subject entity attribute with a labeled arc <u>then</u>

a. OBJECT: It is the form of that nonkey attribute.

D. <u>else if</u> there is an entity which does not appear in the query and the subject entity is joined with it with a labeled arc. The arc must not be dependency arc. <u>then</u>

a. OBJECT: It is built from that entity.

E. <u>else</u>

a. OBJECT: The form of a nonentity attribute is used. The node corresponding to that nonentity attribute must be joined with a labeled arc with the node corresponding to the subject entity.

2. <u>else if</u> there is a nonkey attribute in the WHERE clause with a labeled arc going to an attribute which stands for an entity and which is somewhere in the query <u>then</u>

A. SUBJECT: it is the form of the nonkey attribute.

B. OBJECT: It is built from the entity pointed by the head of the arc.

3. <u>else if</u> there is an entity attribute in the SELECT clause (Higher priority is given to an attribute which is joined with any kind of link with an attribute in the

```
        WHERE clause) then
        A. SUBJECT: It is built from that entity attribute.
        B. if there is a key attribute which is somewhere in
           the query and it is joined with the subject entity
           then
           a. OBJECT: It is built from that entity
        C. else if there is a nonkey attribute which is in the
           query and the subject entity is joined with a
           labeled arc then
           a. OBJECT: It is built from the form of the nonkey
              attribute.
        D. else
           a. OBJECT: It gets the attribute corresponding to
              a node from the relationship network with
              which the subject node is joined with a
              labeled arc. The attribute corresponding to
              that node is not in the query.
    4. else
        A. SUBJECT: For the node corresponding to an attribute
           in the query (it is prefered attribute in the
           WHERE than in the SELECT clause) it finds an
           entity node corresponding to a key attribute with
           which it has any kind of link (e.g. dependent,
           unlabeled arc). That key attribute is used to
           build the subject;
        B. If there is a nonkey attribute in any clause
           (SELECT, WHERE) with which the subject attribute
```

is joined with a labeled arc <u>then</u>

a. OBJECT: It is built from that attribute.

C. <u>else if</u> there is an attribute which is not in the query and the subject attribute is joined with it (key attribute has higher priority than nonkey) <u>then</u>

a. OBJECT: It is built from that attribute;

5. <u>If</u> any column has been left out (it is in the query but has not been used in any sentence <u>then</u>

A. A new sentence will be built using the algorithm in section 7.3.

The following examples illustrate some cases of the above algorithm.

<u>Query no.1</u>

```
   SELECT campus
   FROM   sec-class
   WHERE  course-no = 211
   AND    section = "A1"
```

RESPONCE: The section with identification A1 of the course with the number 211 is given in a class on Loyola campus.

<u>Query no.2</u>

```
   SELECT  course-no
   FROM    sec-class
   WHERE   class-no = 309
   AND     campus = "Loy"
```

RESPONCE: The class with the number 309 on Loyola campus is used for a section of the course with the number 211.

Query no.3
```
   SELECT  section, campus
   FROM    sec-class
   WHERE   course-no = 211
```

RESPONCE: The section with identification A1 of the course with the number 211 is given in a class on Loyola campus.

Query no.4
```
   SELECT  grade
   FROM    stud-crs
   WHERE   course-no = 722
   AND     instructor = "D.Desai"
```

RESPONCE: The course with the number 722 has been given by the instructor D.Desai. It has been taken by some students. They have gotten grades As, Bs and fails.

Query no.5
```
   SELECT  instructor
   FROM    stud-crs
   WHERE   stud-id = 52535
```

RESPONCE: The student with the identification number 52535 has taken some courses. They have been taught by the instructors D.Desai, R.De Mori and R.Shinghal.

8.6 One Level Queries with two Relations

The generation of the responses to queries with two relations has been simplified with the definition of the join operation on the networks. We distinguish two types of queries with two relations based on the type of the relation.

a) Queries with one relationship relation and one entity relation.

b) Queries with two relationship relations.

There are no major differences between the two types of queries. In order to generate a response for queries of this form the algorithm for one level queries with an entity relation or one level queries with a relationship relation are used.

8.6.1 Algorithm

```
1. If there is one entity relation and one relationship
   relation in the query then
   A. Generate a partial response using
      a. The part of the network which corresponds to the
         relationship network.
      b. The attributes of the relationship relation which
         exist in the query.
      c. The algorithm for one level queries with one
         relationship relation.
   B. Genarate another partial response using:
      a. The part of the network which corresponds to the
         entity relation.
      b. The attributes of the entity relation which exist
         in the query.
      c. The algorithm for one level queries with one
         entity relation.
   C. Make use of the common entity to join the partial
```

```
      responses generated in steps 1 and 2. The subject
      in the partial response generated by step 1.B must
      have an anaphoric reference to the NP generated in
      step 1.A from the common entity node.
2. else if there are two relationship relations in the query
   then
   A. Treat the new network as a relationship network;
      apply the algorithm for one level queries with one
      relationship relation.
   B. If the relationship network corresponding to the one
      relation has a dependency arc then
      a. Start the generation from that part of the
         network.
   C. else
      a. Start the genaretion from any node of the
         network as it is specified by the algorithm in
         section 8.5.1.
```

The responses for the following queries are based on the above algorithm to illustrate it. The first two examples are based on the joined network presented in fig. 7.4.

<u>Query no.6</u>

```
   SELECT   stud-id, citizship
   FROM     students, stud-crs
   WHERE    students.stud-id = stud-crs.stud-id
   AND      instructor = "D.Desai"
```

RESPONCE: The students with identification numbers 77733 and

66684 have taken some courses. They have been taught by the instructor D.Desai. They have Canadian and Greek citizenships.

The algorithm for the relationship relation generates the following partial response.

- The students with identification numbers 77733 and 66684 have taken some courses. They have been taught by the instructor D.Desai.

The algorithm for the entity network generates the partial response.

$\left\{\begin{array}{c}\text{The students}\\ \text{They}\end{array}\right\}$ have Canadian and Greek citizenships.

Its subject is changed into a pronoun.

<u>Query no.7</u>

```
SELECT  grade, stud-name, sex
FROM    students, stud-crs
WHERE   students.stud-id = stud-crs.stud-id
```

RESPONCE: Some students have gotten grades A and Fail. They are the students G.Bellos and T.Dally. They have male and male sex.

Fig. 8.1 shows the network resulted from the join of the ADVISOR and DEPT-PROF relationship networks. It is used for the illustration of some more examples.

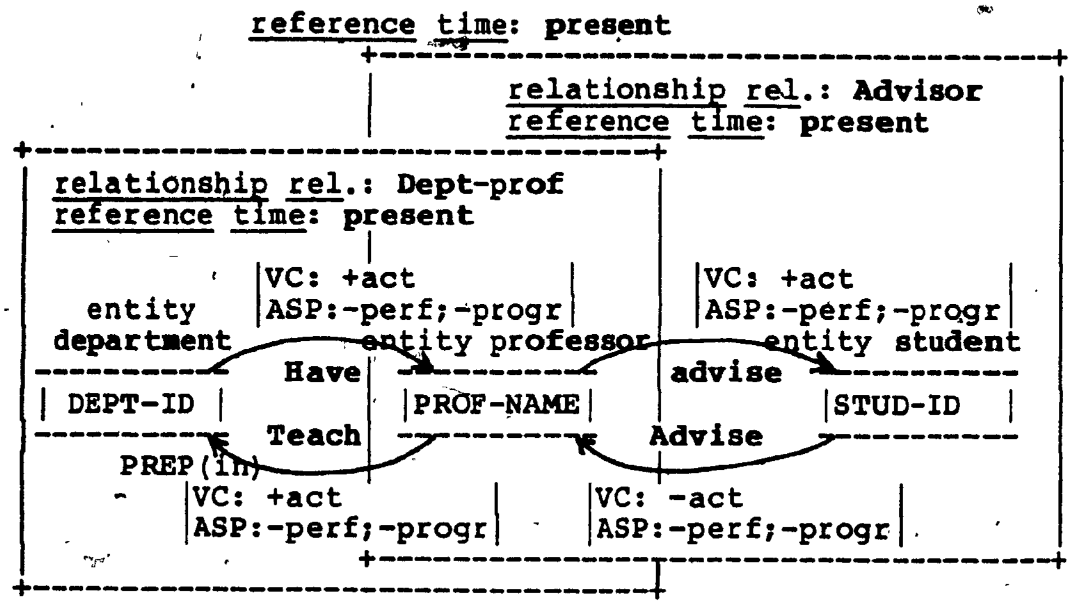


Fig. 8.1 The network resulted from the join of the relationship networks DEPT-PROF and ADVISOR

