ELISE: Office Procedures Automation Tool By State-Transition Model

HIROSHI TSUJI* and FUMIHIKO MORI*

ELISE (Electronic Intelligent Secretary System) is an experimental system to automate office procedures. The basic premise of the system is that office procedures can be expressed as a set of state- transition models, which specifies how an object is to be processed at an event's occurrence, the choice among alternative processings being dependent upon the state of the object. Office procedures are stored in a relational data base as four types of relations. The presented relational structure for storing the office procedure facilitates the consistency check of the office procedures as well as its retrieval, addition, editing, and deletion. ELISE consists of six components: event monitor, procedure monitor, dispatcher, state manager, observer, and procedure manager. These components cooperate with each other to obtain, store, and execute procedures for automatic execution of office work. The presented architecture allows the system and the user to cooperate for carrying the object from its initial state to the final state. This paper describes the representation of office procedures, and the role and function of each component, together with an example of application.

1. Introduction

Office automation has a long history, longer than that of business application of electronic computers. When payroll processing was computerized, it was called office automation. There are many more examples of office work which have already been computerized. In fact, when any office work is sufficiently structured, routine, and therefore programmable, and if it is cost effective to implement it in a computer, chances are that the office work has already been computerized. Therefore, the focus of the buzz word office automation today is the possibility of computerization of a myriad of small and semi-structured office tasks, the computerization (by programming) of which has not been cost effective until now.

Most office work can be viewed as processes to transfer objects (such as forms and documents) from their initial states (such as the state of being a blank form) to their final states (completed and approved) [6]. This observation leads to the premise that most of the potentially automatable office work can be represented in the automaton state-transition model.

ELISE (Electronic Intelligent Secretary System) is a system for automatic execution of the small and semi-structured office tasks represented in the automaton model. The objective in the design and development of ELISE is two-fold: representation of automatable office work, and functional division of software architecture of a system for automatic execution of the office work.

Section 2 is devoted to the discussion of the survey of the previous researches and scope of this research. In section 3, it is shown that office procedures can be represented by four types of relation (procedure relation, event relation, activity relation, and state relation). Section 4 treats the role and function of the six components of ELISE (event monitor, procedure monitor, dispatcher, state manager, observer, and procedure manager), as well as interface among them. An application to secretarial work is also discussed in section 5.

2. Automation of Office Procedures

In spite of developments in computer and related technologies, office worker's productivity has shown very little increase, as compared to the productivity improvements in the manufacturing sector. Office automation is a collective term pertaining to all endeavors to improve office workers' productivity and the quality of their working environment.

Various software for office automation have been developed recently, including word processors for documents creating and editing, spread sheet based table processors with recalculation capability, and record processors for retrieval and manipulation of data stored in a database. These software systems are effective for improvement of office productivity.

However, their functions are to mechanize some works done in the office, but not to automate office procedures. The knowledge necessary to use these software, such as 1) when and under what condition these software should be used, and 2) which software should be used in what sequence, is left to the discretion of office workers. This kind of control information is either stored in a person's memory, or written in the

*Systems Development Laboratory, Hitachi, Ltd., 1099 Ohzenji Asao, Kawasaki, 215 Japan

manuals, not in a machine-executable format.

This fact was pointed out by Zisman [11]. He predicted that office automation would advance from task mechanization to procedure automation. Figure 1 shows the difference between task mechanization and procedure automation. The procedure automation has received much research effort. Work in this field can be classified into two categories as shown in Figure 2:

a) Office modelling methodologies.

These are researches on office description, modelling, and analysis methodologies. Ellis, et al [2] proposed an office model called ICN (Information Control Net), while Hammer, et al [3] proposed a language called OSL (Office Specification Language). Although based on a rigorous mathematical model, ICN gives a graphical representation of office procedures, which facilitates understanding of the sequential control structure underlying office procedures. Based on the functional view of the office, OSL gives a state-transition description of focal objects, which can be an abstract or concrete object processed in the office.

b) Systems for automation of office procedures.

These are researches on event-based systems for automatic execution of office procedures, including OBE (Office procedure By Example) by Zloof [12], FORMAL (Form ORiented MAnipulation Language) by Shu [6], OFS (Office Form System) by Tsichritzis [8], and SCOOP (System for COmputerization of Office Procedures) [10] by Zisman. These systems mostly treat data processing-based procedure automation.

