1986

KASS: A Knowledge-based Auditor Support System

Ramayya Krishnan

Rayman D. Meservy
Brigham Young University, rayman_Meservy@byu.edu

See next page for additional authors

Follow this and additional works at: https://scholarsarchive.byu.edu/facpub

Part of the Accounting Commons, and the Management Information Systems Commons

BYU ScholarsArchive Citation
Krishnan, Ramayya; Meservy, Rayman D.; and Gadh, Vandana, "KASS: A Knowledge-based Auditor Support System" (1986). All Faculty Publications. 3237.
https://scholarsarchive.byu.edu/facpub/3237

This Peer-Reviewed Article is brought to you for free and open access by BYU ScholarsArchive. It has been accepted for inclusion in All Faculty Publications by an authorized administrator of BYU ScholarsArchive. For more information, please contact scholarsarchive@byu.edu, ellen_amatangelo@byu.edu.
KASS: A Knowledge-based Auditor Support System

Ramayya Krishnan
Rayman Meservy
Vandana Gadh

Decision Systems Research Institute
School of Urban and Public Affairs
Carnegie-Mellon University

ABSTRACT

This paper describes the design of a knowledge-based system to assist auditors in the evaluation of internal accounting controls and focuses on the logic-based language AL that has been developed as a knowledge representation formalism. Interesting features of AL include a declarative approach to modeling accounting systems and the means to explicitly describe authority structures typically used to enforce internal controls.
KASS: A Knowledge-Based Language for Auditor Support

1.0 Introduction

Internal controls evaluation of an accounting system is a complex task that has received growing interest in recent years. Past approaches that met with some success include those drawn from the Artificial Intelligence (AI) literature. Examples include TICOM, developed by Bailey et al [1985] and EDP-XPERT developed by Hansen and Messier [1986].

Seminal work in the application of AI to the domain of financial diagnosis is reported in Bouwman [1986]. Bouwman analyzed protocols of an expert and constructed models of a firm with heuristic rules that simulated expert behavior. This research laid the foundation for related research on Expert Systems. Braun and Chandler [1982] developed an expert system to assist auditors in the investigation of Analytical Review Fluctuations while Dungan and Chandler [1980] developed knowledge-based expert systems to model Auditors' decision processes. Bailey et al. [1985] developed TICOM, a system for modeling accounting systems with their embedded external controls, utilizing AI-based representation and search strategies. In modeling accounting systems, TICOM limited itself to modeling the equivalent of a flow chart representation of the system. No explicit means of modeling authority structures or rules to evaluate controls were provided. This paper describes an approach which combines the modeling power of flow charts, albeit within a declarative framework, with the means to model authority structures.
which are organization tools to enforce controls. In addition, our approach allows the specification of axioms to draw a variety of interesting inferences.

The rest of the paper is organized as follows. We begin with a brief description of Internal accounting controls and their role in accounting systems. The design of the KASS system is then discussed and the knowledge representation language AL detailed. Illustrative examples are used to describe the representational power of AL. The paper concludes with a detailed description of the inferences that may be generated from an AL model.

2.0 Accounting systems

Accounting systems, like most organizational systems, are extremely complex. This complexity derives from the variety of interacting procedures, objects and roles in the system. Roles are organizational entities (e.g. such as manager and clerk) which are filled by individuals charged with performing a series of procedures. These procedures typically act on objects such as documents, goods etc. and account for the functionality of accounting systems.

A brief description of the procedures, roles and objects that make up the accounting structure motivates the discussion of internal accounting controls to follow. Accounting systems span organizational units such as purchasing, receiving, stores etc. Roles in each of these units perform specific procedures. For
instance, a clerk in purchasing may be required to receive invoices and receiving reports (types of documents), match their "items" field and in the event of a match transfer the invoice to the cash-disbursements unit. In examining this fragment of a procedure closely it is apparent that:

a) the procedure consists of actions such as receive, match, transfer etc.

b) actions operate on documents such invoice, receiving report etc.

The operations on documents by actions involve other objects in the system such as repositories. Examples of repositories include files, inventory etc.. For instance, an action such as get or put relates documents and the repositories they are placed in. Finally, just as roles and the procedures they perform are attached to organizational units such as departments, so are certain kinds of repositories and documents.

