DEVELOPING DATABASE SEMANTICS AS A COMPUTATIONAL MODEL*

Kiyong Lee

Department of Linguistics,
Korea University,
Seoul 136-701, KOREA

Email : klee@mail.korea.ac.kr

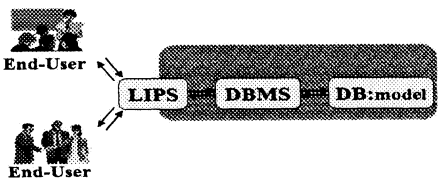
ABSTRACT

Both Hausser [1] and Lee [2][3] proposed Database Semantics as a computational model for natural language semantics that makes use of a database management system, DBMS. As an extension of these efforts, this paper aims at dealing with ways of representing linguistic descriptions in table forms because all the data in a relational model of DBMS is conceived of being stored in table forms. It is claimed here that, if an algorithm can be developed for converting linguistic representations like trees, logical formulas, and attribute-value matrices into table forms, many available tools for natural language processing can be efficiently utilized as part of interface or application programs for a relational database management system, RDBMS.

1. INTRODUCTION

For business transactions or academic administration, a commercial database management system (DBMS) like DB2, Oracle or Informix is widely used. Database Semantics is an attempt to adopt such a system for doing semantics for ordinary language. Since an interpretation model or background is necessary for processing linguistic information, Database Semantics can use as its model a database in a DBMS that provides both lexical meaning and world knowledge information. Furthermore, a DBMS constantly updates its database with new data, as fragments of a natural language like Korean or English are processed through a linguistic processing system LIPS. Hence, Database Semantics can consistently process even a larger fragment of discourse in natural language.

(1) A Model of Database Semantics



But, before developing Database Semantics as an application program into an RDBMS, the issue of representation must be resolved at least in the logical or conceptual level. Unlike ordinary semantics, Database Semantics as a computational model ultimately aims at a computational implementation. Hence, it should be able to build a successful interface between the two different levels of linguistic description and computational processing. As a result, representation comes to play an important role of bridging the

* This work was partially supported by the 1999 research grant from Korea Research Foundation. Here I would like to thank anonymous referees for the PACLIC14 and all my colleagues who helped me to complete this paper, especially, Suk-Jin Chang, Roland Hausser, Jae-Woong Choe, Masatoshi Kawamori and my graduate assistants Jungha Hong and Seungchul Choe.

and computational processing. As a result, representation comes to play an important role of bridging the gap between human and computer interactions. Linguistic descriptions are often represented in trees, logical forms, or attribute-value matrix (AVM) structures.[1] On the other hand, a computer system like RDBMS, which only knows a relational language like SQL, can conceptually recognize data stored in a table form only. This paper will thus focus on ways of representing linguistic information in a table form so that it can be recognized by an RDBMS.

The main task of this paper is then to discuss ways of converting linguistic representations like trees, logical forms, and matrices into table forms. If an algorithm is developed for these conversions, then many of the exiting parsers or interpreters can be directly incorporated into an RDBMS. Head-driven Phrase Structure, Lexical-Functional, and Left-Associative grammars, for instance, produce the results of their analysis represented in a sequence of Trie or AVM structures.

At the present, however, the presentation of such an algorithm is beyond the scope of this paper, for the version of Database Semantics proposed in the framework of RDBMS is still in the nascent state of being designed as a computational model. It is thus only hoped that fragmentary but concrete illustrations will be given to show how linguistic analyses can be represented in table forms.

2. WHY A RELATIONAL MODEL?

Hausser [1] was the first to develop Database Semantics. By adopting a network model for DBMSs, he proposed to construct a Word Bank, a lexical database consisting of word types and their tokens. By navigating this Word Bank, propositional content can then be processed linearly or left-associatively by a step-by-step manner. In a similar vein, Lee [2] also showed how a network model could be used to represent various structural relations in natural language analysis, while assuming that it could be a minimally sufficient model for processing natural language.

Lee [3], however, argued for some advantages of adopting a relational model possibly with object extensions. First, the relational model has given a conceptual basis for implementing most of the currently running commercial DBMSs like Oracle8*i* or Informix7.0.

Secondly, the relational model uses standardized SQL, a structured query language, for describing ways of building database structures and managing them, thus making it possible to apply it to the construction of a query system for natural language semantics.