This paper concerns the latter category.

OBE [12] is an extention of QBE (Query By Example), a query language for relational databases. It is easy for non-programming user to use OBE because OBE unifies OA software, such as data processing, word processing and electronic mail, and a trigger program. OBE implicitly assumes that, when a trigger event occurs, the object to be processed exists in a certain predetermined state. However, an object naturally exists in various states: a form may be approved or not yet approved, or approved with a certain condition, or returned from a manager to his/her secretary with a memo attached to it. Therefore, the function for automating office procedures in OBE is limited.

FORMAL [6] and OFS [8] are form-based systems. Forms are considered to be the focal object in the office and the most suitable object to be used as an interface between the system and the user [5]. Since most of the business applications are form-processing, these systems concentrate on automation of the routine parts of form-processing. Much attention is paid to the structural representation of various forms. However, office workers use various OA software as well as form-processing.

SCOOP [10] is based on APN (Augmented Petri Net) model representation of the office procedures. In APN model, the process is represented by Petri net, and the knowledge concerning the control is represented by the production rules. This representation is powerful enough to represent the state-transition model. However, it is not so easy for a non-programming user to specify the office procedures and to verify them for SCOOP.

The target system ELISE considered here should have the following characteristics:

1) The class of automatable office work is broader than that of OBE,

2) Office procedure includes wide class of tasks executed by OA software,

3) Non-programming user can specify and verify the automatable office procedures.

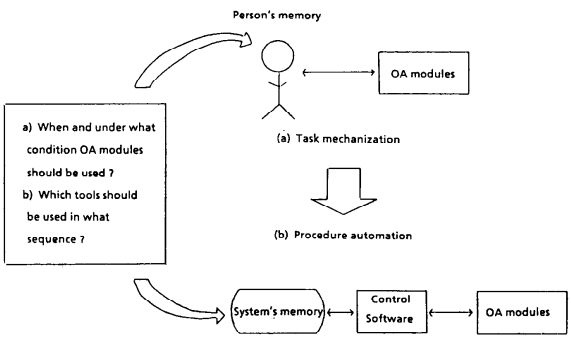

Fig. 1 From Task Mechanization to Procedure Automation.

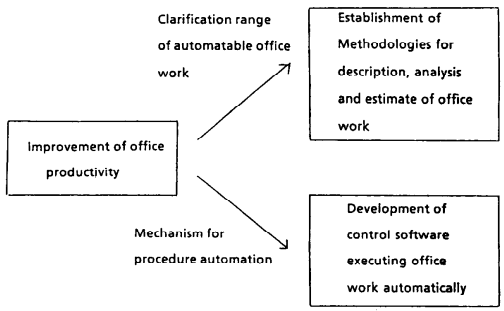

Fig. 2 Theme of Procedure Automation.

3. Office Procedures Representation

3.1 Model for Automatable Office Procedures

The office worker must teach ELISE how to do the automatable part of the office work. This will be done by a non-programmer. Accordingly, ways other than the use of conventional programming languages must be employed. Thus, simple representation of the knowledge concerning how to perform the automatable part of office procedures is required.

An office procedure can be viewed as a process of transferring an object from an initial state to a final state, with a number of intermediate states in between. In the example of travel expense account processing, an

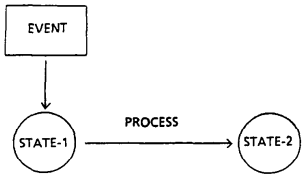

Fig. 3 State-Transition Model.

expense form is transferred from the initial (blank) state to the final (paid) state, with the filled, approved, checked and computed states in between. Therefore, the state-transition representation should be the basis of office procedure automation.

Office procedure has been defined the following automaton format such as Figure 3:

IF event When state-1 THEN process GOTO state-2.

This means that if "event" occurs, perform "process" on objects in "state-1" and update the state of the object to "state-2".

The following definitions are in order here:

Definition 1: Object

An object is a "thing" to be processed during the execution of office work. When an object is processed, its state is updated. Forms, sheets, and files are objects, among others.

Definition 2: Event

An event is an occurrence of a trigger to activate a process. Arrival of a specific time or date, reception of mail, command input by the user, a counter counted up to a specified number, and a change in the state of certain other object are examples of events.