While the discussion thus far, has been limited to physical actions such as transfer and get, there are certain other actions which we refer to as deontic actions. These are actions such as permit and prohibit which reference physical actions and access to assets to ensure that they are executed only in accordance with managements' general and specific authorization.

This variety of interaction between types of actions, roles,
objects etc. define an accounting system whose complexity derives from both the number and structure of these interactions.

3.0 Internal Accounting Controls

Internal accounting controls refer to all policies and procedures embedded within the structure of an accounting system that reduce unintentional exposure to business, financial, and accounting risks [Mair, Woods and Davis, 1980]. Controls are categorized in many different ways, such as preventive, detective or corrective. Preventive controls include policies and procedures that are designed to deter employees from making unintentional errors or committing irregularities. Detective controls, on the other hand, are primarily used to discover the occurrence of errors or irregularities [Loebbecpe and Zuber]. These then may go through a corrective procedures. Most controls are preventive because the cost of installing preventive controls is less than the cost of correcting irregularities discovered by detective controls at a later stage. Segregation of duties, accuracy controls and authorizations are classified as preventive controls. Such controls are implicit in the description of an accounting system. Auditors must identify these controls and evaluate them, determining the appropriate reliance to place on them.

The study and evaluation of internal accounting controls involves the expertise of well-trained auditors and is a requirement of each and every audit performed by CPAs. In order to be able to understand the complex accounting structure in its entirety and in
order to be able to evaluate the internal controls model existing in an organization, there is an urgent need for a tool that can model the system as well as evaluate it. (Note a call for such systems by Loebekke at the Price Waterhouse Audit Symposium, August 1988.) The focus of this paper is the development of such a knowledge-based system that will aid auditors in understanding complex accounting systems and will support the evaluation of internal accounting controls.

4.0 KASS: The Proposed System
As previously mentioned, the proposed system extends TICOM's modeling capabilities with deontic reasoning, utilizing a declarative framework. Additionally, the system incorporates rules and axioms used to evaluate internal controls such as described in Meservy et. al. (1986).

By declarative, we mean a framework that focusses on the key relationships between the important objects in the problem versus a specification which procedurally specifies how a problem is to be solved. We propose to do this in a well understood and sound formalism based first-order logic. The interesting features of the KASS system include the ability to model authorization structures on top of conceptual flow chart representations of accounting systems. This represents an important step from the knowledge representation viewpoint since it allows the explicit representation of authority which is the instrument used by organizations to effect control. Thus important control concepts
such as the segregation of duties can be understood and represented in terms of permissions and prohibitions placed on actions that might collectively compromise the controls of an accounting system. The knowledge representation language AL forms the focus of the rest of the paper.

4.0 Conceptualization of Accounting Systems
Modeling entails the construction of an artifact (the model) of some real-world problem. To control complexity, models abstract relevant aspects of the real-world problem of interest. This is what we refer to as a conceptualization of the problem being modeled. The explicit understanding of this conceptualization is particularly important in the use of logic-based modeling languages for knowledge representation since the semantics (loosely, the meaning) of a logic model is explicated in terms of relationships between the symbols of the model and the individuals that make up the "world being modeled".

Accounting systems in our modeling effort are conceptualized as systems that consist of a variety of objects (individuals). Examples of these objects include documents, repositories, roles, departments, and assets. Particularly interesting in our conceptualization is the representation of actions as objects. Actions are of two types: physical and deontic. Physical actions such as transfer, get, put etc. typically cause the flow of documents while deontic actions such as permit and prohibit determine who can perform any given action. The description of the
accounting system consists of a series of relationships defined on these various types of objects.

Traditionally, accounting systems are described from an action processing perspective. Actions which operate on specific kinds of objects and cause them to flow from one role to another, or from one organization unit to another. Actions are performed by agents who fill roles. The performance of physical actions are in turn controlled by deontic actions. Relationships between objects in our conceptualization are thus used to describe authority structures, actions, and the static functionality and capabilities of individual roles and organization units.

5.0 AL: The Language

The vocabulary of AL, like any other first-order language, consists of individual constants, predicate constants, function symbols, variables and logical constants. These symbols, combined using a set of formation rules, are used to model the individual objects in our conceptualization and describe the relationships they enter into.