Query no.8
```
    SELECT   prof-name
    FROM     dept-prof, advisor
    WHERE    dept-prof.prof-name = advisor.prof-name
    AND      stud-id = 77773
    AND      dept-id = "comp"
```

RESPONCE: The department with department identification COMP has the professor with the name J.Opatrny. He advises the student with the identification number 77733.

Query no.9
```
    SELECT   stud-id
    FROM     dept-prof, advisor
    WHERE    dept-prof.prof-name = advisor.prof-name
    AND      dept-id = "COMP"
```

RESPONCE: The department with identification COMP has some professors. They advise the students with the identification numbers 77766, 33377 and 33329.

8.7 Algorithm for one Level Queries Without WHERE Clause

Here we consider one level queries without WHERE clause. They are classified into queries containing either an entity relation or a relationship relation.

Algorithm

```
1. If the query has an entity relation in the FROM clause
     then
     A. If there is an attribute which stands for an entity
          and identifies completely that entity then
          a. SUBJECT will be a definite NP. It is built
               from the entity corresponding to the entity
               relation. The noun which stands for the
               entity is taken from the relation ENT-ATTR of
               the DD.
          b. OBJECT will be built from the form of that
               entity.
     B. else
          a. SUBJECT will be an indefinite NP built from the
               noun corresponding to the entity.
          b. OBJECT will be the form of an attribute in the
               SELECT clause.
               i. if there is any key attribute in the
                    SELECT clause then
                       use it first;
```

```
C. if there are more than one attributes in the SELECT
   clause with the same reference verb then
   a. The forms of these attributes are appended
      after the object NP of the generated sentence
      which has the same verb. This is done by
      applying the formula ② in section 8.4.1.
D. Repeat
   a. If there are more attributes in the SELECT
      clause then
      i. SUBJECT will be an anaphoric reference to
         the entity.
      ii.OBJECT will be the form of an attribute
         from the SELECT clause followed by the
         form of the attributes with the same
         reference verb. It is done as it is
         specified by the formula ② in section
         8.4.1.
   until there is not left any attribute in the SELECT
      clause;
2. else if the query has a relationship relation in the FROM
   clause then
   A. If the query has one attribute in the SELECT clause
      then
      a. if the attribute is entity then
         generate a sentence having as:
         i. SUBJECT the entity attribute in the SELECT
            clause..
```

```
        ii.OBJECT another entity with which it is
            linked by a labeled arc. An indefinite
            NP is built.
    b. else if the attribute is not an entity then
        i. if that attribute has a labeled arc leaving
            from it then
            SUBJECT will be the form of that
                attribute
        ii. else that attribute must be connected
            with an unlabeled arc with an attribute.
            The two attributes must be related in a
            way. SUBJECT will be the phrase which
            can be built from the relation of the
            attributes as it is specified in the
            network. In this case the attribute
            which has the ingoing and outgoing arc
            is considered as the SUBJECT of the
            response.
        iii. OBJECT: It is built from the node to
            which there is an arc from the node
            corresponding to the subject attribute.
B. else if there are more than one attributes in the
    SELECT clause of the query then
    a. If there is in the query one attribute which
        stands for an entity then
        i. SUBJECT: That attribute is used to build
            the subject NP.
```

b. <u>else</u> there must be a nonentity attribute in the query.

   i. SUBJECT: The form of that attribute is used to build the subject NP;

c. <u>If</u> there is an entity node with which the subject node is joined with a labeled arc and the attribute corresponding to that node is in the query <u>then</u>

   i. OBJECT: It is built from the attribute corresponding to that node.

d. <u>else</u> <u>if</u> there is a nonentity node with which the subject node is joined with a labeled arc and the attribute corresponding to that node is in the query <u>then</u>

   i. OBJECT: It is built from the attribute corresponding to that node.

e. <u>else</u> <u>if</u> there is an entity node with which the subject node is joined with a labeled arc and the attribute corresponding to that node is not in the query <u>then</u>

   i. OBJECT: It is built from the attribute corresponding to that node.

f. <u>else</u>

   i. OBJECT: It is built from the attribute corresponding to the nonentity node which is not in the query. The subject node is connected with a labeled arc with that node.

3. If any attribute exists in the query which has not been used in the generation of any sentence, normaly that attribute should not stand for an entity then

   A. The algorithm in section 7.3 is applied.

In the following examples, the first three illustrate responses to queries with entity relations while the last three relate to queries with relationship relations.

Query no.1
```
   SELECT  stud-id, phone
   FROM    students
```

RESPONCE: Some students have identification numbers 77733 and 88833, phone numbers 77345 and 66345.

Query no.2
```
   SELECT  prog-name, citizship, stud-name
   FROM    students
```

RESPONCE: The students are G.Bellos and T.Dally. They are registered in the bachelor and master program. They have Greek and Canandian citizenship.

Query no.3
```
   SELECT prog-name
   FROM   students
```

RESPONCE: Some students are registered in the bachelor, master and doctoral program.

Query no.4
```
   SELECT  class-no
   FROM    sec-class
```

RESPONCE: The classes with the numbers 925 and 803 on some campuses are used by some sections of some courses.

Query no.5
```
   SELECT  grade, course-no, instructor
   FROM    stud-crs
```
RESPONCE: The courses with the course numbers 753 and 731 have been taught by the instructors V.Alagar and J.Opatrny. They have been taken by some students. They have gotten grades A and A.

Query no.6
```
   SELECT  section, campus, class-no
   FROM    sec-class
```
RESPONCE: The sections with the section identifications AA and AB of some courses are given in the classes with the numbers 925 and 923 on Sir George William and Sir George William campus.

8.8 Nested Queries

The study of the nested queries is based on the number of relations in each level and the type of these relations. The outer level of these queries always has one relation. Queries are divided into two categories:

1) Queries with one relation in both levels;

2) Queries with two relations in the inner level.

We always start to generate responses from the inner level (subquery) unless the algorithm explicitly changes this order.

We consider the following groups of queries, for every case a different algorithm has been developed.

a. Queries with one relation in each level but different type of relation. The inner subquery can have an entity relation and the outer subquery a relationship relation or vice versa.

b. Queries with one relationship relation in both levels.

c. Queries with the same relation in both levels.

d. Queries with two relations in the inner level.