Definition 3: State

A state is a label attached to an object to identify the current status of the object. At each state, except for the final state, an object waits for occurrence of one or more events. When an event occurs, the object receives the specified processing.

Definition 4: Process

A process is a manipulation of an object. A process consists of a series of activities. An activity corresponds to a command to activate a processing module, including database retrieval, table processing, word processing, and electronic mailing.

3.2 Storing Office Procedures

The state-transition representation of office work procedures are stored in a relational database. There are three reasons why a relational database is used:

(1) The state-transition model is represented by a relation P(IF, WHEN, THEN, GOTO) straightforward,

(2) It is easier for an office worker to define relations than to write program,

(3) The relational algebra described later is useful for procedure validation.

The relation "P" is called procedure relation. "IF" is an event code, "THEN" is a process code. "WHEN" and "GOTO" are states.

Consider the "IF" part of the procedure relation. Since an occurrence of an event is always instantaneous, simultaneous occurrences of events need not be considered here. For instance, a complex event such as receiving mail at a specified data can be divided into two distinct events as arrival of the time and reception of the mail. A suitable state representation can be set up so that if either state occurs first the object in question is transferred to a new state waiting for the other state. As shown in Table 1, there are several types of events.

Event occurrence can be detected by checking the relation between the event instance and prespecified constants. For instance, an event identifier "e1" means that the "date" is "later than or equal to" "February 21". Therefore, the "IF" part in the relation "P" can be defined as relation E(Eid, Type, R, C), where "Eid" is an event code and its domain is the same as that of the "IF" part in the relation "P". "Type" is an event type, "R" is a logical operator, and "C" is a constant. Logical operator "R" is one of the following: Equal to (=), greater than (>), greater than or equal to (≧), less than (<), less than or equal to (≦), or not equal to (≠). Relation "E" is called event relation. When an event type "type1" occurs with instance "c1", it can be determined using this relation if an event corresponding to the "IF" part of the procedure relation has occurred or not. The following retrieval condition is used:

Type="type1" AND (R="=" AND C="c1"
or R=">" AND C>"c1"
or R="≧" AND C≧"c1"
or R="<" AND C<"c1"
or R="≦" AND C≦"c1"
or R="≠" AND C≠"c1").

Consider the "THEN" part of the procedure relation. Since any processing which does not contain branching by decision can be expressed as a combination of serial and parallel processing (Fig. 4), relation A(Pid, Pp, Ps, Aid) is defined to represent the process. "Pid" is the process code whose domain is the same as that of the "THEN" part in the relation "P". "Pp" is the

Table 1 Types of Events.

Type	Explanation
Command Event	A command is used by the user
Receive Event	An electronic mail is received
Time Event	A specified date and/or time has arrived
State Event	A specified object is transferred to a specified state

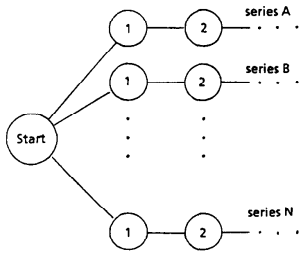

Fig. 4 Parrallel and Serial Processing of Office Work.

identification code for parallel processing. "Ps" is the identification code for serial processing. "Aid" is an activity which corresponds to a command for suitable software module. This relation is called activity relation.

Lastly, consider the state management for the objects. Each object has a state attached to it. An object is transferred from one state to another, depending upon the occurrence of events and execution of processing corresponding to the object, state, and event. Relation S(Oid, Sid) is used to represent the state of the objects where "Oid" is the identification code for object and "Sid" is the state. This relation is called state relation.

4 Architectural Analysis

4.1 System Architecture

The problem considered here is the analysis of functional components of ELISE. ELISE is an office procedure automation system, based on the state-transition representation of office work.

In a series of office activities, which carries an object from its initial state to the final state, some parts can be automated, and other parts must be done by office workers. The latter part includes complex decisions and exception handling. Automation of semi-structured office work should be carried out through cooperation between office workers and the system. The system assists the office worker by carrying out the automatable part of the work. Figure 5 illustrates general structure of ELISE and office automation modules (software such as word processing, electronic mailing).

Roughly speaking, ELISE should consist of two parts: Office procedure acquisition part and execution part. There are two cases of office procedure acquisition.

(a) The user teaches the office procedure explicitly.

(b) The system observes user's activity, and acquires office procedure automatically.

