ProceeDING of The Conference on office Information Systems

AN OIS MODEL FOR INTERNAL CONTROL EVALUATION

Andrew D. Bailey, Jr.
University of Minnesota

James Gerlach
Andrew B. Whinston
Purdue University

R. Preston McAfee
University of Western Ontario

This article presents a precedence model for specifying accounting office systems. Formal analysis procedures are formulated for evaluating the internal controls of the modeled system. The procedures establish precondition and postcondition relationships between select control points.

Key Words and Phrases: Accounting Internal Control, Precedence Model, Precondition, Postcondition

1. INTRODUCTION

Satisfactory accountability, that is, a control structure which protects corporate assets from theft, misuse, and fraud, is an important aspect of office information system design. The OIS design goals of flexibility, efficiency, and modularity must not preclude the accountability needs of managers, stockholders and auditors. This will require meticulous examination of the OIS to ensure that it satisfies the multi-attribute control criteria used by auditors.

Currently, requirements are met by employing traditional flowchart descriptions, questionnaires with a focus on potential weaknesses and exposures, narrative descriptions, decision tables, and other methods that seem appropriate to the individual auditor in each situation. The review and evaluation stage of this process is largely limited to human analytic capabilities, which are surprisingly deep, broad, and complex - but extremely varied and unreliable.

New techniques are needed to support the auditor in the review and evaluation process. Many of these techniques will take advantage of computer technology and its capacity to deal swiftly and accurately with highly complex systems of relationships. Our approach is to model the firm's internal behavior, of which the OIS is a subset, using the computer-acceptable Internal Control Description Language. The model is then analyzed by machine to see that it satisfies auditing criteria. In this way, our system - called TICOM-II - incorporates both OIS theory and auditing criteria.

The TICOM-II modeling and analysis approach to unifying OIS and auditing issues has four distinct components. The first, the Internal Control Description Language, is a modeling language for formally describing a firm's operations. The formal input it provides for the TICOM-II modeling process is consistent with the information collected by Deloitte, Haskins, and Sells in their manual verification of internal control procedures [7]. This formal model is then mapped algorithmically into an internal representation, the second component of the system. The internal representation was designed to facilitate analysis, the third major component of TICOM-II. The analysis methods are controlled by the fourth component, a query processing system that permits the auditors to pose questions concerning the internal control model's behavior.

Under a grant from the Peat, Marwick, and Michell Foundation, a prototype of TICOM-II is being implemented at the University of Minnesota to determine the feasibility of computer-assisted internal control evaluation.

2. ANALYSIS AND COMPLEXITY

The types of questions posed by auditors during internal control evaluation are diverse, many of the questions concern state achievability. That is, is it possible for the firm, given its internal control structure, to reach a particular state. If it is possible to enter such a state, TICOM-II analysis establishes a precondition for entering that state and a postcondition that is necessarily true when the state is achieved. The former permits analysis of the strength of safeguards in the system, while the latter allows identification of what to examine in regards to identifying the perpetrator when a control circumvention is suspected.

Precondition and postcondition evaluation of accounting models is closely related to the analysis of programs. As such, automated accounting internal control evaluation suffers from many of the same ills that cripples program verification [8]. Of particular concern is the complexity of the analysis both in terms of machine calculations and the ease with which the internal control evaluator can comprehend the results of the analysis. To combat these problems, several measures have been taken.

The firm's activities are modeled at a high-level of abstraction which focuses on the major details of the system. The system is described in terms of objects (including documents) and agents' conditional access rights and processing responsibilities that control their use. Details of office procedures are suppressed in favor of a simpler firm-wide perspective. Once the internal control evaluator understands the sequencing of office procedures, detailed examination of the office system can proceed on an individual office procedure basis. In addition, TICOM-II has a system simplification facility for creating "black boxes". A black box is a simplified component of the firm-wide model in which all but the most essential features are suppressed. In effect, system simplification reduces the complexity of the internal control system to a more manageable size while maintaining a system-wide perspective. Two popular approaches to internal control evaluation, the cycles approach and the account classification approach, are based on this technique. Finally, the modelling and analysis of parallelism is restircted to non-interfering office activities. That is, if two or more office activities can be performed in parallel, all permissible execution sequences terminate with identical results. Such models are referred to as semi-commutative models. The reasonableness of this restriction is later argued from an accounting perspective.

In summary, verification of an accounting internal control system is an NP-hard problem [5]. Yet auditors are required to review and evaluate them. It is our contention that the review and evaluation process can be effectively aided by computer-assisted tools such as TICOM-II and that formal modeling and analysis is the first step towards the necessary imposition of accounting internal controls on an operating OIS.

3. INTERNAL CONTROL DESCRIPTION LANGUAGE

The Internal Control Description Language was designed to support description specifications of accounting internal control systems. Its constructs and terminology are closely related to the fundamental concepts and operations associated with internal control and systems design.

The ICDL commands model the manipulation of objects which are referenced by name. Each object is comprised of labelled attributes that represent the various components of the object. A typical object is a document whose fields are identified as attributes. The attribute type specifies the nature of the data that the attribute contains. Thus, a document is viewed as a collection of attributes, or equivalently, as a collection of variables. For additional information concerning the ICDL and TICOM-II consult Bailey, et. al. [1, 2, 3, 4].

4. BASIC MODELING AND ANALYSIS CONCEPTS

Given an OIS specification in ICDL, the next phase is to construct an office model from the ICDL description. Due to the type of analysis to be performed on the model, a precedence-oriented model was developed.

Precedence-oriented models depict the office as a set of tasks whose permissible execution sequences are specified by precedence constraints. The general form of the model is a bilogic directed graph showing both control and data flow. The precedence-oriented model has served as the basis for the Information Control Net office model developed at Xerox PARC [10]. Other OIS modeling techniques have been proposed. Two other such models are Zisman's argumented Petri nets [12] and Omega [6].

Each node of the graph represents some fundamental operation such as document preparation or a consistency test between two documents. The time at which a node is activated is governed by the completion of its immediate predecessor activities. Immediate predecessors of a node, α, are those nodes whose outgoing arcs point to α. A given node may have more than one precedence condition. Multiple precedence conditions are specified through logical expressions of the incoming arcs. A conditional node is restricted to having only two outgoing arcs; one labeled true and the other false, to denote which arc and thus which immediate predecessor is to be activated dependent upon the outcome of the test.

Figure 1 shows a simplified fragment of a purchasing subsystem. The example does not illustrate the complexity of accounting internal control systems. Missing from the model are the interactions of the various accounting systems, the clerical and managerial positions within each department, documentation for recording transactions, descriptions of validation, authorization and approval procedures, and feedback mechanisms for correcting identified irregularities and errors. However, it is adequate for demonstrating the relationship between the ICDL specification and the precedence graph model. The ICDL procedural description consists of five organizational units: VENDOR, RECEIVING, PURCHASING, STORES and CASH DISBURSEMENTS. Modular task descriptions specify the processing capabilities of each organizational unit. The precedence relations of each task are implied by the serial ordering of the task's instructions and information flows between the task and other tasks and repositories. The initial contents of each repository is given within the ICDL description.

state. If it is possible to enter such a state, TICOM-II analysis establishes a precondition for entering that state and a postcondition that is necessarily true when the state is achieved. The former permits analysis of the strength of safeguards in the system, while the latter allows identification of what to examine in regards to identifying the perpetrator when a control circumvention is suspected.