If we exclude the cases c) and d), queries which belong to other cases can always be mapped into equivalent one level queries of two relations. Because of this feature the nested queries with one relation in each level are handled in a similar way as the one level queries with two relations. The responses they generate have small differences. The attributes on which the joining is performed are the attributes in the SELECT clause of the inner subquery with the corresponding attributes of the outer subquery in the WHERE clause. The following example illustrates the above.

```
                                            Equivalent
    Two  Level Query                   One level Query

SELECT stud-id,sex,prog-name SELECT stud-id,sex,pro-name
FROM   students              FROM   students,stud-crs
WHERE  stud-in  IN           WHERE  students.stud-id =
     SELECT stud-id                 stud-crs.stud-id
     FROM   stud-crs         AND    grade = "A"
     WHERE  grade = "A"
```

8.8.1 Algorithm for Queries with Different Type of Relations in Each Level

1. If the relation in the inner query is an ER and the entity nodes on which the two networks will be joined have identical semantic meaning and syntactic representation then
   - A. SUBJECT is built according to the algorithm for one level queries with an ER using the attributes which belong to the ER.
   - B. If an attribute which stands for the common entity exists in the SELECT clause of the outer query then
     - a. OBJECT will be built according to the algorithm for one level queries with an ER (OBJECT will be the form of that attribute).
   - C. else if there is an entity-attribute in the SELECT clause of the outer query which is pointed by a labeled arc from the common entity (entity on which the join has been done). That can be found from the joined network then
     - a. OBJECT will be built according to the algorithm for relationship relations using that entity. (OBJECT will be the entity itself

postmodified by a PP built from the form of that entity-attribute).

D. <u>else if</u> the node corresponding to the common entity-attribute is joined with an outgoing labeled arc with an entity-node whose corresponding attribute is not in the query. <u>then</u>

   a. OBJECT: It is built from the entity-attribute corresponding to that entity node. The object NP must be an indefinite NP.

E. <u>else</u> there must be an attribute (not standing for any entity) with which the subject attribute is joined with a labeled outgoing arc. This attribute will be used to build the OBJECT NP. The object NP will be an indefinite NP;

<u>Outer Query</u>

F. <u>If</u> any attribute in the SELECT clause has not been used in the response <u>then</u>

   a. Generate responses using the algorithm for one level queries with a RR.

   b. Anaphoric references will be made to the common entity;

2. <u>else</u> <u>if</u> the relation in the inner query is an ER and the entity nodes on which the two networks will be joined have not identical semantic meaning and syntactic representation (the node corresponding to the RN has the restrictions on its meaning and its syntactic represenation) <u>then</u>

```
        A. Generate responses for each subquery seperately.
        B. Start generating from the RN because it has the
           entity-node with the syntactic and semantic
           restrictions.
        C. Since the outer query has a RR apply the algorithm
           for queries without WHERE clause with a RR.
        D. For the inner query apply the following algorithm.
           a. SUBJECT will be an anaphoric reference to the
              common entity on which the two networks have been
              joined.
           b. Consider that the predicates are attributes in
              the SELECT clause with the given values.
           c. Generate responses to this restructured query
              applying the algorithm of ER queries without
              WHERE clause.
     3. else if the relation in the inner subquery is a RR and
        the relation in the outer one an ER then
        A. Generate a response for the inner subquery using:
           a. The part from the joined network which
              corresponds to the relationship network (RN)
           b. The attributes in the SELECT clause of the inner
              subquery must be in the response.
              i. If they are in the SELECT clause of the outer
                 subquery then generate definite NPs for
                 these attributes.
              ii. else generate indefinite NPs.
           c. Trace the network generating sentences until the
```

entity (key attribute) on which the joining is performed is involved in a sentence and not predicate has been left out.

d. Use the algorithms for one level queries with a RR.

B. Generate a response for the outer query using:

a. SUBJECT will be an anaphoric reference to the common entity.

b. OBJECT will be created according to the algorithm for one level queries with an ER.

We give below some queries and the responses which are generated by using the above algorithm.

<u>Query no.1</u>
```
    SELECT  grade, course-no, stud-id
    FROM    stud-crs
    WHERE   stud-id  IN
           SELECT  stud-id
           FROM    students
           WHERE   citizship = "Greek"
           AND     prog-name = "M.sc"
```

RESPONCE: The students with Greek citizeship in the master program have identification numbers 77773 and 77774. They have taken the courses with the numbers 622 and 722. They have gotten grades A and C.

<u>Query no.2</u>
```
    SELECT  course-no
    FROM    stud-crs
    WHERE   stud-id  IN
           SELECT  stud-id
           FROM    students
           WHERE   stud-name = "G.Bellos"
```

RESPONCE: The student G.Bellos has taken the courses with the numbers 622 and 722.

Query no.3

```
SELECT  stud-id, prog-name
FROM    students
WHERE   stud-id IN
      SELECT  stud-id
      FROM    stud-crs
      WHERE   grades = "A"
```

RESPONCE: Grade A has been gotten by the students with the identification numbers 77773 and 88884. They are registered in the master and doctoral program.

Query no.4

```
SELECT  status, citizship
FROM    students
WHERE   stud-id IN
      SELECT  stud-id
      FROM    stud-crs
      WHERE   instructor = "J.Opatrny"
```

RESPONCE: Some students have taken some courses. They have been given by the insrtuctor J.Opatrny. They have full-time and full-time status, Greek and Canadian citizenship.

8.8.2 Algorithm for Queries with one Relationship Relation in each Level

1. If the nodes on which the networks will be joined have the same semantic and syntactic representation then
   A. Consider the joined network as a RN and generate sentences using the algorithm for one level queries with a RR.

```
    B. Start generating responses from the inner subquery.
2. else if (the nodes on which the networks will be joined
   have different semantic and syntactic representation)
   and (the RR which has the attributes with the
   restrictions is in the inner subquery) then
    A. Start generating responses from the inner subquery.
    B. For each (inner/outer) subquery generate responses
       using only that part of the network which
       corresponds to the RR of that subquery. The
       algorithm for one level queries with a RR must be
       used.
    C. For the outer subquery consider the attribute in the
       WHERE clause as predicate.
3. else
    A. Start generating responses from the outer subquery.
    B. Use as predicates the attributes in the WHERE clause
       of the outer subquery in order to generate the
       response for that subquery.
    C. For the inner subquery consider the predicates as
       attributes in the SELECT clause.
    D. For each (inner/outer) subquery generate responses
       using only the part of the network which corresponds
       to the RR of that subquery. The algorithm for one
       level queries with a RR must be used;
    E. SUBJECT NP for the response of the inner subquery
       (at least for the first sentence) will be an
       anaphoric reference to the common entity;
```

```
4. If we start generating responses from the inner subquery
   then
     A. If any of the entity-attributes on which the two
          relations will be joined appears either in the
          SELECT clause of the outer subquery or as
          predicate in the inner subquery then
          a. Generate definite NP for that attribute.
     B. else if it does not then
          a. Generate indefinite NP for that attribute.
5. else if we start generating responses from the outer
   subquery then
     A. If any of the entity-attributes on which the two
          relations will be joined appears in the SELECT
          clause of that subquery (outer) then
          a. Generate definite NP for that predicate
     B. else if it does not then
          a. Generate indefinite NP for that attribute.
```

The following examples illustrate the above algorithm. They use the network in fig. 8.2. The frame of each verb is shown in fig. 8.3. The number near each arc in fig. 8.2 points to the frame for that arc in fig. 8.3.