While a core vocabulary is sufficient to describe the generic characteristics of accounting systems, the language allows the users (auditors) to supply vocabulary intended to model the specific accounting system under consideration. To differentiate between them, we refer to the core vocabulary as the closed vocabulary, and the vocabulary to be supplied by the user as the
open vocabulary. The vocabulary and the grammar of AL are described below.

5.1 Closed Vocabulary of AL

**Individual Constants:** roles, objects, repositories, departments, assets, []

**n-place predicates:** isa, ins-of, can-process, does-job, held-in, located-at, action, job-composition

**Function symbols:** transfer, assign, put, receive, review, destroy, get, copy, match, nomatch, approve, reject, available, unavailable, permit, prohibit, .

5.2 Open Vocabulary of AL

**Individual Constants:** denumerably many individual constants

**Individual Variables:** denumerably many variables

By denumerable, we simply indicate that the user may introduce as many constants and variables as he needs to describe the accounting system under consideration.

5.3 Syntactic Rules of AL

Before we present the grammar of AL, we need to define a term in the language. A term in AL is either an individual constant, an individual variable or a functional expression. If t1,...,tn are terms, then [t1,t2...,tn] is also a term.

In describing the grammar of AL, we use greek characters such as \( \alpha, \beta, \psi, \phi \) as part of our meta language to represent constants while \( \mu \) is used to designate variables.

Grammar of AL
PM has a grammar very similar to fairly standard versions of the first-order predicate calculus with equality.

A well formed formula (wff) of AL is defined recursively as follows.

1. If \( \Phi \) is a predicate of \( n \) places, \( (n \geq 0) \) and \( \alpha_1, \ldots, \alpha_n \) are terms, then \( \Phi(\alpha_1, \ldots, \alpha_n) \) is a wff.

2. If \( \Phi \) is a wff, then so is \( \neg \Phi \)

3. If \( \Phi \) is a wff and \( \psi \) is a wff, then so are \( \Phi \) and \( \psi \); \( \Phi \) or \( \psi \)

4. If \( \Phi \) is a wff and \( \beta \) is a wff then if \( \Phi \) then \( \beta \) is also a wff.

5. If \( \Phi \) is a wff and \( \beta \) is a wff then \( \Phi \) if and only if \( \beta \) is also a wff.

6. \( \forall \mu \, \Phi \) is a wff.

7. \( \exists \mu \, \Phi \) is a wff

6.0 Examples in AL

The representational power of a language is best illustrated using examples. AL is expressive enough to capture all the information in the conventional flow chart approach and in addition offers the means to describe authority structures which determine the flow of permission and access that are fundamental to the controls of an accounting system. These features of AL are described using a fragment of a purchasing system described in Bailey et al (1985).
Objects in an accounting system can be classified by type. The isa predicate serves to relate objects of similar type. Thus, the sentence isa(clerk,roles) asserts that clerk is a type of role. Similarly, documents are types of object. Note that classification can go down several levels as in isa(po,document) which classifies purchase-orders (po) as documents.

```plaintext
isa(clerk,roles)
isa(goods,object)
isa(documents,object)
isa(po,documents)
isa(rr,documents)
```

Instances of objects are described using the ins-of predicate. For instance, rrl is an instance of a receiving report. Similarly john is an instance of a clerk. The ins-of and isa predicates allow the description of classification hierarchies. The choice of the level of aggregation at which objects in the system are described is left entirely up to the auditor.

```plaintext
ins-of(rrl,rr)
ins-of(john,clerk)
```

Various types of documents are stored or placed in types of repositories. Checks are held in safes, purchase-orders in on-order files etc. This is described using the held-in predicate.

```plaintext
held-in(checks,safe)
```

The roles in an organization are typically associated with one or more procedures. The list notation using the square brackets ([[]]) is used to model a set of procedures associated with a role. The predicate does-job is used to relate roles to lists of procedures.

```plaintext
does-job(clerk,[receiving-procedure])
```

Procedures are made up of transactions which in AL are modeled as actions. Specifically actions such as transfer, send etc. are modeled in AL using functions. The receipt of goods at the receiving department is modeled using the function receive(goods, receiving). All actions are distinguished using the predicate action which also associates each action with an unique name.

```plaintext
action(al,receive(goods,receiving))
```