Finally, its basic representational scheme at the logical or conceptual level is of a table form consisting of attributes and their values like an attribute-value matrix for linguistic representation. In the relational model, a database is a set of tables representing various relations among objects in a domain. This paper thus proposes to adopt a relational model RDBMS for developing Database Semantics.

3. WHY DATABASE SEMANTICS?

There are at least two reasons for adopting a DBMS as a basis for doing semantics. First, natural language generates a large amount of linguistic information with great complexity. Even a tiny fragment of text contains a lot of data. Especially when it contains indexical or context-dependent expressions, even a short sentence consisting of three or four words can theoretically create almost an infinite number of interpretations. Consider the following sentence:

(2) I am happy today.

---

[1] The original version of this paper discussed the tripartite representation of quantified sentences in AVM structures and converting these matrices into table forms. But, due to the editorial constraint on the limitation of space, the discussion of AVM structures has to be postponed for another occasion.

This string of words in English constitutes a well-formed sentence of English, meaning that, whoever the speaker is, she or he is happy on the day of her or his making the statement. Hence, it can have an indefinitely large number of interpretations depending on who the speaker is and also when or what day it is spoken.

Let's consider another example. Suppose two lovers say to each other:

(3) I love you.

This sentence consists of only three simple words. But, depending on the context of its use, it can produce infinitely many different interpretations. Here, neither *I* nor *you* refers to a fixed person. Even in a situation where two persons, say Mia and Bill, say to each other *I love you*, each of the pronouns has two different referents. The pronoun *I* once refers to Mia, while the pronoun *you* refers to Bill. And then the pronoun *I* refers to Bill, whereas the pronoun *you* refers to Mia. Hence, a model in which their referents are fixed cannot interpret sentences like (3).

By allowing a DBMS to keep updating its database, Database Semantics as a model-theoretic semantics should be able to use the constantly updated set of data supplied from the database to interpret contextually dependent statements or utterances. By appropriately partitioning the database into smaller parts through DBMS, Database Semantics can get the effect of subdividing its interpretation model into smaller sub-models so that the indexical expressions like *I* or *you* or even anaphoric pronouns like *she* or *he* may have varying referents within a larger model.

Event statements have the same problem of varying reference. Indexical expressions like *today* or *yesterday* have different referents as discourse situations vary. Then there are events that consist of smaller events, some of which may overlap each other. While statements are being made continucusly, their interpretation model should be able to keep track of their sequential as well as overlapping relations,. This task, Database Semantics is expected to perform efficiently by making a DBMS to control and manage its database.

While using a database as its interpretation model, Database Semantics installs in it various lexical and grammatical modules for processing natural language. The lexicon or word bank should be part of the database. This lexicon may be of various types. It can, for instance, be a multi-lingual dictionary or may contain a virtual dictionary temporarily built to deal with a given fragment of discourse.

The entire grammar consisting of both syntactic and semantic rules should also be installed in the same database. There may be sets of constraints or principles governing the generation of well-formed sentences. DBMS then manages and upgrades all these modules with new lexical items as well as newly required grammatical rules and constraints, as a larger linguistic corpus is introduced into the system. By allocating all these managing functions to DBMS, Database Semantics can make its module LIPS for linguistic information processing simply function as an application interface between its users and DBMS.

4. TABLES

A relational DBMS is normally identified with tables.[2] A table is a very simple and rigid form of representation, a triplet <Name, Attributes, Values> consisting of a name, a set of attributes or fields and their corresponding values. Nevertheless, it can carry a lot of information with varying types. The following is an example containing some personal information on professors:

[2] More detailed discussions of the notion of table and its properties in an RDBMS can be found any ordinary books on database management systems like [4], [5], [6], and [7].

(4) Table name: PHONE_BOOK

PROF_NO	NAME	INITIAL	OFFICE_PHONE
01	Lee	K.	3290-2171
02	Choe	J.	3290-2172
03	Kang	B.	3290-2173
04	Kang	M.	3290-2174

The name of this table is PHONE_BOOK. The table has four attributes: PROF_NO, NAME, INITIAL, and OFFICE_PHONE. Under each attribute, 4 different values are given with each of the four rows forming a record. The first row, for instance, gives information on Lee's office phone number.