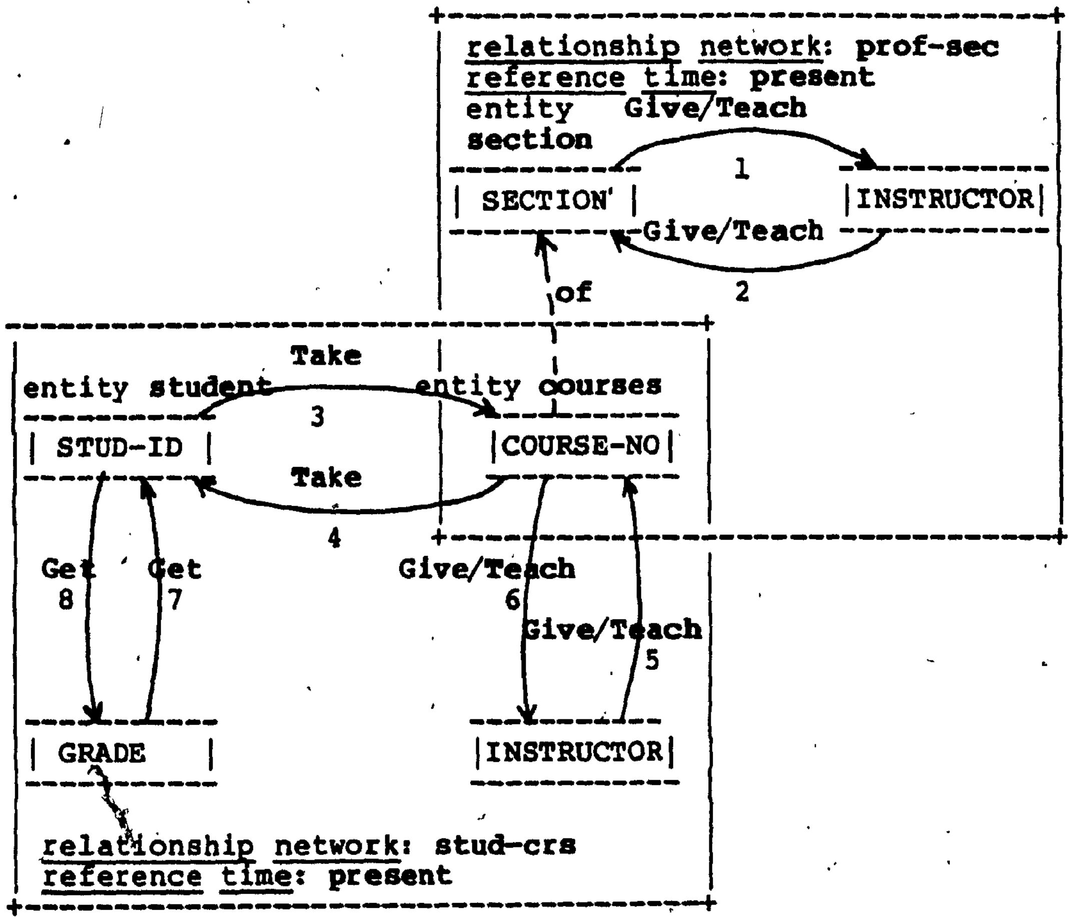


Fig. 8.2 The network resulted from the join of the relatioship networks STUD-CRS and PROF-SEC.

1.	VC: -act ASP:-perf;-progr	2.	VC: +act ASP:-perf;-progr
3.	VC: +act ASP:+perf;-progr	4.	VC: -act ASP:+perf;-progr
5.	VC: +act ASP:+perf;-progr	6.	VC: -act ASP:+perf;-progr
7.	VC: -act ASP:+perf;-progr	8.	VC: +act ASP:+perf;-progr

Fig. 8.3 Frames of the verbs of the RN in fig. 8.2.

Query no.1
```
   SELECT  section, course-no
   FROM    prof-sec
   WHERE   course-no   IN
         SELECT  course-no
         FROM    stud-crs
         WHERE   instructor = "J.Opatrny"
```

RESPONCE: The sections with section identifications AA and AA of the courses with the course numbers 743 and 731 are given by some instructors. They have been given by the instructor J.Opatrny.

Query no.2
```
   SELECT  instructor
   FROM    stud-crs
   WHERE   course-no  IN
         SELECT  course-no
         FROM    prof-sec
         WHERE   instructor = "V.Alagar"
```

RESPONCE: Some sections of some courses are given by the instructor V.Alagar. They have been taught by the instructors R.Shinghal and C.Suen.

8.8.3 Nested Queries with the Same Relation in Both Levels

There are queries with two level nesting, each level refering to the same relation. For example consider the request:

"Find the names of the students who are in the same program as the student G.Bellos."

One level queries with only the relation STUDENTS can not answer this question, but the following nested query can answer it.

```
SELECT   stud-name
FROM     students
WHERE    prog-name IN
      SELECT  prog-name
      FROM    students
      WHERE   stud-name = "G.Bellos"
```

Nested queries with the same entity relation in the inner and the outer subqueries have the following meaning.

"They find the values of some attributes (attributes in the SELECT clause of the outer query) which have the same values in some attributes (attributes in the WHERE and SELECT clause of the outer and inner query respectively) with the entity which is specified in the predicates of the inner query."

The queries of this category have the following common structural features.

a. The inner subquery must have as predicate an attribute which stands for an entity.
b. The common attributes on which the two subqueries will be joined must be nonkey attributes.

The noun in certain object NPs generated by the following algorithm needs to be preceded by the adjective **same**. This adjective can not be inserted by the PS-rules because of that a special routine must be written to do it.

8.8.3.1 Algorithm

1. If the relation in the inner and outer subquery have the same entity relation then
   - A. Generate the response first for the inner subquery.
   - B. The predicates must have a key attribute which can identify the entity either completed or partly.
   - C. Generate the response for the inner subquery using the algorithm for one level queries with an ER. For the attributes in the SELECT clause generate indefinite NP.
   - D. Response for the outer subquery:
     - a. The attributes in the SELECT clause will be used to built the SUBJECT NP.
     - b. The atrributes in the WHERE clause will be used to built the OBJECT NP. The attributes in this clause do not have values and hence indefinite NP will be generated.
     - c. The algorithms for one level queries with an entity relations will be used to generate the response with the following assumptions.
       - i. As predicates will be treated the attributes in the SELECT clause with their values.
       - ii. The attributes which are required by the algorithm from the SELECT clause of a query are replaced by the attributes in the WHERE clause of the outer query.

iii. The noun in the OBJECT NP of the sentences generated for the outer query must be preceded by the adjective **same**.

2. <u>else</u> <u>if</u> the inner and outer subquery have the same relationship relation <u>then</u>

   A. Generate a response first for the inner subquery. The NPs generated from the attributes in the SELECT will be indefinite.

      a. Apply the algorithm for one level queries with a RR.

   B. Generate a response for the outer subquery assuming the following changes:

      a. The attributes in the SELECT clause will be used as predicates.

      b. The step of the algorithm which require attributes from the SELECT clause to generate the OBJECT NP must take the attributes from the WHERE clause of the outer subquery. Indefinite NPs are generated from these attributes.

      c. Trace the same RN as for the inner subquery. The noun in the object NP must be preceded by the adjective **same**.

      d. The algorithm for RR is used assuming these changes.

The following examples illustrate this algorithm.

Query no.1
```
   SELECT  stud-name
   FROM    students
   WHERE   prog-name  IN
         SELECT  prog-name
         FROM    students
         WHERE   stud-name = "G.Bellos"
```

RESPONCE: The student G.Bellos is registered in a program.

The students T.Dally and V.Molino are registered in the same program.

Query no.2
```
   SELECT  stud-name, phone
   FROM    students
   WHERE   citizship, sex IN
         SELECT  citizship, sex
         FROM    students
         WHERE   stud-id = 77777
```

RESPONCE: The student with identification number 77777 has a citizeship and a sex. The students with the names V.Molino and T.Dally and the phone numbers 77333 and 77666 have the same citizenship and sex.