Precondition and postcondition evaluation of accounting models is closely related to the analysis of programs. As such, automated accounting internal control evaluation suffers from many of the same ills that cripples program verification [8]. Of particular concern is the complexity of the analysis both in terms of machine calculations and the ease with which the internal control evaluator can comprehend the results of the analysis. To combat these problems, several measures have been taken.

The firm's activities are modeled at a high-level of abstraction which focuses on the major details of the system. The system is described in terms of objects (including documents) and agents' conditional access rights and processing responsibilities that control their use. Details of office procedures are suppressed in favor of a simpler firm-wide perspective. Once the internal control evaluator understands the sequencing of office procedures, detailed examination of the office system can proceed on an individual office procedure basis. In addition, TICOM-II has a system simplification facility for creating "black boxes". A black box is a simplified component of the firm-wide model in which all but the most essential features are suppressed. In effect, system simplification reduces the complexity of the internal control system to a more manageable size while maintaining a system-wide perspective. Two popular approaches to internal control evaluation, the cycles approach and the account classification approach, are based on this technique. Finally, the modelling and analysis of parallelism is restircted to non-interfering office activities. That is, if two or more office activities can be performed in parallel, all permissible execution sequences terminate with identical results. Such models are referred to as semi-commutative models. The reasonableness of this restriction is later argued from an accounting perspective.

In summary, verification of an accounting internal control system is an NP-hard problem [5]. Yet auditors are required to review and evaluate them. It is our contention that the review and evaluation process can be effectively aided by computer-assisted tools such as TICOM-II and that formal modeling and analysis is the first step towards the necessary imposition of accounting internal controls on an operating OIS.

3. INTERNAL CONTROL DESCRIPTION LANGUAGE

The Internal Control Description Language was designed to support description specifications of accounting internal control systems. Its constructs and terminology are closely related to the fundamental concepts and operations associated with internal control and systems design.

The ICDL commands model the manipulation of objects which are referenced by name. Each object is comprised of labelled attributes that represent the various components of the object. A typical object is a document whose fields are identified as attributes. The attribute type specifies the nature of the data that the attribute contains. Thus, a document is viewed as a collection of attributes, or equivalently, as a collection of variables. For additional information concerning the ICDL and TICOM-II consult Bailey, et. al. [1, 2, 3, 4].

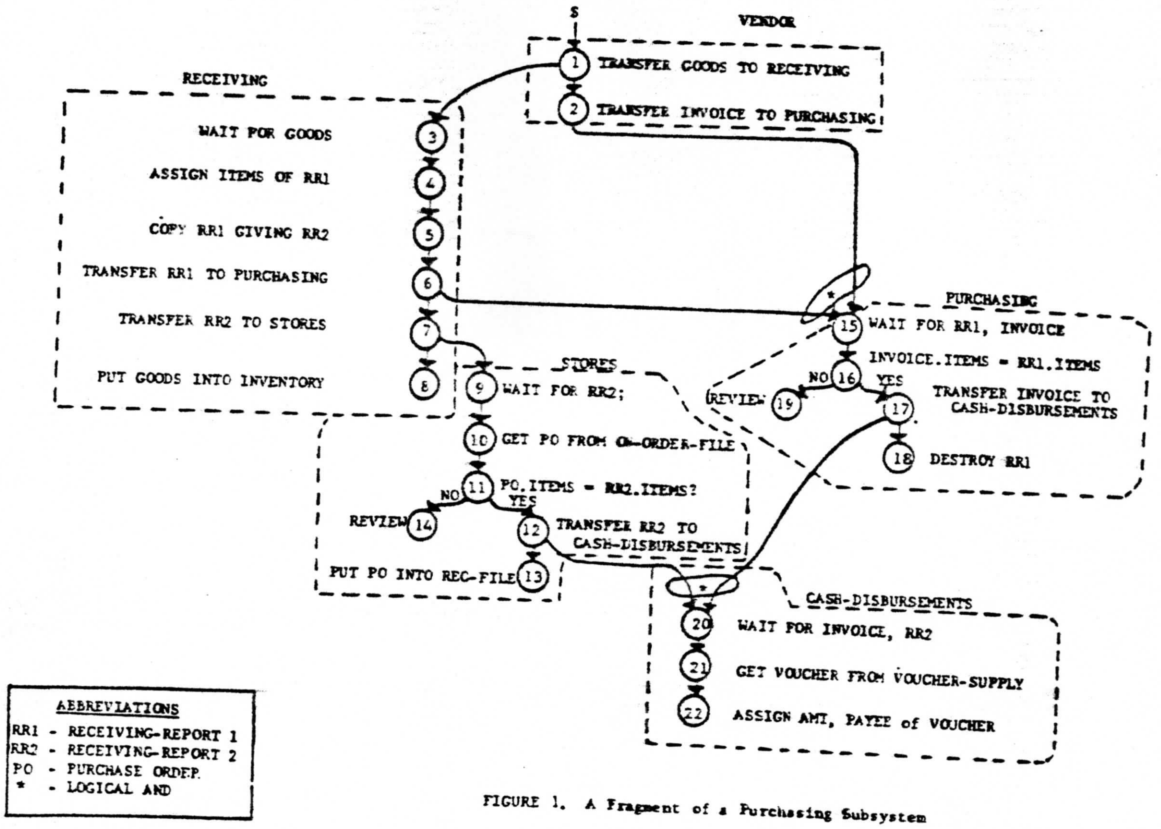
4. BASIC MODELING AND ANALYSIS CONCEPTS

Given an OIS specification in ICDL, the next phase is to construct an office model from the ICDL description. Due to the type of analysis to be performed on the model, a precedence-oriented model was developed.

Precedence-oriented models depict the office as a set of tasks whose permissible execution sequences are specified by precedence constraints. The general form of the model is a bilogic directed graph showing both control and data flow. The precedence-oriented model has served as the basis for the Information Control Net office model developed at Xerox PARC [10]. Other OIS modeling techniques have been proposed. Two other such models are Zisman's argumented Petri nets [12] and Omega [6].

Each node of the graph represents some fundamental operation such as document preparation or a consistency test between two documents. The time at which a node is activated is governed by the completion of its immediate predecessor activities. Immediate predecessors of a node, α, are those nodes whose outgoing arcs point to α. A given node may have more than one precedence condition. Multiple precedence conditions are specified through logical expressions of the incoming arcs. A conditional node is restricted to having only two outgoing arcs; one labeled true and the other false, to denote which arc and thus which immediate predecessor is to be activated dependent upon the outcome of the test.

Figure 1 shows a simplified fragment of a purchasing subsystem. The example does not illustrate the complexity of accounting internal control systems. Missing from the model are the interactions of the various accounting systems, the clerical and managerial positions within each department, documentation for recording transactions, descriptions of validation, authorization and approval procedures, and feedback mechanisms for correcting identified irregularities and errors. However, it is adequate for demonstrating the relationship between the ICDL specification and the precedence graph model. The ICDL procedural description consists of five organizational units: VENDOR, RECEIVING, PURCHASING, STORES and CASH DISBURSEMENTS. Modular task descriptions specify the processing capabilities of each organizational unit. The precedence relations of each task are implied by the serial ordering of the task's instructions and information flows between the task and other tasks and repositories. The initial contents of each repository is given within the ICDL description.