Tables in an RDBMS are characterized by their data and structural independence. The above table (4), for instance, can easily be expanded by adding new data without destroying its basic structure.

(5) Table name: PHONE_BOOK(NEW)

PROF_NO	NAME	INITIAL	OFFICE_PHONE
01	Lee	K.	3290-2171
02	Choe	J.	3290-2172
03	Kang	B.	3290-2173
04	Kang	M.	3290-2174
05	Kim	S.	3290-2175
06	You	S.	3290-2176

This new table can be considered as being obtained by joining table (4) with table (6) containing information on Kim and You.

(6) Table name: PHONE_BOOK(PART)

PROF_NO	NAME	INITIAL	OFFICE_PHONE
05	Kim	S.	3290-2175
06	You	S.	3290-2176

Here, the attribute PROF_NO on each of the two tables (4), (5) and (6) uniquely identifies each row, thus playing a role as a primary key for linking one table to the other or joining them into a new expanded table.

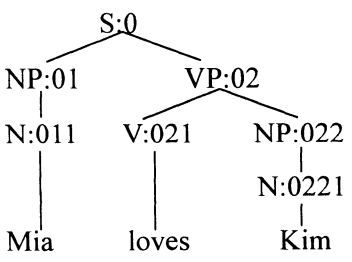
The notion of table is very simple as sketched here. It is purely a conceptual construct. This paper will, however, show the expressive power of tables for linguistic description. For this purpose, the paper aims at showing how seemingly complex-looking linguistic objects like Phrase Structure trees and logical formulas can be converted into table forms.[3]

5. FROM TREES TO TABLES

One possible way of representing the constituent structure of a sentence is to use a tree form. The following sentence (7), for example, may be analyzed into a phrase structure tree (8):

(7) Mia loves Kim.

(8)



[3] The conversion of AVM structures into table forms is briefly discussed in [3].

This tree is understood as representing how sentence (8) is formed: it consists of a noun *Mia*, a verb *loves*, and another noun *Kim*.

The same structural information can be represented in the following table form where the terminal nodes *Mia*, *loves*, and *Kim* are marked with a dummy symbol *t*:

(9) Table name: TREE8

NODE_CODE	MOTHER	DAUGHTER1	DAUGHTER2
0	S	NP	VP
01	NP	N	
02	VP	V	NP
011	N	t	
021	V	t	
022	NP	N	
0221	N	t	

Here, a unique code number is assigned to each node from which one can obtain information on the dominance and precedence relations among the nodes. The NODE_CODE 0, for instance, stands for the root node labeled S, and the NODE_CODEs 01 and 02 are its two daughters, NP and VP respectively. It is again understood here that 0 dominates 01 and 02 and that 01 precedes 02. This is so because the number of digits in each NODE_CODE is understood as representing its level in the tree and also the sequence of numbers except for the one in the last digit in each NODE_CODE as representing the code of its mother.

In Phrase Structural Grammar, the terminal nodes *Mia*, *loves*, and *Kim* in a tree like (8) are first marked with a dummy symbol like *t*, as in (9). It is then replaced by an appropriate lexical item selected from a lexicon by the process of lexical insertion. This fact can also be easily captured in a sequence of tables linked to each other. For this, we first construct a lexicon table like the following:

(10) Table name: LEXICON

LEX_CODE	CATEGORY	WORD_FORM
n_1	N	Kim
n_2	N	Mia
v_1	V	hates
v_2	V	loves

Secondly, the TREE table can be linked to the LEXICON table through the following bridging table:

(11) Table Name: BRIDGING_FOR_ANALYSIS

NODE_CODE	LEX_CODE
011	n_2
021	v_2
0221	n_1

This is a very rigid table, for it only allows the selection of a single word for each pre-terminal node. But it makes it possible to convey the exact amount of information as represented by tree (8).

For generation, we need a more flexible way of building tables and linking them to each other. From the pre-terminal tree table (9), it should be possible to construct a table frame like the following:

(12) Table name: PARSING8

S_CODE	N	V	N

Here, we first expand the bridging table (11) by adding a few more possible links to it.