The component which learns the office procedure explicitly specified by the user is called procedure manager. Another component, called observer, is responsible for the second case.

Since the office procedure is specified in the production rule format representing the state-transition upon

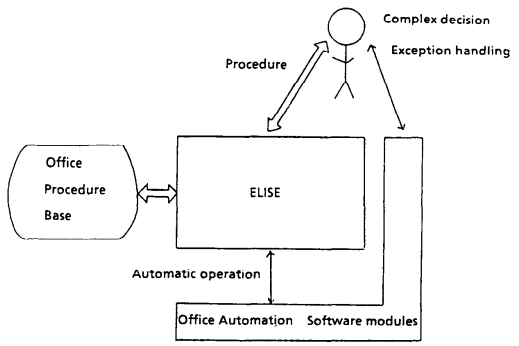

Fig. 5 ELISE, Office Procedure Base and OA software modules.

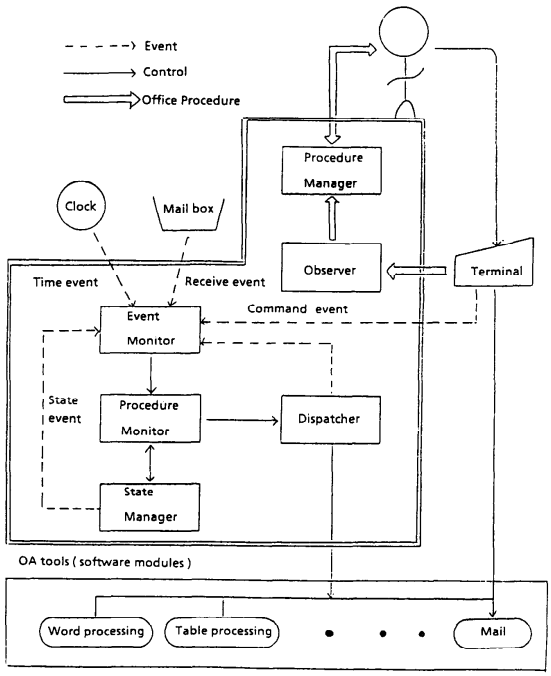

Fig. 6 Architecture of ELISE.

occurrence of an event, the execution part consists of the following components: The part which monitors the occurrences of events, called event monitor, the part which executes the process, called the dispathcer, the part responsible for the procedure retrieval, called procedure monitor, and the part responsible for the management of the states of the objects, called the state manager.

4.2 Role and Function of the Components

Component structure of ELISE is shown in Figure 6. The role and function of each component is described below:

(1) Event monitor

The event monitor observes occurrences of events such as arrival of a specific time or date, reception of

mail, or command input by the user. When an event occurs, the event monitor determines whether any of the cataloged office procedures should be activated. If one or more events which trigger office procedures are found, the event monitor passes the information to the procedure monitor.

(2) Procedure monitor

When an event is detected, this component searches for procedures which should be activated by the event. The procedure monitor also searches for objects which are in a state waiting for processing with state manager. If one or more objects are found to be processed, the procedure monitor activates the dispatcher to perform the activities defined in the procedure.

(3) Dispatcher

The dispatcher activates software modules to perform the processing found to be executed by the procedure monitor. A process usually consists of several activities. The dispatcher is responsible for the orderly execution of a series of activities.

(4) State manager

This component is responsible for the management of the states of objects. When an event has occurred, and the corresponding process have been performed on an object, the state manager updates the object's state to a new one defined in the corresponding procedure.

(5) Procedure manager

This component accepts office procedures explicitly given by a user, and stores them in the office procedure base. When requested, the procedure manager displays procedures on the screen for editing purpose. If either any error or inconsistency is detected in the newly specified procedures such as using undefined objects or activities, the procedure manager engages in conversation with the user to clarify the problem.

(6) Observer

This component observes the activities of the user, such as the history of the use of commands by the user, and develops an office procedure to repeat the activities.

4.3 System Operation

As seen before, four types of relations are designed to represent the office procedures. These relations are stored in the office procedure base. The system, ELISE, refers to these four types of relations and performs a part of the office work automatically.

The event monitor continuously observes event occurrence, and if an event occurs, this monitor obtains the instance and determines if the instance is a trigger for cataloged office work by referring to the event relation. The retrieval condition is the one given in section 3.2.