S
VENDOR
1 TRANSFER GOODS TO RECEIVING
2 TRANSFER INVOICE TO PURCHASING
RECEIVING
WAIT FOR GOODS 3
ASSIGN ITEMS OF RR1 4
COPY RR1 GIVING RR2 5
TRANSFER RR1 TO PURCHASING 6
TRANSFER RR2 TO STORES 7
PUT GOODS INTO INVENTORY 8
*
PURCHASING
15 WAIT FOR RR1, INVOICE
INVOICE.ITEMS = RR1.ITEMS
NO 16 YES
REVIEW 19
17 TRANSFER INVOICE TO CASH-DISBURSEMENTS
18 DESTROY RR1
STORES
9 WAIT FOR RR2;
10 GET PO FROM ON-ORDER-FILE
NO 11 PO.ITEMS = RR2.ITEMS? YES
REVIEW 14
12 TRANSFER RR2 TO CASH-DISBURSEMENTS
PUT PO INTO REC-FILE 13
*
CASH-DISBURSEMENTS
20 WAIT FOR INVOICE, RR2
21 GET VOUCHER FROM VOUCHER-SUPPLY
22 ASSIGN AMT, PAYEE of VOUCHER

ABBREVIATIONS	
RR1	- RECEIVING-REPORT 1
RR2	- RECEIVING-REPORT 2
PO	- PURCHASE ORDER
*	- LOGICAL AND

FIGURE 1. A Fragment of a Purchasing Subsystem

Regarding Figure 1, a usable precondition for the preparation of a voucher (node 22) is: i) Stores claims that the items listed on the purchase order correspond with the items received as reported by Receiving and ii) Purchasing claims that the items listed on the invoice (i.e., items shipped by vendors) also correspond with the items received as reported by Receiving. A usable postcondition for the same event is: i) Stores claims that the items on the purchase order correspond with the items on receiving report 2. The assertion made at node 16 is omitted since RR1 is destroyed at node 18 which may precede node 22 in executing. If the integrity of RR2 is assured then the assertion could be included by substituting RR2 for RR1 since they are duplicates.

5. INTERNAL REPRESENTATION

In general each ICDL instruction is uniquely labeled and formally modeled by one or more PC-elements which list a precedence condition for the execution of the instruction and its execution effects in terms of variable assignments. Each PC-element is also identified by a unique index. Each PC_α is mapped to its corresponding ICDL instruction by a special function specified as id. If $id(\alpha) = j$ then PC_α lists a condition upon which ICDL instruction j, denoted I_j, can be reached. All PC-elements are of the form $PC_\alpha = (N_\alpha; R_\alpha; AD_\alpha)$. N_α is a set of indices of immediate predecessor ICDL instructions for PC_α. R_α is the condition under which the corresponding ICDL instruction follows these immediate predecessors. AD_α is a set of variable assignments (attribute definitions) that become effective upon the execution of the corresonding ICDL instruction for PC_α. If $v/e \in AD_\alpha$ then e is assigned to the variable v when $I_{id(\alpha)}$ follows the instructions indexed by N_α and R_α is asserted. Since R_α is restricted to be a boolean expression formed from n-ary predicates and the logical connectives $\{\wedge, \sim\}$, the precedence constraint for performing a given ICDL instruction is expressed in disjunctive normal form. That is, given $PC_{\alpha_1}, PC_{\alpha_2}, \ldots, PC_{\alpha_n}$ s.t., $id(\alpha_1) = id(\alpha_2) = \ldots = id(\alpha_n) = j$ and $N_{\alpha_1} = N_{\alpha_2} = \ldots = N_{\alpha_n}$ then the precedence constraint for I_j to follow the ICDL instructions indexed by N_{α_1} is $R_{\alpha_1} \vee R_{\alpha_2} \vee \ldots \vee R_{\alpha_n}$.

5.1 BASIC OIS MODEL

The formal definition of the TICOM-II model for an OIS specified by q ICDL instructions labeled $I_1, I_2, \ldots, I_q$ is given next. The model incorporates principles taken from first order logic [11].

$\mathcal{M} = (U, V, C, P, F, S, PC, id)$

U is the universe containing C, a set of constant symbols denoted $c_1, \ldots, c_m$.

V is a set of variables denoted $v_1, \ldots, v_m$.

P is a set of n-ary predicate symbols and propositional letters denoted $P_1, \ldots, P_m$ and $P_i^n: U^n \to \{T, F\} \subseteq U$.

F is a set of n-ary function symbols denoted $f_1, \ldots, f_m$ and $f_i^n: U^n \to U$.

S is a special root index that is not an element of $\{1, \ldots, q\}$ for some given q.

PC is a set of elements denoted $PC_1, \ldots, PC_m$ and each

$PC_\alpha = (N_\alpha; R_\alpha; AD_\alpha)$ where

N_α is a subset of $\{1, \ldots, q\} \cup \{S\}$

R_α is an element of the set of well-formed formulas built from V, C, F, P and the logical symbols $\{\wedge, \sim\}$

AD_α is a set of variable assignments where each element of AD_α is of the form v_i/t denoting that variable v_i is bound to t. Or equivalently, that t is substitutable for v_i.

id is a unary function that maps indices of PC to $\{1, \ldots, q\}$.

5.2 NOTATION

Substitution: If e is a term and x is a variable then φ_e^x is the result of substituting e for all free occurrences of x in φ.

Variable Binding: If v is a variable and t is a term then v/t denotes that v is bound to t, or equivalently, t is substitutable for v.

Operator "//": If R is an element of the set of wffs and $AD_\alpha = \{v_1/t_1, v_2/t_2, \ldots, v_n/t_n\}$ and $AD_\beta = \{\hat{v}_1/\hat{t}_1, \hat{v}_2/\hat{t}_2, \ldots, \hat{v}_q/\hat{t}_q\}$ and v_i, $\hat{v}_i$ are variables and t_i, $\hat{t}_i$ are terms then

$$R//AD_\alpha = R_{t_1 t_2 \ldots t_n}^{v_1 v_2 \ldots v_n}$$

and

$AD_\beta//AD_\alpha$ results in the set Z defined below:

i) for i = 1, q: $\hat{v}_i/(\hat{t}_i)_{t_1 t_2 \ldots t_n}^{v_1 v_2 \ldots v_n}$ is the i-th element in Z and

ii) for i = 1, n: if there does not exist a $\hat{v}_j/\hat{t}_j \in AD_\beta$ s.t. $v_i = \hat{v}_j$ then v_i/t_i is the next right most element in Z.