(13) Table Name: BRIDGING_FOR_GENERATION

NODE_CODE	LEX_CODE
011	n_1
011	n_2
021	v_1
021	v_2
0221	n_1
0221	n_2

On the basis of this bridging table, we can now complete the table frame (12) and obtain the following:

(14) Table name: GENERATION8

S_CODE	N	V	N
1	Kim	hates	Mia
2	Mia	hates	Mia
3	Kim	loves	Kim
4	Mia	loves	Kim
5	Kim	hates	Mia
6	Mia	hates	Mia
7	Kim	loves	Kim
8	Mia	loves	Kim

This table gives a complete list of well-formed sentences that can be generated on the basis of the pre-terminal tree table (9) and the bridging table for possible generation (13). The construction of this bridging table (13), of course, depends on various co-occurrence restrictions.

It has been shown here how phrase structure information can be represented in a table form and also how the processes of language analysis and generation can be captured by a sequence of tables with appropriately constructed bridging tables. It should, however, be noted that the use of tables for representing constituent structures should not be understood as claiming the adequacy of Phrase Structure Grammar for linguistic description. It rather shows the possibility of accommodating a grammar system that generates trees into an RDBMS as a genuine part of the LIPS of Database Semantics.

6. FROM LOGICAL FORMULAS TO TABLES

In formal semantics for natural language, sentences are first disambiguated through syntactic analysis and then translated into a logical language like Intensional Logic for their model-theoretic interpretations. In this section, I will show how logical formulas can be converted to tables, which represent the propositional content of sentences in natural language.

6.1 SIMPLE SENTENCES

Simple sentences are normally translated into atomic formulas in a Predicate Logic-like language, which consists of a relation and a set of its arguments. Suppose we have a simple sentence:

(15) Mia loves Kim.

This sentence can be translated into an atomic formula like the following:

(16) **love'(mia', kim')**

This formula consists of a relation **love'** and two arguments **mia'** and **kim'**.

This formula can easily be turned into a table consisting of four attributes, PROP_CODE, RELATION, ARG1, and ARG2.

(17) Table name: RELATIONS_1

PROP_CODE	RELATION	ARG1	ARG2
1	love	mia	kim

The first row in this table is marked with PROP_CODE 1, containing information on the loving relation between Mia and Kim.

To this table, other relations may be added.

(18) Table name: RELATIONS_2

PROP_CODE	RELATION	ARG1	ARG2
1	love	mia	kim
2	love	kim	mia
3	sisters	mia	kim

Two more records or pieces of information are added to the table without changing its basic structural frame.

By applying some inference engine to the above table, Database Semantics must be able to infer the propositions expressed by the following two related sentences:

(19) a. Mia and Kim love each other.
b. They are sisters.

Table (18) is thus understood as representing the propositional content of both of the sentences in (19) as well.

6.2 EMBEDDED SENTENCES

Sentence (20) contains an embedded sentence.

(20) Mia knows that Kim loves her.

Here, the embedded sentence *Kim loves her* plays its role as the Object of the verb *knows*.

This sentence may have a logical representation like the following:

(21) **knows'(mia', loves'(kim', mia))**

The predicate **knows'** takes two arguments: one is an individual constant **mia'** and the other a proposition **loves'(kim', mia')**.

By adopting Lee's equation-solving approach [8], this formula may be decomposed into two atomic formulas as in:

(22) **knows'(mia', p)**
p = (loves'(kim', mia'))

The letter **p** is used here as a variable of the propositional type. It is equated in (22) to the proposition **loves'(kim', mia').**

A more accurate way to represent the use of the pronoun *her* in sentence (20) is to represent it as a variable because it may not necessarily refer to Mia. Only if some background is provided appropriately, it can be equated to **mia'** and then refer to Mia. Otherwise, it is left unbound as a free variable.

By accepting such a treatment of pronouns as variables, translation (21) can be reformulated as in (23):

(23) **knows'(mia', p)**
p = (loves'(kim', x')
x = mia'

This translation consists of a set of an atomic formula and its related equations. One equation refers to the content of the relation **knows'** and the other binds the variable **x** to **mia'**.

These formulas can also be converted to a table like the following:

(24) Table name: PROP_(23)

PROP_CODE	RELATION	ARG1	ARG2
prop_1	knows'	mia'	p
prop_2	=	p	prop_3
prop_3	loves'	kim'	x
prop_4	=	x	mia'

Here the equation **x=mia'** is treated as part of the propositional content while its PROP_CODE is assigned prop_4 in the table PROP_(23). But the information that is conveyed by this equation is not part of the propositional content of sentence (20). It is rather part of the background for interpreting it. Hence, it should be taken away from the table PROP_(23) and placed in a separate table for storing background information.