Query no.3
```
   SELECT  instructor
   FROM    stud-crs
   WHERE   course-no  IN
          SELECT  course-no
          FROM    stud-crs
          WHERE   instructor = "C.Lam"
```

RESPONCE: The insrtuctor C.Lam has given some courses. The instructors V.Alagar and T.Bui have given the same courses.

Query no.4
```
   SELECT  class-no, campus
   FROM    sec-class
   WHERE   course-no  IN
          SELECT  course-no
          FROM    sec-class
          WHERE   class-no = 925
```

RESPONCE: The class with the class number 925 on some campus is used by some sections of some courses. The classes with the class numbers 803 and 314 on Sir George William and Loyola campus are used by some sections of the same courses.

8.8.4 Nested Queries with two Relations in the Inner Subquery

The nested queries with two relations in the inner subquery are divided into the following groups.

a. Queries with an ER in the outer subquery and with an ER and a RR in the inner subquery.

b. Queries with an ER in the outer subquery and two RRs in the inner subquery.

c. Queries with a RR in the outer subquery and with an ER and a RR in the inner subquery.

d. Queries with a RR in the outer subquery and with two RRs in the inner subquery.

The algorithm for these type of queries uses the concepts and the algorithms for one level queries and nested queries with one relation in each subquery. The networks of the relations are combined into one applying the join operation.

8.8.4.1 Algorithm

a. Generation of the Response for the Inner Subquery

```
1. If an attribute in the SELECT clause of the inner
      subquery does not appear either in the predicates (of
      the inner subquery) or in the SELECT clause of the
      outer subquery then
      A. Generate indefinite NP for these attributes
2. else
      A. Generate definite NP;
```

```
1. If there is no relation in the inner subquery with
      dependency link then
      A. If the inner subquery consists of two RR then
            a. Use the algorithm for one level queries with a
               RR assuming the joined network as a RN.
      B. else if (there is one ER) and (there is one RR) and
            (the ER has not attributes in the SELECT clause of
            the inner query) then
            a. Start generating responses from the ER.
```

b. SUBJECT NP: Use the algorithms for one level queries with an ER and the predicates which exist in the query from the ER.

c. OBJECT NP: The common entity-attribute of the ER and RR is joined with a labeled arc with another entity attribute of the RR. That entity-attribute is used to built the object NP. The algorithm for one level queries with a RR must be used.

d. <u>If</u> there are attributes of the RR of the query which have not participated in the generation process <u>then</u>

   i. generate responses traversing the network as it is specified by the algorithm in section 7.3.

C. <u>else</u> <u>if</u> (there is one ER) and (there is one RR) and (there are attributes of the ER in the SELECT clause of the inner subquery) <u>then</u>

   a. Start generating responses from the ER.

   b. SUBJECT NP: Use the algorithms for one level queries with an ER and the predicates with attributes from the ER.

   c. OBJECT NP: Use the algorithms for one level queries with an ER. The attributes of the ER in the SELECT clause must be used.

   d. For the attributes of the RR generate responses using the algorithms for one level queries with

```
            a RR.  Anaphoric references will be made to the
            common entity wherever it is needed.
2. else if (there is a RR with dependency link in the inner
   subquery) then
   A. Start generating responses from the RR with the
      dependency link (ER do not have dependency link).
      a. Sentences involving only that relation and its
         network will be generated. Attributes from other
         relations should not be used to generate NPs.
         The common entities in the joined network will be
         used to join the sentences making anaphoric
         references to them.
   B. If the relations in the inner subquery are RRs then
       a. Use the subnetwork corresponding to the other
          RR and continue to generate responses treating
          all the networks as one. Use the algorithm in
          section 7.3 to traverse it.
   C. else if (there is one RR) and (there is one ER) then
       a. For the ER:
          i. SUBJECT NP is generated from the common
             entity on which the two relation have been
             joined. Anaphoric reference are made to
             it.
          ii. OBJECT NP is generated using the predicates
              (the attributes and their values) which
              exist in the query from the ER. The
              algorithms for one level queries with an
```

ER must be used.

b. <u>Generation</u> <u>of</u> <u>the</u> <u>Response</u> <u>for</u> <u>the</u> <u>Outer</u> <u>Subquery</u>

1. Attributes in the SELECT clause of the outer subquery which have been used in the generation of the response for the inner subquery are not used in the generation of the response for the outer subquery.
2. <u>If</u> the relation in the outer subquery is an ER <u>then</u>
   - A. Generate one or more sentences using the algorithms for one level queries with an ER.
   - B. SUBJECT NP will be an anaphoric reference to the entity-attribute which is common to both subqueries (inner/outer). It is the attribute on which the two queries are joined. Actualy the attributes in the WHERE clause of this subquery are treated as predicates.
   - C. OBJECT NP: It will be built from the attributes in the SELECT clause as it is specified by the algorithm for one level queries with an ER.
3. <u>else</u> <u>if</u> ((the relation in the outer subquery is a RR with or without dependency link) and (there is a relation in the inner subquery with dependency link)) or ((the relation in the outer query is a RR and it does not have dependency link) and (there is not RR relation in the inner query with dependency link)) <u>then</u>
   - B. Consider the network corresponding to the outer

relation as a continuation of the network corresponding to the inner subquery;

C. The attributes which have been used in the generation process of the inner subquery should not be used again.

D. Traverse the required network generating sentences as specified by the algorithm in section 7.3.

4. <u>else</u> <u>if</u> (there is not RR in the inner subquery with dependency link) and (the RR in the outer subquery has dependency link) <u>then</u>

A. SUBJECT NP: The entity-attributes in the WHERE clause of the outer subquery are used to build NPs which will have the following syntax:

```
NP --> DET NOUN
DET --> This/these
NOUN --> noun corresponding to the entity.
```

A NP built in this way must be the SUBJECT of at least the first sentence.