In the first case, the operator "//" is used to substitute variable definitions of AD_α for free occurrences of variables in R. In the second case the operator is used to combine the computations encoded in the AD sets of two PC-elements under the assumption that PC_β immediately follows PC_α in the execution sequence. Thus, if the execution sequence PC_1, PC_2, PC_3 is given and $AD_1 = \{z/y\}$, $AD_2 = \{y/c\}$ and $AD_3 = \{x/f(z)\}$ then $AD_3//AD_2//AD_1 = \{x/f(y), y/c, z/y\}$. For the execution sequence PC_2, PC_1, PC_3,

$AD_3//AD_1//AD_2 = \{x/f(c), z/c, y/c\}$. Thus the "//" operator is noncommutative, but it is associative since it preserves execution order effects.

6. ANALYSIS OF THE INTERNAL REPRESENTATION

State: a state reachable in $\mathcal{M}$ is depicted by the triple (N; R; AD) where N is an element of the set of ICDL instruction indices $\{1, \ldots, q\}$, R is a well-formed formula, and AD is a set of variable assignments.

State Achievability: $\mathcal{M}$ achieves (N; R; AD) iff there exists a finite sequence $PC_{\alpha_1}, PC_{\alpha_2}, \ldots, PC_{\alpha_n}$ such that

i) if $j \le n$ then $\forall i \in N_{\alpha_j}$ either i = S, the root index, or $\left\{ \begin{array}{l} \text{there exists a } k<j \text{ s.t. } id(\alpha_k) = i \\ \text{and for all } \ell \text{ s.t. } k<\ell<j,\ \alpha_\ell \ne \alpha_j \end{array} \right.$

and

ii) $N = id(\alpha_n)$

$R = R_{\alpha_1} \wedge (R_{\alpha_2}//AD_{\alpha_1}) \wedge \ldots \wedge (R_{\alpha_n}//AD_{\alpha_{n-1}}//\ldots//AD_{\alpha_1})$

$AD = AD_{\alpha_n}//AD_{\alpha_{n-1}}//\ldots//AD_{\alpha_1}$

and

iii) R is satisfied under interpretation s.

According to the preceding definitions of state and state achievability, a state is reached by applying elements of PC in some order that honors the precedence constraints implicit in the ICDL description of the model. Restriction (i) guarantees that for each occurrence of PC_{α_j} in the sequence that PC_{α_j} is preceded by its immediate predecessors with no intervening occurrences of PC_{α_j} (PC_{α_j} may occur at most once for each occurrence of its immediate predecessors in the sequence). Note that this restriction does not rule out loops since a PC-element may occur in a sequence each time its precedence conditions are met. Restriction (ii) designates that the last PC-element in the series must be associated with I_N, R is the condition for PC_{α_n} to be reached via the sequence and AD is the set of variable assignments that are in effect upon the completion of PC_{α_n}. The evaluation of R and AD is based on and consistent with Dijkstra's notion of weakest precondition [9]. The last restriction (iii), limits the states that are reachable to those states whose conditions for reachability are satisfiable under some interpretation s. The interpretation s is assumed to be specified by the internal control evaluator. Thus, for state (N; R; AD) to be reached via $PC_{\alpha_1}, \ldots, PC_{\alpha_n}$, the condition R for the initial state of the model must be true.

Unfortunately, the satisfiability of R at the time the model is activated does not guarantee that state (N; R; AD) will be reached. This is due to uncontrolled concurrent processing. Consider $PC_\alpha = (\{i\}; \emptyset; \{x/t_1\})$ and $PC_\beta = (\{i\}; \emptyset; \{x/t_2\})$. Clearly after the common immediate predecessor constraint is satisfied, either PC_α could precede PC_β or vice versa. Since PC_α, PC_β is not generally equivalent to PC_β, PC_α, at the activation of the model it cannot be ascertained as to which order PC_α and PC_β will occur. And therefore state (N; R; AD) cannot be guaranteed.

Of utmost concern to the auditor and accountant is the reliability, accuracy and consistency of accounting information. This requires an accounting information system to reliably capture accurate accounting information, verify it against other relevant accounting information for consistency, and store it to document the validity of the business event it supports. With these objectives in mind, it is unimaginable that any auditor or accountant would choose an office system whose results and intermediate operations are partially controlled by arbitrarily ordered interacting office activities. Yet, accountants and auditors are well aware of the importance and need for parallel processing in an office system. As a compromise to this dilemma, TICOM-II analysis is restricted to semi-commutative models.

Semi-Commutative Model: is a basic OIS model as defined previously with the added restriction that if the sequence, S_1, achieves (N; R; AD)

$$S_1 = PC_{\alpha_1}, \ldots, PC_{\alpha_{j-1}}, PC_{\alpha_j}, PC_{\alpha_{j+1}}, \ldots, PC_{\alpha_n} \text{ and}$$

$id(\alpha_j) \notin N_{\alpha_{j+1}}$ then the sequence, S_2, formed by switching PC_{α_j} and $PC_{\alpha_{j+1}}$ also achieves (N; R; AD)

$$S_2 = PC_{\alpha_1}, \ldots, PC_{\alpha_{j-1}}, PC_{\alpha_{j+1}}, PC_{\alpha_j}, \ldots, PC_{\alpha_n}$$

and is therefore equivalent to S_1.

Thus, the semi-commutative model prevents interacting PC-elements from being arbitrarily ordered by requiring the scheduling of such PC-elements to be deterministically encapsulated in their respective N and R components. This restriction does not prohibit the sharing of information between instructions (PC-elements) executing concurrently. It does, however, prevent an instruction from updating a variable before all users of this instance of the variable have completed their operations. The concept of precondition follows from the semi-commutative model. By definition of state achievability, if $\mathcal{M}$ is a semi-commutative model and achieves (N; R; AD) then R is a precondition for that state.

7. ANALYTIC CAPABILITIES OF TICOM-II

The preceding discussion has dealt with the analysis of given permissible execution sequences. Evaluating accounting systems from a state achievability perspective requires the identification of the goal state under study and consideration of all the execution sequences that lead to that goal. In TICOM-II analysis, the goal state of the system is identified by critical ICDL instructions and restrictions placed upon sequences leading to the goal state. By expressing this subsystem of critical ICDL instructions in terms of precondition and postcondition relationships, the underlying control structure governing these commands is made obvious. This capability is an important feature of TICOM-II since it is supportive of the cycles approach and the account classification approach to internal control evaluation.

Precondition: a precondition for ICDL instruction N is defined to be a condition for the initial state of the system such that activation of the semi-commutative model guarantees that some sequence $PC_{\alpha_1}, \ldots, PC_{\alpha_n}$ will be generated achieving (N; R; AD).

Postcondition: a postcondition for ICDL instruction N is a condition for the state of the system that is necessarily true immediately after the completion of the sequence $PC_{\alpha_1}, \ldots, PC_{\alpha_n}$ of the semi-commutative model which achieves (N; R; AD).

This definition of precondition is consistent with the concept of precondition that was presented earlier. The postcondition that seems most desirable from an internal control evaluator's viewpoint is one which can be resubstantiated by viewing the contents of the objects at the time an ICDL instruction has finished execution.

