Artificial Intelligence
Representing Knowledge in AIMDS

N. S. Sridharan


1. Logic, Artificial Intelligence and Psychology

The purpose of this paper is to provide a tutorial on artificial intelligence: especially on methods of developing a knowledge base. However I have a second aim that colors the manner in which I conduct this tutorial, namely, to introduce to you a high level computer language, AIMDS [2] [3], that I have been developing over the past five years. As I introduce this language I shall emphasize the facilities for the representation of knowledge.

AIMDS has been used in the development of TAXMAN II system [4]. This paper also serves as a companion paper to [4], using the examples from TAXMAN II, but attempting to give the reader a general idea of the techniques of knowledge representation.

The language that I'm going to talk about is based on logic and set theory, but it is a programming language. It will be somewhat different from computer programming languages that you might be acquainted with. It arises out of an interesting reconciliation between the power of logic and the limits of a computer!

AIMDS is guided in its development by the influence of a psychologist, Professor Charles Schmidt, with whom I am collaborating on developing a psychological theory of recognizing intentions and understanding of certain.

N. S. Sridharan is Associate Professor in the Department of Computer Science at Rutgers University, New Brunswick (New Jersey, USA).

[Editor's Note] This article reproduces the paper presented by the Author at the International Conference on «Logica Informatica, Diritto» (Florence, April 1981) organized by the Istituto per la documentazione giuridica. The final draft of this paper, having arrived too late to be included in the Proceedings of the Conference at present being published by North Holland (Amsterdam), is printed here because of the importance to our readers of the themes discussed.
actions. In developing such a theory, modelled as a computer program, we have attempted to deal specifically with limitations of memory and limitations of processing power that humans have. But people do exhibit extraordinary intelligence. They can organize their information, focus their attention on what is relevant and structure their processes to overcome their own limitations to meet the demands of a task. This raises the prime challenge for the field of Artificial Intelligence — how do we make the computer do the same.

Thus logic attempts to set the ideals in rational thinking. Models of thinking developed in Artificial Intelligence take into account the need to conserve computing resources. Cognitive models in Artificial Intelligence go further in attempting to display processing characteristics which are similar to those of humans.

2. Statement of Purpose

The structure of my talk would be to give you a quick overview of artificial intelligence; and come to focus on the idea of knowledge as being the essence of how our intelligence operates, and then to go directly into the language AIMDS. I shall draw examples and illustrations for this presentation from the TAXMAN application, so that you will see some representation of legal concepts and facts. I am keenly aware of the diversity of this audience, therefore, I am trying to keep my presentation at a very simple level, and I hope to communicate as much as I can. My purpose will be served if I arouse enough interest in you that you will follow up the newly developing field of Artificial Intelligence.

In the ideas underlying the language there are some solutions, at least some suggestions about solutions, to some of the problems that are being discussed at this conference. For example, we heard discussions about the distinction drawn between «descriptions» and «prescriptions», between «propositions» and «practices». In AIMDS we also make the distinction between simulating the effects of an action and the description of an action (whether past or contemplated, suggested, accepted or rejected). In further discussions it would be fruitful to examine the nature of the distinctions being drawn and to relate them to the analyses of the problems presented at this conference. Similarly, in AIMDS there is an object level with descriptions of some domain of discourse (for TAXMAN this is the world of corporate structure and transactions); and there is a meta-level that describes the conceptual organization of the object level. It's quite impossible to go into details of these issues which the conference participants have brought to the forefront this morning.
There are 4 basic terms that I would like to introduce to you at a survey of artificial intelligence. The first idea is that of a symbol. The beginnings of artificial intelligence are rooted in the realization that the computer is primarily a symbol processor not just a number processor. Symbolic information can be such things as names, text material, rules, definitions and so on. A symbol standing alone in isolation is not of much value. So we bring in the next idea which is structure — symbol structure. Symbols are combined to form symbol structures with specified connections among its parts. Aggregation is the most familiar method of building symbol structures. Take a common example — in an address label, the name, the street address, the city, the region and the country all appear together. That is aggregation of different pieces of different symbols to make a symbol structure. Association is another form of bringing the symbol structure together which is a very useful concept in organizing a computer memory. The use of a pointer or an address is a way of achieving this in a computer. At the end of your paper you include citations; that is association. If we take a set of papers from these conference proceedings and follow the citations we see the pattern of association. These are the two main ideas in how we construct symbol structures with symbols.