B. OBJECT NP: It will be determined from the algorithm for one level queries with a RR.

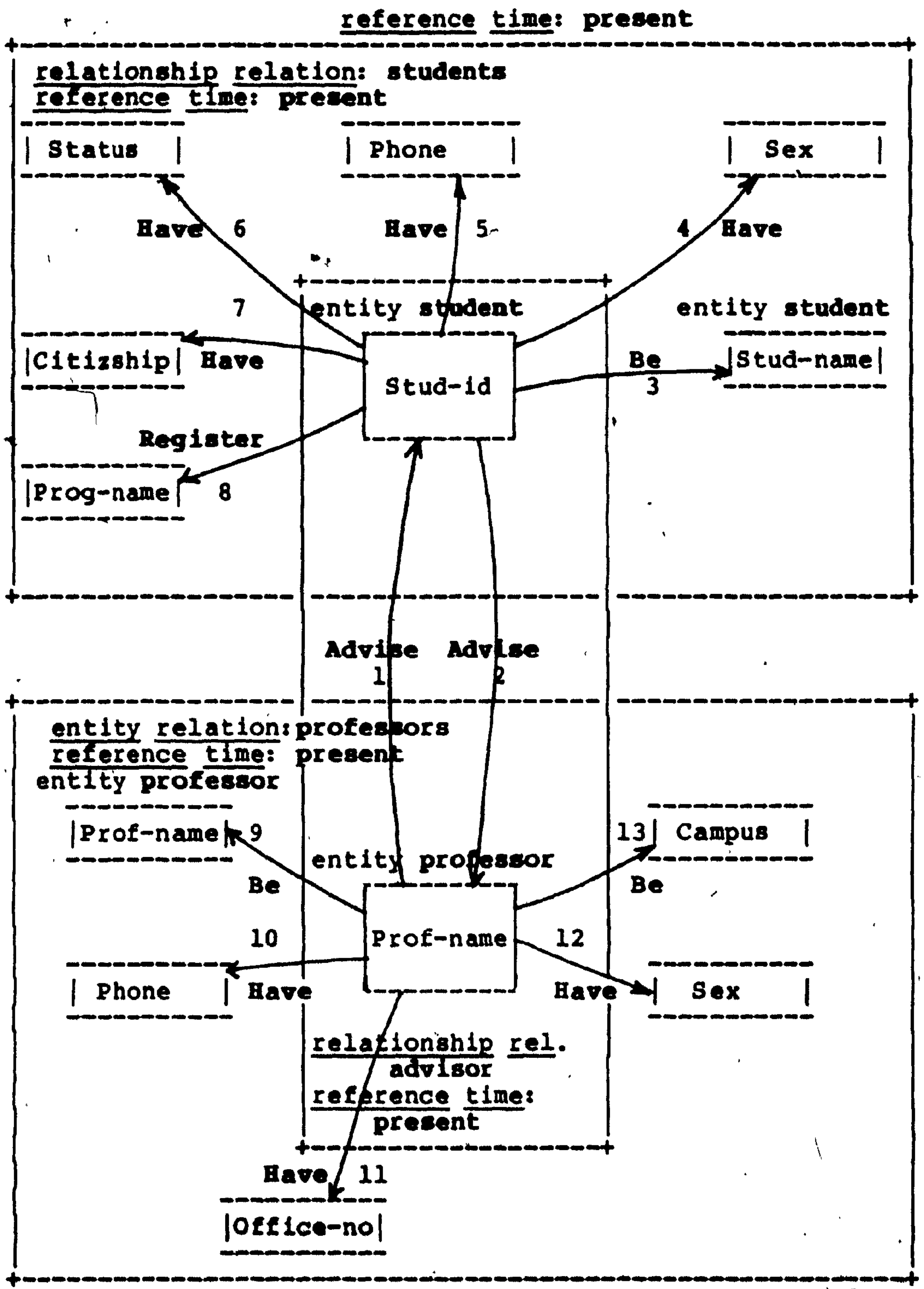


Fig. 8.4 The network created from the join of the PROFESSORS, STUDENTS and ADVISOR networks.

1.	$\left\lvert\begin{array}{l}\text{VC: +act}\\ \text{ASP:-perf;-progr}\end{array}\right\rvert$	2.	$\left\lvert\begin{array}{l}\text{VC: -act}\\ \text{ASP:-perf;-progr}\end{array}\right\rvert$
3.	$\left\lvert\begin{array}{l}\text{VC: +act}\\ \text{ASP:-perf;-progr}\end{array}\right\rvert$	4.	$\left\lvert\begin{array}{l}\text{VC: +act}\\ \text{ASP:-perf;-progr}\end{array}\right\rvert$
5.	$\left\lvert\begin{array}{l}\text{VC: +act}\\ \text{ASP:-perf;-progr}\end{array}\right\rvert$	6.	$\left\lvert\begin{array}{l}\text{VC: +act}\\ \text{ASP:-perf;-progr}\end{array}\right\rvert$
7.	$\left\lvert\begin{array}{l}\text{VC: +act}\\ \text{ASP:-perf;-progr}\end{array}\right\rvert$	8.	$\left\lvert\begin{array}{l}\text{VC: -act}\\ \text{ASP:-perf;-progr}\end{array}\right\rvert$
9.	$\left\lvert\begin{array}{l}\text{VC: +act}\\ \text{ASP:-perf;-progr}\end{array}\right\rvert$	10.	$\left\lvert\begin{array}{l}\text{VC: +act}\\ \text{ASP:-perf;-progr}\end{array}\right\rvert$
11.	$\left\lvert\begin{array}{l}\text{VC: +act}\\ \text{ASP:-perf;-progr}\end{array}\right\rvert$	12.	$\left\lvert\begin{array}{l}\text{VC: +act}\\ \text{ASP:-perf;-progr}\end{array}\right\rvert$
13.	$\left\lvert\begin{array}{l}\text{VC: +act}\\ \text{ASP:-perf;-progr}\end{array}\right\rvert$		

Fig. 8.5 Frames of the verbs of the RN in fig. 8.4.