In order to evaluate postconditions, the PC-element is expanded to $PC_\alpha = (N_\alpha; R_\alpha; AD_\alpha; B_\alpha; T_\alpha)$. N_α, R_α and AD_α are defined as before. B_α is a set of variables that have been or could have been redefined on the path from N_α to PC_α. T_α is a well-formed formula that expresses a postcondition relationship between variables that necessarily exists after PC_α follows its immediate predecessors, N_α. B_α and T_α are initialized as follows:

i) If $x/t \in AD_\alpha$ then $x \in B_\alpha$

and

ii) If there exists a β s.t., $N_\beta \subseteq N_\alpha$ and $R_\alpha \wedge R_\beta$ is satisfied under interpretation s then B_β is contained in B_α

and

iii) $T_\alpha = R_\alpha - B_\alpha$ where T_α is a conjunction of predicates of R_α (and their negation signs), all of whose arguments do not appear in B_α.

The logic behind (i) above is simple, if PC_α redefines x then x is obviously redefined on the path from N_α to PC_α and belongs in B_α. The reasoning behind (ii) is slightly more complicated. If PC_β's immediate predecessors are also immediate predecessors of $PC_\alpha (N_\beta \subseteq N_\alpha)$ and PC_α and PC_β could occur in the same execution sequence ($R_\alpha \wedge R_\beta$ is satisfied under interpretation s) then PC_β can occur before PC_α and any variables PC_β redefines need to be included in B_α. Finally (iii) drops any previously established relationship that contains variables that might have been or were redefined since the relationship was established.

Also, if PC_β follows PC_α with T_β asserted and B_β lists the variables redefined by PC_β and T_α is the condition still asserted upon completing PC_α then $(T_\alpha - B_\beta) \wedge T_\beta$ is a postcondition that is asserted by the sequence PC_α, PC_β.

Procedure 2 - PC-Simplification: PC-simplification is a process by which PC-elements for any I_j are reduced to a simpler but equivalent form according to the following reduction rules, given PC_α, $PC_\beta \in PC$ and $id(\alpha) = id(\beta) = j$ and $\alpha \neq \beta$. ($R_\alpha = R_1 \wedge R_2$ partitions R_α into two elements. R_1 is an atomic formula and R_2 is an element of the set of well-formed formulas.)

1. If $N_\alpha \subseteq N_\beta$ and $R_\alpha \subseteq R_\beta (R_\beta \rightarrow R_\alpha)$ and $AD_\alpha \subseteq AD_\beta$ and $B_\alpha \subseteq B_\beta$ and $T_\alpha \subseteq T_\beta (T_\beta \rightarrow T_\alpha)$ then drop PC_β from PC and set $B_\alpha = B_\beta$.
2. If $N_\alpha = N_\beta$ and $R_\alpha = R_1 \wedge R_2$ and $R_\beta = \sim R_1 \wedge R_2$ then set $R_\alpha = R_\beta = R_2$.
3. If $N_\alpha = N_\beta$ and $R_\alpha = R_1$ and $R_\beta = \sim R_1 \wedge R_2$ then set $R_\beta = R_2$.
4. If $id(\alpha) \in N_\alpha$ and there does not exist a PC_γ s.t. $id(\alpha) = id(\gamma)$, and $id(\alpha) \notin N_\gamma$ then drop PC_α from PC.
5. If $S \in N_\alpha$ and $|N_\alpha| > 1$ then set $N_\alpha = N_\alpha - \{S\}$
6. If R_α is unsatisfiable for interpretation s then drop PC_α from PC.
7. If $i \in N_\alpha$ and there does not exist a PC_γ s.t. $id(\gamma) = i$ then drop PC_α from PC.

Rules 2 and 3 can easily be adapted for simplifying T_α and T_β.

Rule 1 is valid since it selects the weaker of two conditions for I_j to follow its immediate predecessors listed in N_α. Rule 2 is based on the logical rule $(A \wedge B) \vee (\sim A \wedge B) \equiv B$. Thus, if I_j follows N_α under condition $R_1 \wedge R_2$ and I_j also follows $N_\alpha (N_\alpha = N_\beta)$ under condition $\sim R_1 \wedge R_2$ then it is concluded that I_j follows N_α under condition R_2. Rule 3 is similarly based on the logical rule $A \vee (\sim A \wedge B) \equiv A \vee B$. Rule 4 identifies an unsatisfiable condition, namely that I_j can only be initially reached after it has first been executed. Rule 5 simply eliminates the redundant statement of a restriction if I_j follows N_α then it also follows S. The sixth rule drops contradictory paths from further consideration. Lastly, rule 7 eliminates unreachable instructions and "execution" paths dependent upon them.

Procedure 3 - Loop Elimination for i:

1) For any $PC_{\alpha_1}, PC_{\alpha_2}, \ldots, PC_{\alpha_n}$ s.t. $id(\alpha_1) = \ldots = id(\alpha_n) = i$ and $n > 1$ and $i \notin N_{\alpha_1}$ and $i \in N_{\alpha_j}$ for $2 \leq j \leq n$ add PC_γ to PC s.t. $id(\gamma) = i$ where

$$N_\gamma = (N_{\alpha_1} \cup N_{\alpha_2} \cup \ldots \cup N_{\alpha_n}) - \{i\}$$

$$R_\gamma = R_{\alpha_1} \wedge (R_{\alpha_2} // AD_{\alpha_1}) \wedge (R_{\alpha_3} // AD_{\alpha_2} // AD_{\alpha_1}) \wedge \ldots \wedge (R_{\alpha_n} // AD_{\alpha_{n-1}} // \ldots // AD_{\alpha_1})$$

$$AD_\gamma = AD_{\alpha_n} // AD_{\alpha_{n-1}} // \ldots // AD_{\alpha_1}$$

$$B_\gamma = B_{\alpha_1} \cup B_{\alpha_2} \cup \ldots \cup B_{\alpha_n}$$

$$T_\gamma = (T_{\alpha_1} - (B_{\alpha_2} \cup B_{\alpha_3} \cup \ldots \cup B_{\alpha_n})) \wedge (T_{\alpha_2} - (B_{\alpha_3} \cup B_{\alpha_4} \cup \ldots \cup B_{\alpha_n})) \wedge \ldots \wedge T_{\alpha_n}$$

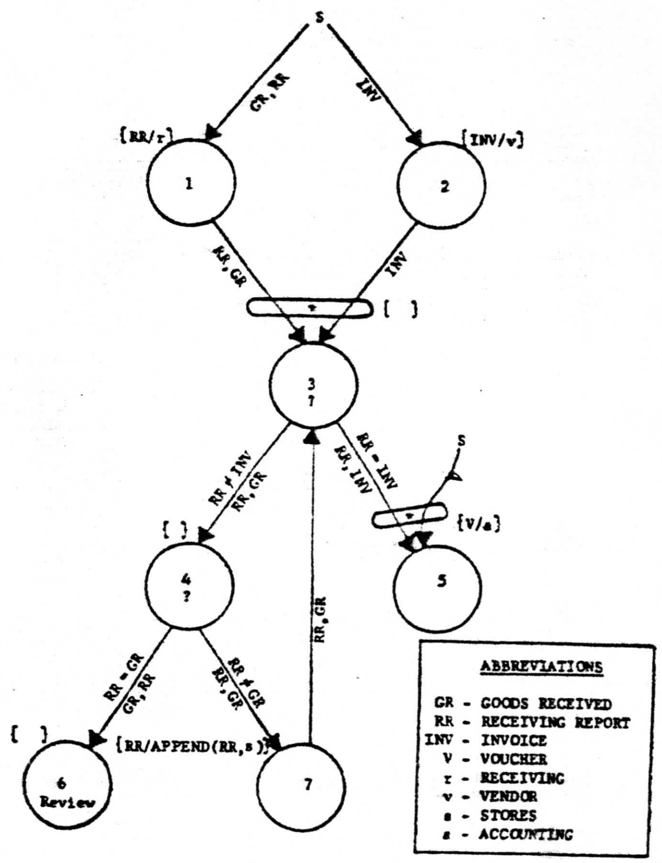
2) Drop all elements from PC s.t. $PC_\alpha \in PC$ and $i \in N_\alpha$.