The example describes an action named AL (an arbitrary choice)
which denotes the action receive(goods,inventory). Each transaction in the accounting system is modeled using the action predicate and the functions such as transfer, get etc. Some examples are shown below.

\[
\begin{align*}
\text{action(a2,assign(rr1))} \\
\text{action(a3,copy(rr1,rr2))} \\
\text{action(a4,transfer(rr1,purchasing))} \\
\text{action(a5,transfer(rr2,stores))} \\
\text{action(a6,put(goods,inventory))}
\end{align*}
\]

The procedures that roles are required to perform consist of a set of actions. The predicate \text{job-composition} is used to relate each procedure to the list of actions it is comprised of. For instance, the receiving procedure is described as shown below.

\[
\text{job-composition(receiving-procedure,[a1,a2,a3,a4,a5,a6])}
\]

The data flow in an accounting system is typically accomplished in flow charts using a set of links between transactions. In AL actions are related using precedent relationships. For instance, the receiving report may be assigned only after the goods have been received. Thus actions follow one another. Precedent relationships are described using the predicate \text{follows}.

\[
\begin{align*}
\text{follows(a2,a1)} \\
\text{follows(a3,a2)} \\
\text{follows(a4,a3)} \\
\text{follows(a5,a4)} \\
\text{follows(a6,a5)}
\end{align*}
\]

Flow charts also model data control which represent decision points based on some condition. Such data controls need to be modeled explicitly only when the dynamic functionality of accounting systems is of interest. Since we are interested only in static functionality, we model alternatives available at such data control points as distinct actions. Thus, matching a typical example of a data control point is modeled using actions such as match and nomatch which account for alternatives that might be conditioned on a satisfactory or unsatisfactory match.

A very important feature of accounting systems are controls. Controls typically are concerned about access, be it to objects, transactions or repositories. Organizations enforce controls using a system of authorization. AL offers the means to explicitly describe these authorizations using the functions permit and prohibit which are special types of actions referred to as deontic actions. The examples describes the act of permitting a clerk to put goods in the inventory and of authorizing a manager to approve a requisition.

\[
\text{action(a7,permit(clerk,put(goods,inventory))}
\]
action(a8,permit(manager,approve(requisition)))

These authorization structures can be generalized to model the handing down of permission across several organization layers and are used to analyze the existence of controls in the AL model of an accounting system.

The next section describes axioms in AL that are used to generate inferences.

7.0 Inferences in AL

The previous section described how accounting systems may be represented in AL. Axioms are used to derive inferences from this AL representation.

Transactions which are part of various procedures in an accounting system need to be related. In terms of the flow chart representation, this implies the need to link the symbols used in the flow chart. Since transactions are represented as actions in AL, this implies the need to infer precedent relationships between actions that are part of distinct procedures in the system. We assume that precedent relationships between actions within a procedure are supplied by the auditor describing the accounting system. Axioms that relate actions establish precedent relationships between them. Consider a simple example.

if action(A,transfer(X,L)) and action(B,receive(X,L)) then follows(B,A)

The axiom relates the transfer and receive actions in an intuitive manner. Given actions such as

action(a12,transfer(requisition,purchasing))
action(a13,receive(requisition,purchasing))
that are part of job descriptions of the store clerk and the purchase clerk respectively, the axiom results in follows(a13,a12)
Which states that action a13 follows action a12. The application of similar axioms which relate other physical actions result in the definition of an action graph. The nodes of this graph are actions and the arcs represent the precedence relationships modeled in the follows predicate. Figure 2 illustrates an action graph.

Auditors often evaluate the controls in an accounting system by studying important transactions and their precedents. This enables the auditor to understand the sequence of actions that need to be performed prior to a critical action. Additionally, alternative sequences of actions may exist. The action graph in AL provides a convenient representation to study sequences of actions which correspond to paths in the graph. Further these paths may analyzed to identify critical actions where controls need to be enforced. The problem of identifying critical transactions among a set of precedent transactions has been approached by Bailey et. al. (1985) as analogical to determining the weakest precondition among a set of preconditions. We adopt the approach of analyzing multiple paths in the action graph using rules drawn from past experience to identify critical actions. The path from a root node (a node with no ancestors; typically an action that represents an exogenous action to the accounting system such as the transfer of invoices by a vendor) to an action of interest to the auditor can be
inferred using the recursive relation path shown below. The action nodes in each list that satisfies the path relation are analyzed using rules shown below.