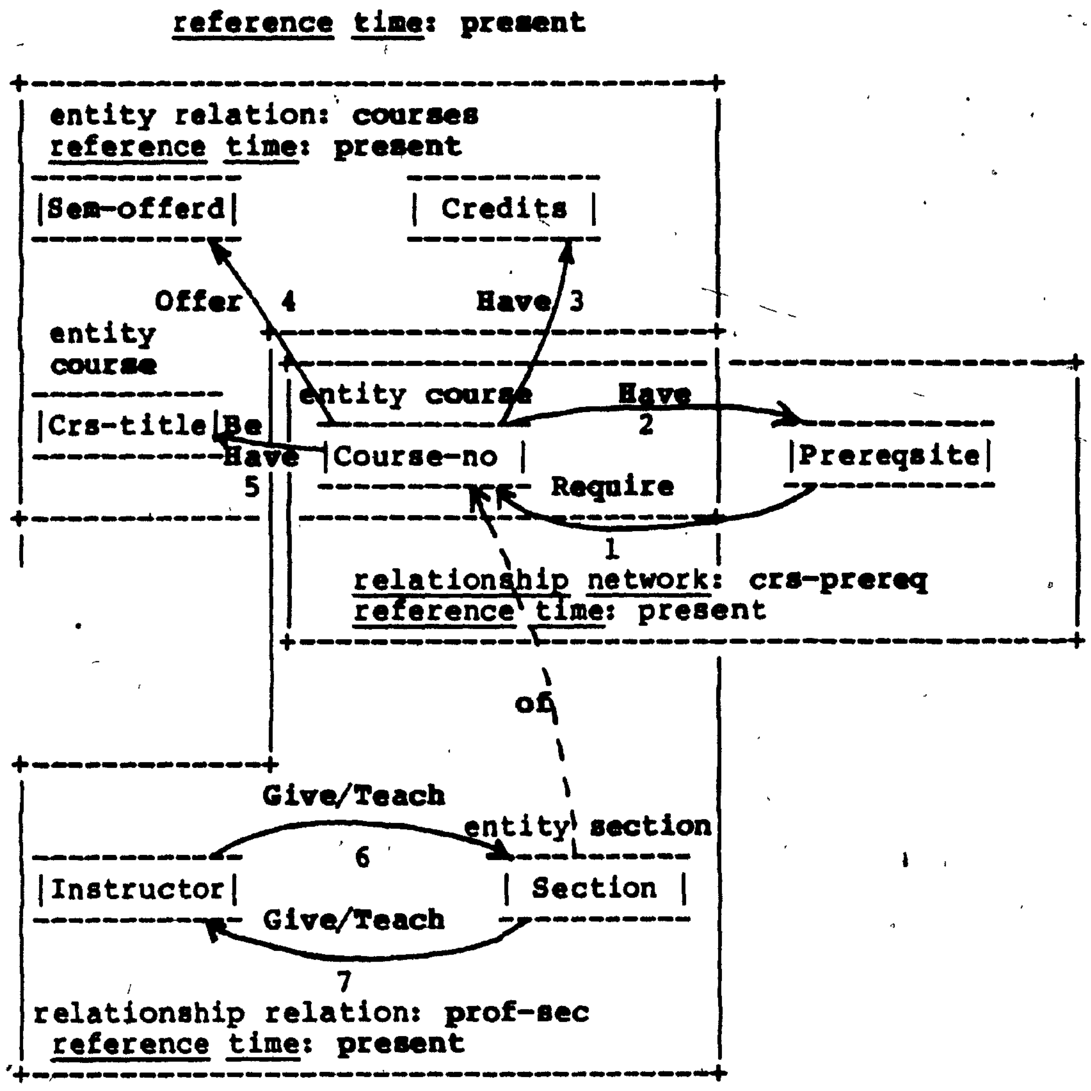


Fig. 8.6 Illustration of the join of the networks for PROF-SEC, COURSES and CRS-PREREQ.

```
1. |VC: -act          |     2. |VC: +act          |
   |ASP:-perf;-progr  |        |ASP:-perf;-progr  |

3. |VC: +act          |     4. |VC: -act          |
   |ASP:-perf;-progr  |        |ASP:-perf;-progr  |

5. |VC: +act          |     6. |VC: +act          |
   |ASP:-perf;-progr  |        |ASP:-perf;-progr  |

7. |VC: -act          |
   |ASP:-progr;-perf  |
```

Fig. 8.7 Frames of the verbs of the RN in fig. 8.6.

The following queries illustrate this algorithm.

Query no.1
```
   SELECT phone
   FROM   professors
   WHERE  prof-name IN
          SELECT prof-name
          FROM   advisor,students
          WHERE  students.stud-id = advisor.stud-id
          AND    students.stud-name = "G.Bellos"
```

RESPONCE: The student G.Bellos is adviced by a professor.

He has the phone number 77733.

Query no.2
```
   SELECT  prog-name, status
   FROM    students
   WHERE   stud-id IN
           SELECT stud-id
           FROM   advisor,professors
           WHERE  advisor.prof-name = professors.prof-name
           AND    phone = 68686
```

RESPONCE: The professor with the phone number 68686 advises some students. They are registered in the master and master program. They have full-time and full-time status.

<u>Query no.3</u>

```
SELECT sem-offerd,crs-title
FROM   courses
WHERE  course-no  IN
       SELECT course-no
       FROM   prof-sec,crs-prereq
       WHERE  prof-sec.course-no = crs-prereq.course-no
       AND    instructor = "J.Opatrny"
       AND    prereqsite = 652
```

RESPONCE: Sections of some courses is given by the instructor J.Opatrny. They have the prerequisite course 652. They are the courses compiler construction and language processing. They are offered in the Fall and Winter semesters.

<u>Query no.4</u>

```
SELECT instructor
FROM   prof-sec
WHERE  course-no  IN
       SELECT course-no
       FROM   courses,crs-prereq
       WHERE  courses.course-no = crs-prereq.course-no
       AND    sem-offerd = "F"
       AND    prereqsite = 622
```

RESPONCE: The courses which are offered in the Fall semester have some course numbers. The prerequisite course 622 is required by them. A section of these courses is given by the instructor D.Desai.

IX. CONCLUSIONS AND FUTURE WORK

The research in this thesis is a contribution in the area of natural language interface to data bases. Using tools such as a query language, a data dictionary, a lexicon, and a DBMS, a sentence generator is designed for communicating the database search to an end user. It is assumed that the input query is expressed in SEQUEL.

This research has shown me that the generation of meaningful text requires the existance of semantic structures which must represent the text to be generated. In this thesis the semantic structures designed to support the generator are the entity and the relationship networks. The generated response is a translation of the knowledge represented in such networks. The response to be generated depends on the input query. The query conveys in some form the question that the user has formed in his mind. We use the query input to traverse the networks. As the network is traversed sentences in English are generated.

The information extracted from the query is based on:

a) The structure of the SEQUEL query language. The place where each attribute appears in the query.
b) The fact that the design of our data base is based on the E-R model (we distinguish between entity relations and relationship relations).

In a SEQUEL query, information in the WHERE clause is what the user knows. Information in the SELECT clause is what the user would like to know. If the relation in the FROM clause is for example an entity relation then the discourse will be about the entity corresponding to this entity relation. In this way we confine the discourse to a specific subnet of the network corresponding to the query. For every query there is one network. The definition of the join operation (on networks) proposed in this thesis allows us to have for each query a network and to logically connect networks into more complex new networks.

Queries are divided into several types. For each type of query there is an algorithm which determines the response to be generated. The classification of queries is based on the structure of the query, namely the nested level of the query, the number and the type of the relations appearing in the query. Moving from simple queries such as "one level queries with one relation", to more complex queries we see that the algorithms for complex queries make calls to the algorithms of the simpler query types. This gives modularity and simplicity to the more complex algorithms.

The design of the data dictionary and the lexicon which are used as sourses of knowledge has greatly facilitated the lexical insertion task. The role of the data dictionary is different from the way it is generally used in data base systems. It contains among other things the mapping of the objects of the E-R model into groups of words of the English language. In other words the data dictionary contains some linguistic information.

Comparing our system with other systems we note the following:

a) The systems [30, 41, 47] can generate more types of sentences than our system. In the development of this system we did see the need for the types of sentences presented in section 5.3.

b) Our system generates meaningful sentences as responses to SEQUEL queries. Generators like [21] deal with random sentence generation.

c) Our system can generate multisentence text. Systems like [21, 38] can generate only single sentence responses.

d) Our system does not support dialogue with the user. A query in any query language such as ALFA, SEQUEL etc is assumed to be well defined. The user request in such a query language can not be vague nor be ambiguous. As a result of that, clarification dialogue is not considered in our system.

e) Our system uses a transformational grammar. Different grammars are used in other systems. The generators in [38, 47, 41] use ATN grammars and the one in [30] uses functional grammar.

f) Our system is not data base independent like the systems

[23, 24, 43]. It has been designed for a particular data base. The lexicon, the entity networks and the relationship networks must be changed in order to apply it for another data base.

A modification can be done into our system to improve the quality of the generated responces in the following case.

i) When the data base search retrieves more than one tuple with all the entries of an attribute identical then only one value for that attribute can appear in the generated response. The "V" part of the form of that attribute will be replaced with only one value.

The present version of our system can be expanded in many different ways:

1) Responses to queries with OR algebraic operators in the predicates has not been studied for certain types of queries. Only in the algorithm for one level queries with an entity relation OR operators have been considered in this version.

2) Dialogue can be used in case that the user wants to know something about the data base schema or the syntax of the acceptable queries. The present system can be extended to provide that information. Consider that the user does not know well about the attribute names of a relation or the syntax of a particular clause in the query. He can interrogate the system to provide the relevant

information. In this way he can minimize syntactic and semantic errors in his queries.

3) In our system response to a query is independent of the responses to previous queries. Responses related with the previous queries can be generated if the user is allowed to put in his queries relations which have been created during the processing of the previous queries.

4) A global expansion of our system is to make it a question-answering system. To do this the system must accept requests for retrieval from the database in a natural language. A module must be designed to accept requests in English, to analyse them, and to create a "SEQUEL type" internal representation. It can then be fed to our present system which will generate the sentential response. In such a case, the requests of the user may be vague and ambiguous, anaphoric references may be unclear, abbreviated words may be given etc. Then clarification dialogue would be required.

4. REFERENCES

1. Akmajian A. & Heny F., An Introduction to the Principles of Transformational Syntax, MIT press, 1975.

2. Astrahan M., System R: Relational Approach to Database Managment, ACM Trans. on Database Systems, vol.1, no.2, June 1976, pp.97-137.