3) If there exists a PC_β, $PC_\alpha \in PC$ s.t. $\alpha \neq \beta$ and $N_\beta \subseteq N_\alpha$ and $R_\alpha \wedge R_\beta$ is satisfied under interpretation s, then set $B_\alpha = B_\alpha \cup B_\beta$ and $T_\alpha = T_\alpha - B_\beta$.

The loop elimination procedure needs to consider all possible looping paths. In TICOM-II, a finite number of finite paths are selected for analysis that are representative, from an internal control perspective, of all possible looping paths. That is, if $PC_{\alpha_1}, \ldots, PC_{\alpha_k}$ achieves (N, R, AD) and $PC_{\alpha_1}, \ldots, PC_{\alpha_k}, \ldots, PC_{\alpha_w}$ also achieves (N, R, AD) (or an acceptable substitute) then $PC_{\alpha_1}, \ldots, PC_{\alpha_k}$ is representative of both sequences and only needs to be considered. When analysis is restricted to loops containing simple assignment statements (i.e., $x \leftarrow e$ where e is a variable or a constant) then such a representative set of looping paths is enumerable.

8. AN EXAMPLE OF TICOM ANALYSIS

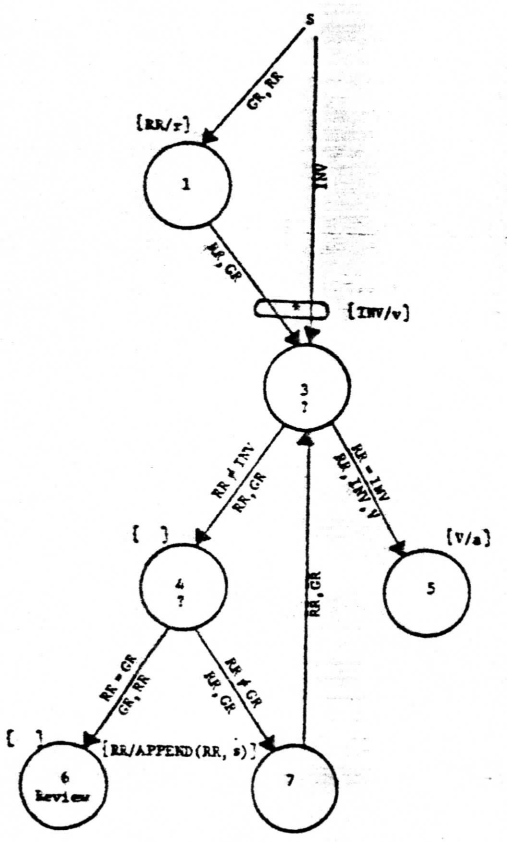
Figure 2a displays an augmented precedence model consisting of seven basic instructions comprising a simple purchasing system. The example is shown graphically and in the internal representation. Listed along the underside of each precedence arc are the objects that are passed between nodes. Any assertions that are made along the arc are listed above it. Associated with each precedence condition is a set of variable assignments. For example, node 1 unconditionally follows the root, S. The receiving report, RR,



id(1) = 1	$PC_1 = (\{S\}; \emptyset; \{RR/r\}; \{RR, INV\}; \emptyset)$
id(2) = 2	$PC_2 = (\{S\}; \emptyset; \{INV/v\}; \{RR, INV\}; \emptyset)$
id(3) = 3	$PC_3 = (\{1, 2\}; \emptyset; \emptyset; \emptyset; \emptyset)$
id(4) = 3	$PC_4 = (\{7\}; \emptyset; \emptyset; \emptyset; \emptyset)$
id(5) = 4	$PC_5 = (\{3\}; RR \neq INV; \emptyset; \emptyset; RR \neq INV)$
id(6) = 5	$PC_6 = (\{3, S\}; RR = INV; \{V/a\}; \{V\}; RR = INV)$
id(7) = 6	$PC_7 = (\{4\}; RR = GR; \emptyset; \emptyset; RR = GR)$
id(8) = 7	$PC_8 = (\{4\}; RR \neq GR; \{RR/APPEND(RR, s)\}; \{RR\}; \emptyset)$

FIGURE 2a

A Semi-Commutative Model



id(1) = 1	$PC_1 = (\{S\}; \emptyset; \{RR/r\}; \{RR, INV\}; \emptyset$
id(3) = 3	$PC_3 = (\{1, S\}; \emptyset ; \{INV/v\}; \{RR, INV\}; \emptyset)$
id(4) = 3	$PC_4 = (\{7\}; \emptyset; \emptyset; \emptyset; \emptyset)$
id(5) = 4	$PC_5 = (\{3\}; RR \neq INV; \emptyset; \emptyset; RR \neq INV)$
id(6) = 5	$PC_6 = (\{3\}; RR = INV; \{V/a\}; \{V\}; RR = INV)$
id(7) = 6	$PC_7 = (\{4\}; RR = GR; \emptyset;\emptyset; RR = GR)$
id(8) = 7	$PC_8 = (\{4\}; RR \neq GR; \{RR/APPEND(RR, s)\}; \{RR\}; \emptyset)$

FIGURE 2b

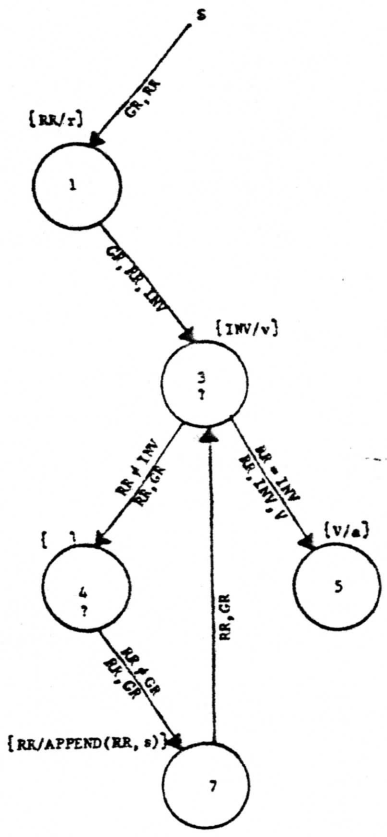
Simplification of Arc $\widehat{S5}$

Elimination of Node 2

and the goods received, GR, are available to node 1. The result of executing node 1 is given by the set, {RR/r}, which stipulates that RR is assigned a value, denoted as r, by the Receiving department. This same information is encoded in the first three components of PC_1. Similarly, node 7 follows node 4 given that RR ≠ GR. The result of performing node 7 is that the RR is appended with a value, denoted as s, by the Stores department. PC_8 contains this same information. All comparisons (nodes 3 and 4) are performed by the Stores department and so subscripting of the relationals to denote the comparer is omitted. Node 5 and PC_6 models the preparation of a voucher, V, by Accounting, a. Node 6 and PC_7 stipulate the entrapment of a discrepancy requiring intervention.