******* Ray, please add rules to evaluate a sequence of actions to determine a critical action such as match **********

\[
\text{path}(X,X,[]) \\
\text{path}(X,G,[X.R]) \text{ if } \text{follow}(X,P) \text{ and } \text{path}(P,G,R)
\]

Controls are closely related to the authorization structures in an organization that are used to enforce access to objects. As noted earlier, since physical actions are the means employed to access objects, they are controlled using deontic actions such as permit and prohibit. Axioms are used to deduce the actions in the system that may be performed by agents in the organization.

Since agents fill roles, roles are either explicitly authorized to perform an action that is part of its job description or permitted to do so through the flow of authorization up or down the hierarchy. Examples of permissions to perform actions associated with a job description are as shown below.

\[
\text{if does-job}(\text{Role},[J_1, \ldots, J_n]) \text{ and } \\
\quad \text{job-composition}(J_1,[A_{11}, \ldots, A_{1m}]) \text{ and } .. \\
\quad \text{job-composition}(J_n,[A_{n1}, \ldots, A_{nm}]) \text{ and } \\
\quad \text{action}(A_{11},F_{11}) \text{ and } .. \text{ and } \text{action}(A_{nm},F_{mn}) \\
\text{then action}(B_{11},\text{permit}(\text{Role},F_{11})) \text{ and } .. \text{ and } \\
\quad \text{action}(B_{mn},\text{permit}(\text{Role},F_{mn}))
\]

The axiom simply associates explicit permissions with all actions that are part of job descriptions associated with a role. For
instance, in the example the clerk would be permitted to perform all actions that make up the procedure.

Permissions may also flow down the organizational hierarchy. For example a manager may authorize the clerk to perform certain actions. The general form of such an axiom is shown below. Note that by combining the permit function, authorization hierarchies may be modeled to arbitrary depths.

\[
\text{if } \text{supervise}(X,Y) \text{ and } \text{supervise}(Y,Z) \text{ and action}(A_1, \text{permit}(X, \text{permit}(Y, \text{permit}(Z,F)))) \text{ then action}(B_1, \text{permit}(Z,F))
\]

Similarly permissions may flow up the hierarchy. If a manager supervises the clerk, then the manager is authorized to perform any action that the clerk could have.

\[
\text{if } \text{supervise}(X,Y) \text{ and action}(A_1, \text{permit}(Y,F)) \text{ then action}(C_1, \text{permit}(X,F))
\]

All these alternative means of obtaining authorization need to be controlled since the vesting of authorization to perform certain actions to a particular role or individual may result in the controls being compromised. A common example involves actions such as assigning vouchers and performing bank reconciliation. If the same person performed these actions, controls may be potentially compromised. Axioms are used to detect lapses in controls caused by the assignment of access to conflicting actions. These are particularly useful since they keep track of all the alternative ways authorization may be obtained. An example of such an axiom is
shown below.

\[
\text{if } \text{action}(A, \text{permit}(\text{Role}, F_1)) \text{ and } \text{action}(B, \text{permit}(\text{Role}, F_2)) \text{ and }
\text{segregate}(\text{Role}, A, F_1, B, F_2)
\text{ then breakdown}(\text{Role}, F_1, F_2)
\]

\[
\text{segregate}(\text{Role}, A, F_1, B, F_2) \text{ if and only if }
((\text{if } \text{action}(A, \text{permit}(\text{Role}, F_1)) \text{ then } \text{action}(B, \text{prohibit}(\text{Role}, F_2))) \text{ or }
((\text{if } \text{action}(B, \text{permit}(\text{Role}, F_2)) \text{ then } \text{action}(A, \text{prohibit}(\text{Role}, F_1)))
\]

Actions that need to be segregated are axiomatically related using 
permit and prohibit actions.

In addition to these inferences, queries may be posed to the AL 
model to deduce information such as the identity of roles that 
perform critical operations etc. Thus AL provides a means for 
describing an accounting system, axiomatizing its behavior and 
facilitates understanding of the system and its controls using 
deductive inference.

Conclusion and Future Additional Research


7. Loebbecke, and Zuber,