3. Astrahan M. & Chamberlin D., Implémentation of a Structured English Query Language, Communications of the ACM, vol.18, no.10, Oct. 1975, pp.580-588.

4. Atre S., Data Base: Structure Techniques for Design, Performance and Managment, Wiley-Interscience, 1980, ch.3.

5. Barr A. & Feigenbaum E., The Handbook of Artificial Intelligence, vol.I, William Kaufmann, Inc. 1981.

6. Bates M., The Theory and Practice of Augmented Transition Networks, pp.191-260, New York: Springer, 1978.

7. Bates M. & Ingria R., Controlled Transformational Sentence Generation, Proc. of the 19th Annual Meeting of the Assoc. for Computational Lingustics, Stantford California June 1981.

8. Carbonell Jaime, AI in CAI : An Artificial Intelligence Approach to Computer - Assisted Instruction, IEEE Transactions on Man-Machine Systems, vol.MMS-11. no.4, Dec.1970, pp.190-202.

9. Carbonell Jaime & Collins Allan, Natural Semantics in Artificial Intelligence, Proceedings of the 3rd

International Joint Conference on Artificial Intelligence, 1973, pp.344-351.

10. Chamberlin D., SEQUEL 2: A Unified Approach to Data Definition, Manipulation and Control, IBM Journal of Research and Development, vol.20, no.6, Nov. 1976, pp.560-575.

11. Chamberlin D. & Boyce R., SEQUEL: A Structured English Query Language, Workshop on Data Description, Access and Control, ACM SIGMOD, May 1974, pp.249-264.

12. Chen P., The Entity-Relationship Model -- A Basis for the Enterprise View of Data, National Computer Conference, vol.46, 1977, pp.77-84.

13. Chen P., The Entity-Relationship Model -- Towards a Unified View of Data, ACM Transaction on Database Systems, vol.1, no.1, March 1976, pp.9-36.

14. Chen P., English Sentence Structure and Entity-Relationship Diagrams, Information Science, vol.29, 1983, pp.127-149.

15. Chomsky N., Aspects of the Theory of Syntax, Cambridge Mass. MIT Press, 1965.

16. Codd E.F., Seven Steps to RENDEZVOUS with the Casual User, in Data Base Managment edited by J.W. Klimbie & K.I. Koffeman, North-Holland, Amsterdam, 1974, pp.179-200.

17. Date D., An Introduction to Database Systems, Addison-Wesley 1981.

18. Fillmore C., The Case for Case, In Universal in

Linguistic Theory Edited by E.Bach and R.Harms, Holt, Rinehart and Winston Inc. 1968, pp.1-88.

19. Flavin M., Fundamental Concepts of Information Modeling, Yourdon Press New York, 1981.

20. Frank M., Modern English, a Practical Reference Guide, Prentice-Hall, 1972, pp.170.

21. Friedman J., Directed Random Generation of Sentences, Communications of the ACM, vol.12, no.1, Jan.1969.

22. Friedman J., A Computer Model of Transformational Grammar, American Elsevier Publishing Comp., New York 1971.

23. Harris L.R., User Oriented Data Base Query with the ROBOT Natural Language Query System, International Journal of Man-Machine Studies, vol.9, 1977, pp.697-713.

24. Hendrix G., Sacerdoti E., Sagalowicz D. & Slocum J., Developing a Natural Language Interface to Complex Data, ACM Transactions on Database Systems, vol.3, no.2, June 1978, pp.105-147.

25. Jacobs R. & Rosenbaum P., English Transformational Grammar, Baisdell Publishing Company 1968.

26. Jaworski W., Letourneau G., & Mack G., RISS/170 Data Base Managment System: User Guide, March 1981, Concordia University.

27. Kaplan S.J., Cooperative Responces from a Portable Natural Language Query System, Artificial Intelligence, vol.19, 1982, pp.165-187.

28. Lieberman D., Specification and Utilization of a

Transformational Grammar, IBM Thomas J. Watson Research Center, Project 4641, Scientific Report no.1, March 1966, Air Forch Cambridge Research Laboratories, Massachusetts.

29. Mann W., Text Generation, American Journal of Computational Linguistics, vol.8, no.2, April-June 1982, pp.62-69.

30. Mckeown K., The TEXT System for Natural Language Generation: An Overview, Proceedings of the 20th Annual Conference of the Association for Computational Linguistics, Toronto, Canada, June 1982, pp.113-120.

31. McLeod D. & King R., Applying a Semantic Database Model in Entity-Relationship Approach to Systems Analysis and Design, editor P.Chen, North-Holland, 1980, pp.193-210.

32. Meldeman M.J., RISS: A Relational Data Base Managment System for Minicomputer, Van Nostrand Reinhold, Toronto.

33. Mylopoulos J., Borgida A., Cohen P., Rousopoulos N., Jsotsos J. and Wong H., TORUS - A Natural Language Understanding System for Data Managment, Advance Papers on the 4th Int. Joint Conf. on Artificial Intelligence, Tbilisi, Georgia, USSR, Sept. 1975, pp.414-421.

34. Plath W.J., REQUEST: A Natural Language Question - Answering System, IBM Journal of Research and Development, July 1976, pp.326-335.

35. Radhakrishan T., De Mori R., Marakakis E. and Castillo R., Spoken Responces to Database Queries, Proceeding of

the International Conference on Systems Man and Cybernetics, vol.II, Dec. 1983, pp.825-829.

36. Rich E., Artificial Intelligence, McGraw-Hill, 1983.

37. Samlowski Wolfgang, Case Grammar, In Computational Semantics: An Introduction to Artificial Intelligence and Natural Language Comprehension Edited by Charniak E., & Wilks Y., North-Holland, 1976, pp.55-72.

38. Shapiro S., Generation as Parsing from a Network into a Linear String, American Journal of Computational Linguistics, 1975, Microfiche 33, pp.45-62.

39. Shapiro S, Generalized Augmented Transition Network Grammars for Generation from Semantic Networks, Proceedings of the 17th Annual Meeting of the Association for Computational Linguistics, August 11-12, 1979, California, pp.25-29.

40. Simmons R., Semantics Networks: Their Computation and Use for Understanding English Sentences, in Computer Models of Thought and Language Edited by Roger Schank & Kenneth Coldy, W.H.Freeman and Company, San Francisco, 1973 pp.63-113.

41. Simmons R., & Slocum J., Generating English Discourse from Semantic Networks, Communications of the ACM, Oct.1972, vol.15, no.10, pp.891-905.

42. Simmons R., Some Semantic Structures for Representing English Meaning, in Language Comprehension and the Acquisition of Knowledge, Edited by John Carrol & Roy Freedle, V.H.Winston Sons, 1972 Washington D.C.,

pp.71-79.

43. Templeton M. & Burger J., Problems in Natural Language Interface to DBMS with Examples from EUFID, Proceedings of the Conference on Applied Natural Language Processing, Febr. 1983, Santa-Monica California, pp.3-16.

44. Uhrowczik P., Data Dictionary/Directories, IBM Systems, Journal, vol.12, no.4, 1973.

45. Waltz D., An English Language Question Answering System for a Large Relational Database, Communications of the ACM, July 1978, vol.21, no.7, pp.526-539.

46. Winograd T., Language as a Cognitive Process, Volume I: Syntax, Addison Wesley, 1983.

47. Wong H., Generating English Sentences from Semantic Structures, Technical Report no.84, Aug.1975, University of Toronto.

48. Woods, W., Transition Network Grammars for Natural Language Analysis, Communications of the ACM, vol.13, no.10, Oct.1970, pp.591-606.