Postcondition information is omitted from the graph but is contained in the B and T components of the PC-elements. For instance, RR and INV are elements of B_1 since nodes 1 and 2 can perform in parallel (the precondition of path S-1 "anded" logically with the precondition of path S-2 is satisfiable). T_1 is empty since no assertions are made along the path S-1. B_6 contains V since V is defined at node 5. T_6 contains RR = INV since it is asserted on the path S-3-5 and none of its arguments are listed in B_6, that is, none of its arguments can be redefined along the path S-3-5. T_8 does not contain RR ≠ GR since RR is updated at node 7.

The initial system shown in Figure 2a is given in terms of preconditions and postconditions. By employing the contraction, loop elimination PC-simplification procedures, a precondition and a postcondition for node 5 is deduced. These analytic procedures are demonstrated in Figures 2b-2h. Figure 2b shows the result of simplifying the precedence condition that node 5



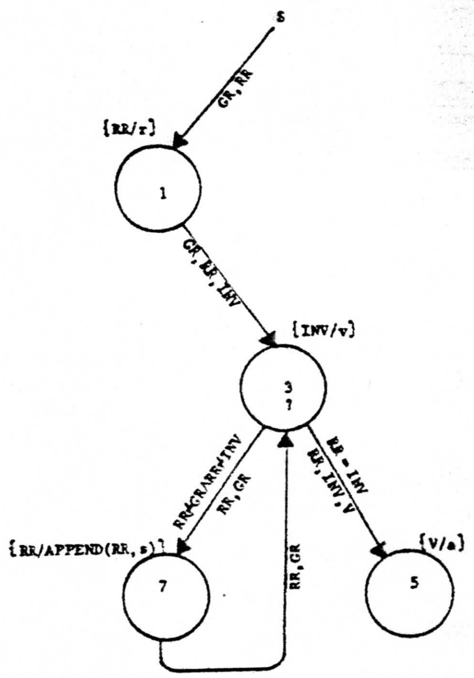
id(1) = 1	$PC_1 = (\{S\}; \emptyset; \{RR/r\}; \{RR, INV\}; \emptyset)$
id(3) = 3	$PC_3 = (\{1\}; \emptyset;;\{INV/v\}; \{RR, INV\}; \emptyset)$
id(4) = 3	$PC_4 = (\{7\}; \emptyset; \emptyset; \emptyset; \emptyset)$
id(5) = 4	$PC_5 = (\{3\}; RR \neq INV; \emptyset; \emptyset; RR \neq INV)$
id(6) = 5	$PC_6 = (\{3\}; RR = INV; \{V/a\}; \{V\}; RR = INV)$
id(8) = 7	$PC_8 = (\{4\}; RR \neq GR; \{RR/APPEND(RR, s)\}; \{RR\}; \emptyset)$

FIGURE 2c

Simplification of Arc $\widehat{S3}$

Elimination of Node 6

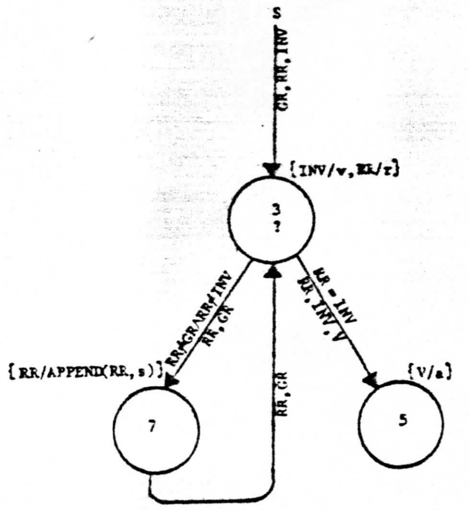
follows the root, S, and node 3. The specification of S is redundant since if node 5 follows node 3 then it must also follow S. Node 2 is removed by pairing the precedence condition of node 2 with the precedence condition of node 2's immediate successor. The intermediate result on the variables by taking the path S-2-3 is given by {INV/v}. In Figure 2c, the precedence condition for node 3 is simplified and node 6 is contracted. Since node 6 has no immediate successors its execution effect is purged from the model. Figure 2d shows the contraction of node 4 which results in the combining of assertions along path 3-4-7, specifically, RR ≠ GR ∧ RR ≠ INV. If RR or GR were assigned values at node 4, their values would have been substituted into RR ≠ GR. The assertion, RR ≠ INV, is not added to T_8 since RR ∈ B_8 which implies that RR is redefined at node 7. Figure 2e illustrates the combining of variable assignments on the path S-1-3. The contraction of node 7 shown in Figure 2f is straightforward. Figure 2g demonstrates the loop elimination technique for removing node 3. Each pass through the loop enables the Stores department to append a value to the receiving report. By regarding APPEND(r, s) as equivalent to APPEND(APPEND(r, s), s), then path S-3-3-5 is representative of all looping paths leading to node 5. That is, node 3 is eliminated by considering the nonlooping path S-3-5 and the looping path S-3-3-5. For each path, variable assignments are appropriately substituted into the assertions and the variable assignment sets are updated along with the other components of the PC-elements. Assertions such as RR/r = INV/v is interpreted as s claims that RR, as prepared by r, matches the INV, as prepared by v. At this point the precondition for attaining node 5 is (RR/r = INV/v) ∨ (RR/r ≠ GR ∧ RR/r ≠ INV/v ∧ RR/APPEND(r, s) = INV/v). This logically reduces to the precondition shown in Figure 2h, another example of PC-simplification. Regardless of the path taken to node 5, its postcondition is RR = INV.



id(1) = 1	$PC_1 = (\{S\}; \emptyset; \{RR/r\}; \{RR, INV\}; \emptyset)$
id(3) = 3	$PC_3 = (\{1\}; \emptyset; \{INV/v\}; \{RR, INV\}; \emptyset)$
id(4) = 3	$PC_4 = (\{7\}; \emptyset; \emptyset; \emptyset; \emptyset)$
id(6) = 5	$PC_6 = (\{3\}; RR = INV; \{V/a\}; \{V\}; RR = INV)$
id(8) = 7	$PC_8 = (\{3\}; RR \neq INV \wedge RR \neq GR; \{RR/APPEND(RR, s)\}; \{RR\}; \emptyset)$

FIGURE 2d

Elimination of Node 4



id(3) = 3 PC_3 = ({S}; ∅; {INV/v, RR/r}; {RR, INV}; ∅)