When the event monitor detects an event to trigger an office procedure, the procedure monitor uses the procedure relation and the state relation to determine whether any object is waiting for processing at this event. To do this, the procedure monitor retrieves the relation obtained by the JOIN operation [1] of the procedure relation and state relation with kevs "WHEN" and "Sid" respectively. There can be a case where no record is obtained by this retrieval, since there may be no object waiting to be processed corresponding to the event detected by the event monitor. If a record is retrieved, the process code of the "THEN" part and the object to be processed in the "Oid" part are passed to the dispatcher and the state specified in the "GOTO" part is passed to the state manager.

The dispatcher retrieves activities from the activity relation with the received process code as the key. The dispatcher divides the retrieval results using the "Pp" which is the branching identifier for the parallel processing. The results of division is sorted by the order identifier "Pp". These sorted activities are executed sequentially. When the process is completed normally, the completion is posted to the state manager, which in turn updates the state of the object according to the information passed to the state manager from the procedure manager.

Consistency of the office procedures stored in ELISE must be considered. Any inconsistency or error contained in the office procedures should be pointed out to the user. The procedure manager performs this task. The use of the relational data base management system [1] facilitates various checks of the procedures, as well as the ease of catalog, adding, and modification operation, Several examples are given in the following:

(1) Detailed contents of the procedure can be obtained using JOIN operation in the relational algebra on the procedure relation, the event relation, and the activity relation.

(2) Combination of event "IF" and state "WHEN" in the procedure relation must be unique. This condition can be checked using the PROJECTION operation.

(3) Element of the event "IF" catalogued in the procedure relation "P" must also be cataloged in the "Eid" in the event relation E, and the converse is also true. These conditions can be checked using the DIFFERENCE operation on the two sets.

(4) Elements of the process "THEN" catalogued in the procedure relation "P" must also be catalogued in the "Pid" in the activity relation "A", and the converse is also true. These can be checked by the same method as in (3) above.

5. Example

Suppose that a secretary for a department manager asks each of several section managers how much expense budget will be necessary by the end of the fiscal half year. The secretary sends a memo to each section manager requesting his/her estimate.

To define the office procedure in ELISE, following four steps are required.

STEP 1: Description of the office work.

(1) Approximately one month before the end of the fiscal half year (that is February 21), the secretary sends a memo to each of, say, three section managers for him

to fill in the estimated total expense budget required by the end of the half year period.

(2) When a memo is returned from a section manager, the secretary writes down the estimate on a work sheet.

(3) A few days after sending the first memo (that is February 24), the secretary sends a second prompting memo to any section manager who has not returned the first memo.

(4) When answer memos have been received from all the section managers, the secretary computes the total expense, using the work sheet.

STEP 2: Identification of the objects involved.

The focal object in this procedure is the work sheet.

STEP 3: Identification of the event involved.

This procedure involves the following three events:

e1: arrival of the date, "February 21",

e2: arrival of the date, "February 24",

e3: return of the memo to the secreatary.

Step 4: Identification of the state of focal objects.

State of the work sheet:

—before February 21 (not-active),
—no section manager has turned in the reply (waiting),
—one section manager has turned in the reply (one-received),
—two section managers have turned in the reply (two-received),
—three section managers have turned in the reply (end).

As a result, this example procedure is represented in Table II-V. OA software module in the prototype system [9] is HITACHI VOS3/EXCEED (Executive Management Decision Support System). EXCEED is a command language for end-user: database manipulation, statistical analysis, business graph, mailing, and so on [4]. The command syntax in Table IV differs from precise syntax of EXCEED for readability.

Table II Procedure Relation for the Example.

IF	WHEN	THEN	GOTO
e1	not-active	a1	waiting
e2	waiting	a2	waiting
e2	one-received	a2	one-received
e2	two-received	a2	two-received
e3	waiting	a3	one-received
e3	one-received	a3	two-received
e3	two-received	a4	end

IF: event code, THEN: process code, WHEN, GOTO: state, "end" is final state

Table III Event Relation for the Example.

Eid	Type	R	C
e1	time	=	February 21
e2	time	=	February 24
e3	receive	=	memo-1

Eid: event code, Type: event type, R: logical operator, C: constant

Table IV Activity Relation for the Example.