(25) Table name: BACKGROUND

BG_CODE	RELATION	ARG1	ARG2
bg_1	=	x	mia'

As is revised as in (26), the PROP table can link to the BACKGROUND table.

(26) Table name: PROP_(23)

PROP_CODE	RELATION	ARG1	ARG2	BG_CODE
prop_1	knows'	mia'	p	
prop_2	=	p	prop_3	
prop_3	loves'	kim'	x	bg_1

Table (26) is now linked to table (25) by relating the PROP_CODE prop_3 to the BG_CODE bg_1.

The table approach can distribute information into appropriately classified tables. But it can also link them systematically, as the need arises. Consider another type of embedded sentences:

(27) Mia promised Kim to help her.

The logical form of this sentence may look like the following:

(28) **PAST(promise'(mia', kim', help'(mia',kim')))**

Through Lee's equation-solving approach [8], the above representation can be decomposed into the following set of equations with a propositonal formula.

(29) **PAST(promise'(mia', kim', p))**
p = help'(x, y)
x = mia', y = kim'

The equation **x=mia'** is supported by the so-called EQI-NP constraint or some control mechanism associated with the verb *promise* in English, while the equation **y=kim'** is introduced by some background information that governs the use of pronouns.

In order to represent these pieces of information in a table form, we construct a PROP table first.

(30) Table name: PROP_(29)

PROP_CODE	TENSE	RELATION	ARG1	ARG2	ARG3	BG_CODE	CL_CODE
Prop_1	past	promise	mia	kim	prop2		
Prop_2		help	x	y		bg_10	subject

We then need two tables, a table for BACKGROUND and a table for CONTROL.

(31) Table name: BACKGROUND

BG_CODE	RELATION	ARG1	ARG2
Bg_10	=	y	kim'

(32) Table name: CONTROL

CL_CODE	RELATION	ARG1	ARG2
Subject	=	x	mia'

Table (30) is linked to tables (31) and (32) through the keys supplied by PROP_CODE, BG_CODE, and CL_CODE in these tables.

6.3 EVENT-RELATED STATEMENTS

In natural language, each verb must carry information on tense. In a coordinate or complex sentence, some verbs may lack any overt tense marking, as with the verb *marry* in the following sentence:

(33) Bill promised Mia to marry her.

Here the event of marrying is understood to take place not before, but after the promise was made. This interpretation is possible because any promised event must occur later than the time of making the promise itself. Consider the following pair of sentences:

(34) a. Bill promised that he married her.
b. Bill promised that he would marry her.

Both sentences are syntactically acceptable. Unlike (34b), however, sentence (34a) cannot be properly interpreted semantically.

By borrowing the representation scheme of neo-Davidsonian Event Semantics, the content of sentence (33) and its background may be captured as in:

(35) **promise'(e_1, bill', mia', p)**
p = marry'(e_2, x, y)
PAST({e_1})
precedes'(e_1,e_2)
x = bill', y = mia'

Here are two events, e_1 and e_2: one is an event of Bill's making a promise to Mia, and another, an event of marrying. The event e_1 is a past event. Hence, it is represented as part of the interval **PAST**. The predicate **promise'** relates two events or actions by arranging the temporal sequence of their occurrences. The formula **precedes'(e_1,e_2)** states that the event of promising occurs before the event of the promised event or action.

Two equations for checking coreference are again introduced here. Since the verb promise is a subject-control verb, the invisible subject of the infinitival clause is understood to be the same as the main subject. The equation **x = bill'** expresses this fact. The second equation **y = mia'** again depends on presumable background information, which thus makes it possible to interpret the pronoun *her* as referring to Mia.

The semantic information represented by formulas in (35) will be stored separately in four different tables for PROP, TEMP_REL, CONTROL, and BACKGROUND information. The PROP table will contain the following set of data:

(36) Table name: PROP_(35)

P_CODE	REL	EVENT	ARG1	ARG2	ARG3	BG_CODE	CL_CODE	T_CODE
prop_1	PAST	e1						
prop_2	promise'	e1	bill'	mia'	prop_3			temp_2
prop_3	marry'	e2	x	y		bg_9	equi-np	

The PROP table contains three propositions, prop_1, prop_2, and prop_3. The first row prop_1 simply states that the main event of promising occurred in the PAST. The second row represents the main content of sentence (33), but a part of it refers to the third row prop_3. Furthermore, it contains information on T_CODE. Its value temp_2 then relates the PROP table to the TEMP_REL table that tells how the two events are temporally related to each other.