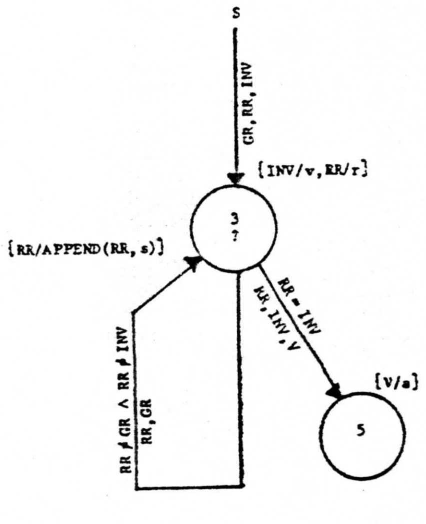
id(4) = 3 PC_4 = ({7}; ∅; ∅; ∅; ∅)

id(6) = 5 PC_6 = ({3}; RR = INV; {V/s}; {V}; RR = INV)

id(8) = 7 PC_8 = ({3}; RR ≠ INV ∧ RR ≠ GR; {RR/APPEND(RR, s)}; {RR}; ∅)

FIGURE 2e

Elimination of Node 1



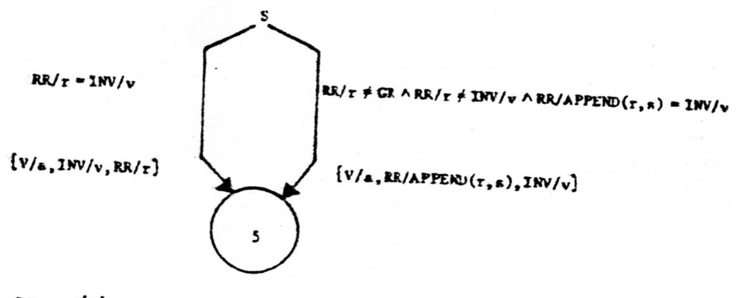
id(3) = 3 PC_3 = ({S}; ∅; {INV/v, RR/r}; {RR, INV}; ∅)

id(4) = 3 PC_4 = ({3}; RR ≠ INV ∧ RR ≠ GR; {RR/APPEND(RR, s)}; {RR}; ∅)

id(6) = 5 PC_6 = ({3}; RR = INV; {V/s}; {V}; RR = INV)

FIGURE 2f

Elimination of Node 7

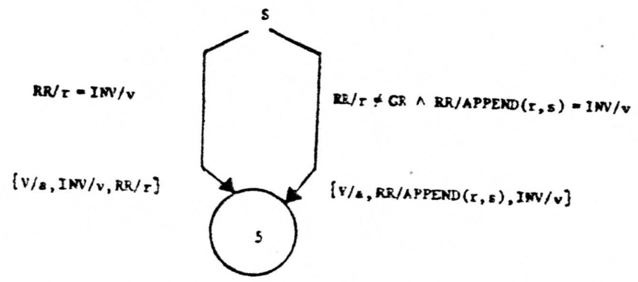

S
RR/r = INV/v
RR/r ≠ GR ∧ RR/r ≠ INV/v ∧ RR/APPEND(r,s) = INV/v
{V/a, INV/v, RR/r}
{V/a, RR/APPEND(r,s), INV/v}
5

$id(6) = 5 \quad PC_6 = (\{S\};\ RR/r = INV/v;\ \{V/a,\ INV/v,\ RR/r\};\ \{V,\ RR,\ INV\};\ RR = INV)$

$id(9) = 5 \quad PC_9 = (\{S\};\ RR/r \neq INV/v \wedge RR/r \neq GR \wedge RR/APPEND(r,\ s) = INV/v;$
$\{V/a, RR/APPEND(r,s), INV/v\};\ \{V,\ RR,\ INV\};\ RR = INV)$

FIGURE 2g

Elimination of Loop $\widehat{33}$ and Node 3


S
RR/r = INV/v
RR/r ≠ GR ∧ RR/APPEND(r,s) = INV/v
{V/a, INV/v, RR/r}
{V/a, RR/APPEND(r,s), INV/v}
5

$id(6) = 5 \quad PC_6 = (\{S\};\ RR/r = INV/v;\ \{V/a,\ INV/v,\ RR/r\};\ \{V,\ RR,\ INV\};\ RR = INV)$

$id(9) = 5 \quad PC_9 = (\{S\};\ RR/r \neq GR \wedge RR/APPEND(r,\ s) = INV/v;$
$\{V/a, RR/APPEND(r,s), INV/v\};\ \{V,\ RR,\ INV\};\ RR = INV)$

FIGURE 2h

Simplification Using $A \vee (\sim A \wedge B) \equiv A \vee B$

9. CONCLUSION

We believe that OIS research possesses great potential in its impact on the design and operation of business. Current research makes it quite evident that OIS modeling and evaluation is essential for assuring the successful introduction of OIS. TICOM-II contributes to these developments by equipping the office analyst with a controls-oriented tool for internal control evaluation of OISs. The rising increase of computer-assisted fraud emphasizes the need for implementing strong controls in the OIS. Though the verification of an internal control system is an NP-hard problem, TICOM-II harnesses the power of the computer to analyze internal control issues that lend themselves to mechanical analysis.

ACKNOWLEDGEMENTS

The authors are indebted to Dr. Clarence Ellis of Xerox PARC for his support of the concept that auditing and internal control should play a key role in automated office design.

The research was supported in part by a grant from the Peat, Marwick, Mitchell Foundation through its research opportunities in auditing program; however, the views expressed are those of the authors and do not necessarily reflect those of the sponsoring organization.

REFERENCES

1. A.D. Bailey, Jr., James Gerlach, R. P. McAfee, and A. B. Whinston, "Office Automation," Handbook of Industrial Engineering, Gavriel Salvendy ed. (John Wiley & Sons, Inc., New York, forthcoming).

2. ________, "Internal Accounting Controls in the Office of the Future," IEEE Computer, May 1981, pp. 59-70.

3. ________, "TICOM-II - The Internal Control Language," Florida Symposium Internal Controls, ed. by A. Rashad Abdel-Khalik, forthcoming.

4. ________, "An Introduction to the Theoretic and Analytic Capabilities of TICOM-II," Proceeding of the Second International Workshop on Office Information Systems, IRIA, Rocquencourt, France, forthcoming.

5. A. D. Bailey, Jr., R. P. McAfee and A. B. Whinston, "Application of Complexity Theory to the Analysis of Internal Control Systems," Auditing: A Journal of Practice and Theory, Vol. 1, No. 1, Summer, 1981, pp. 38-52.

6. G. Barber and C. Hewitt, "Research in Work-station Network Semantics," Working Paper MIT, Cambridge, Mass., 1980.

7. Deloitte, Haskins & Sells, "Internal Accounting Control," An Overview of the SH & S Study and Evaluation Techniques, New York, NY, 1979.

8. R. A. DeMillo, R. J. Lipton, A. J. Perlis, "Social Processes and Proofs of Theorems and Programs," CACM 22, 271-280.

9. E. W. Dijkstra, "Guarded Commands, Non-determinancy and Formal Description of Programs," CACM 18, 453-457, August 1975.

10. C. A. Ellis, "Information Control Nets: A Mathematical Model of Office Information Flow," ACM Proc. Conf. Simulation, Modeling and Measurement of Computer Systems, August 1979, pp. 225-240.

11. H. B. Enderton, A Mathematical Introduction to Logic, Acedemic Press, New York, 1973.

12. M. D. Zisman, "Representation, Specification and Automation of Office Procedures," Ph.D. Dissertation, Wharton School, University of Pennsylvania, Philadelphia, 1977.