Pid	Pp	Ps	Aid
a1	p11	1	create memo-1 for section-a
a1	p11	2	send memo-1 to section-a-manager
a1	p12	1	create memo-1 for section-b
a1	p12	2	send memo-1 to section-b-manager
a1	p13	1	create memo-1 for section-c
a1	p13	2	send memo-1 to section-c manager
a2	pa2	1	send memo-2 to section manager who did not return memo-1
a3	pa3	1	load work-sheet from file-0
a3	pa3	2	copy memo-1 to work-sheet
a3	pa3	3	save memo-1 to file-1
a3	pa3	4	save work-sheet to file-0
a4	pa4	1	load work-sheet from file-0
a4	pa4	2	copy memo-1 to work-sheet
a4	pa4	3	calculate work-sheet
a4	pa4	4	save work-sheet to file-0
a4	pa4	5	save memo-1 to file-1

Pid: process code, Pp: code for parallel processing, Ps: order for serial processing, Aid: command for processing

Table V State Relation for the Example.

Oid	Sid
work-sheet	not-active

Oid: object, Sid: state

6. Conclusion and Discussion

The architecture of a system for office procedure automation is discussed which stores office procedures and executes them in an event-driven manner. Representation of office procedures is based on the state-transition model. Introduction of the state concept broadens the range of office works which can be automated.

In our experimental system, office procedures are represented not in a programming language, but as a relation based on the state-transition model. The representation based on this model is easy for a non-programming user to specify, and the relational structure for storing the office procedure facilitates the retrieval, addition, editing, deletion, and consistency check of the office procedures.

A prototype system, called ELISE, was developed to implement the architecture. The proposed architecture consists of the event monitor, procedure monitor, dispatcher, state manager, procedure manager, and observer. The function of each components and interfaces between the components are discussed in some detail. And function described were confirmed by automating example procedure. The presented architecture allows the system and the user to cooperate with

each other to obtain, store, and execute office procedures. Until now, the observer function has not yet been implemented in the prototype system. It is not easy for us to implement the observer, but how to implement it will be an interesting topic.

Improvements in office productivity will be realized by a cooperative execution of office work by office workers and integrated systems for office automation. There must be a simple but powerful representation language for office procedures.

Acknowledgement

Authors would like to express sincere thanks to Dr. J. Kawasaki, Mr. Y. Aoyama for their valuable guidance and encouragement. Special thanks are also due to Dr. T. Sato for his useful advices and valuable discussion on the manuscript.

References

1. DATE, C. J. An Introduction of Database Systems, Third Edition, Addision Wesley (1981).
2. ELLIS, C. A. and NUTT, G. J. Office information systems and computer sciences. *ACM Comput. Surv.* **12**, 1 (1980), 3-36.
3. HAMMER, M. and KUNIN, J. S. Design principles of an office specification language, AFIPS Conference, *Proc. of NCC* (1980), 541-547.
4. ISOBE, H. and YAMASHITA, K. EXCEED—Exective Management Decision Support System, *Proc. of APL Users Meetings*, **1** (1982), 296-324.
5. LEFKOVITZ, H. C. et al. A Status Report on the Activities of the CODASYL End User Facilities Committee (EUFC), *SIGMOD RECORD*, **10** (1979).
6. SHU, N. C. FORMAL: A Form-Oriented Visual Directed Application Development System, *IEEE COMPUTER, August* (1985), 38-49.
7. SIRBU, M., SCHOICHET, S., KUNIN, J. S., Hammer, M. and Sutherland, J. OAM: An Office Analysis Methodology, *AFIPS Office Automation Conference* (1982), 317-330.
8. TSICHRITZIS, D. C. Form Management, *Comm. ACM.* **25**, 7 (1982), 453-478.
9. TSUJI, H. and Mori, F. Table-Based Expert System for Office Automation, *Proc. 34th Annual Convention IPS Japan* (1987), 1593-1594.
10. ZISMAN, M. D. Representation, Specification, and Automation of Office Procedures, Ph. D. Dessertation. Department of Decision Sciences, The Wharton School, University of Pennsylvania (1977).
11. ZISMAN, M. D. Office Automation: Revolution or Evolution ?, *MIT Sloan Management Review*, **19** (1978), 1-16.
12. ZLOOF, M. M. Office-by Example: A Business Language that Unifies Data and Word Processing and Electronic Mail, *IBM SYST J.*, **21**, 3 (1982), 272-304.

(Received April 26, 1987; revised August 10, 1988)