(37) Table name: TEMP_REL

T_CODE	REL	ARG1	ARG2
temp_1	overlaps	e1	e2
temp_2	precedes	e1	e2
temp_3	follows	e2	e1

The value temp_2 of T_CODE shows that e1 precedes e2.

The third row prop_3 in the PROP table requires an expanded table for BACKGROUND.

(38) Table name: BACKGROUND(expanded)

BG_CODE	RELATION	ARG1	ARG2
bg_9	=	y	mia'
bg_10	=	y	kim'

On the basis of this BACKGROUND table, the variable **y** marked with the value bg_9 of BG_CODE is equated to **mia'.**

The third row also contains information on CL_CODE. This again tells the invisible subject of the infinitival complement of the verb *promise* is controlled to be coreferential with the main subject *Bill.*

In this section, it has been shown how the propositional content of simple and complex sentences and their related background or control information can be represented in table forms and also how these tables can be linked to one another to share some information. As should be noted again, these representations are only conceptual constructs. Hence, further refinements may become necessary at the physical level of implementation.

7. CONCLUDING REMARKS

A version of Database Semantics proposed here is based on an RDBMS. It is first characterized by the representation of data in table forms and secondly by a relational query language SQL. In this paper, nothing is discussed on the use of SQL for developing a query system of NL understanding or generation. Instead, efforts were made to discuss the use of tables for linguistic description.

The use of tables in linguistics is not totally unknown. Ordinary grammar books and dictionaries contain tables of verbal inflections and nominal declensions. Allomorphic variations and irregular and exceptional word forms are often listed in table forms. In computational work, too, we find a technique like Chart Parsing that uses tables. The use of tables in Database Semantics may be understood as a formal way of extending such traditional use to NL semantics.

The use of tables in Database Semantics is simply a consequence of adopting a relational DBMS. In an RDBMS, data is conceptually viewed as being stored in table forms. Although they are pure conceptual constructs, tables are meta-theoretic objects to which mathematically explicit operations can apply. Hence, to build a theoretically adequate version of Computational Semantics, a design mechanism based on such constructs may be of great use.

The use of tables in currently running commercial RDBMSs is very much restricted. It can, however, gain a more expressive power for linguistic description through object extensions and further implementation. Different types of data or records are exchanged or augmented through primary or foreign keys that are specified in tables and also as part of linking rules. If such a manner of linking from row to row can be extended to a new manner of linking from table to table, then it must be a greatly desirable improvement. For linguistic analysis often requires globally linked blocks of information.

8. REFERENCES

[1] Roland Hausser. Foundations of Computational Linguistics: Man-Machine Communication in Natural Language, Berlin: Springer-Verlag, 1999.

[2] Kiyong Lee. "Computational Linguistics" [written in Korean], in Beom-mo Kang et al., Formal Semantics and Description of Korean [written in Korean], pp. 498-551, Seoul: Hanshin, 1999.

[3] Kiyong Lee. "Language and Representation: A Basis of Database Semantics" [written in Korean], Proceedings of the Eleventh Conference on Hangul and the Korean Language Information Processing, pp. 297-303, 1999.

[4] C.J. Date and Hugh Darwen. Foundation for Object/Relational Databases: The Third Manifesto, Reading, MA: Addison-Wesley, 1998.

[5] Ramez Elmasri and Shamkant B. Navathe. Fundamentals of Database Systems, 2nd edition, Reading, MA: Addison-Wesley, 1994.

[6] Raghu Ramakrishnan, Database Management Systems, Boston: McGraw-Hill, 1998.

[7] Peter Rob, and Carlos Coronel. Database Systems: Design, Implementation, and Management, New York: Course Technology, a Division of International Thomason Publishing, 1997.

[8] Kiyong Lee. "Computing Information by Equation Solving", in Chungmin Lee and Beom-mo Kang (eds.), Language, Information, and Computation, pp. 14-26, Seoul: Taehaksa, 1993.