Symbol structures can be transformed computationally into other symbol structures. Now a set of symbol structures and their transformations constitutes a space. One important space for artificial intelligence is the search space. A problem is formulated to a computer when you define the search space, and you define the criteria by which a solution must be judged. Very early work in artificial intelligence counts success in various forms of game playing and proving theorems in the propositional calculus. Search spaces for game playing, for example Chess, can be large. That is an understatement; they can be enormously large. Years ago, it was fashionable for various popular writers to describe how the fastest of the conceivable computers, if they could operate a thousand times faster than they can, would take longer than the putative age and lifetime of the universe to search such spaces exhaustively. So, exhaustive search through a search space is clearly infeasible as the core idea of artificial intelligence. Thus enters the idea of doing heuristic search. Heuristics are used in two ways. Firstly, at the outset, generate a subspace for search, that is generate only a small fraction of the search space, not all of it. This subspace is a plausible subspace; solutions are more likely found there. Secondly, use the idea of pruning the search space. After you consider a candidate you can eliminate a fragment

1. Newell and Simon [6] in their Turing Award lecture advance the notion that symbol structures in physical symbol systems capable of Designation and Interpretation are necessary and sufficient for the general simulation of intelligence. We concur with this view. We extend this, claiming association and aggregation methods make it convenient for the general simulation of intelligence.
of search space and therefore can confine the search to even a smaller fraction. Programs such as the Geometry Theorem Prover and DENDRAL are excellent examples of heuristic search that exploit both ideas. Ineffective search of a space is a mark of ignorance. I adopt a pragmatic view of knowledge. Knowledge is characterized as that which leads to effective reasoning and effective action. Can knowledge be exhibited only as effective behavior? Recent work in Artificial Intelligence attempts to be systematic in describing such knowledge in the form of symbol structures. Knowledge Representation consists in the systematic attempts to provide symbol structures and transformations for encoding knowledge, and forms a primary focus of current work in Artificial Intelligence. Knowledge acquisition and validation forms another focus.

Table 3-1: What is Knowledge?

Knowledge is that which leads to effective reasoning and effective action.

The basic method of reasoning involves search; knowledge leads to abbreviated search.

All structures we deal with are, at present, discrete and symbolic.

4. Knowledge, Its Types and Levels

What kinds of knowledge are there? We formulate four kinds of knowledge: Knowledge of individuals, knowledge of classes, "how to" knowledge and meta-knowledge.

Knowledge about individuals: We have to represent knowledge about individuals. We have to talk about their attributes and properties, and we have to describe relationships among individuals. Knowledge about a given collection of individuals and their relationships constitutes knowledge about a state of affairs (minimally).

Knowledge about classes: Individuals belong to classes. We have to represent knowledge about classes. Such knowledge includes properties of classes, relations to other classes, relationships between classes and individuals.

Knowledge of procedures: Recognition of membership of an individual in a class, comparison of individuals for similarity and differences, existence of

2. The Handbook of Artificial Intelligence [1]. edited by Avron Barr and Edward Feigenbaum, contains a very representative set of articles covering not only fundamental concepts, theoretical results, but also applications of Artificial Intelligence and performance results. This book is highly recommended to anyone interested in learning more about AI.
an individual with specified combination of attributes, modeling of state changes are some examples of procedures that utilize knowledge. Knowledge about these procedures include descriptions of what the procedures accomplish and how the procedures operate.

Meta-knowledge: The above may be considered knowledge at the object level. At the meta-level we talk about the scope and limits of what is known about instances, about classes and about procedures. Furthermore at the meta-level we can describe the relevance and the importance of what we know. We can also talk about sources of information – if you don’t know something, how to go about finding out what you want to know. What I hope to do in going through the formalism of AIMDS is to give you samples of each of these kinds of knowledge and how they are represented.

<table>
<thead>
<tr>
<th>Instances</th>
<th>their properties</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>their relation to other instances</td>
</tr>
<tr>
<td>Classes</td>
<td>their properties</td>
</tr>
<tr>
<td></td>
<td>relation to other classes</td>
</tr>
<tr>
<td></td>
<td>relation to instances</td>
</tr>
<tr>
<td>Procedures &amp; Actions</td>
<td>what they do and how they do it</td>
</tr>
<tr>
<td></td>
<td>what are the components of procedures, and how are they related</td>
</tr>
<tr>
<td>Meta-knowledge</td>
<td>scope and limits of what is known</td>
</tr>
<tr>
<td></td>
<td>relevance and importance of what is known</td>
</tr>
<tr>
<td></td>
<td>what can be known and how</td>
</tr>
</tbody>
</table>

### Table 4-1: What kind of knowledge is there?

#### 4.1. Individuals and Situations

Here is an application from TAXMAN, a simple situation consisting of a collection of instances, their properties and relationships. In simple English this situation may be described as «Phellis owns 100 shares of the common stock of the New Jersey corporation». The reader is now advised to follow the discussion by repeatedly referring to Figure 4-1. The figure depicts the New Jersey corporation as NJ, and indicates that NJ is an instance of the class CORPORATION which is a subclass of ACTOR. Phellis is depicted as PHELLIS which is an instance of the class PERSON which is a also a subclass of ACTOR. The common stock issued by NJ is depicted as
STOCK-1, which is an instance of STOCK, SECURITY and INTEREST. STOCK-1 is described as being COMMON stock, with (INSHARES) 2000 shares outstanding. SHARE-1 is a share of STOCK-1, and also an instance of SHARE and PROPERTY. The QUANTITY of SHARE-1 is 100. The relationship between Phellis and the shares owned by Phellis is described by means of an OWNership record. The relationship between SHARE-1 and PHELLIS is described by means of the OWN-1, which is an instance of the class OWN, a class of ownerships. The OWNER of OWN-1 is

"Phellis owns 100 shares of the common stock of the New Jersey corporation."

*Figure 4-1. — Symbolic representation of the above network*
PHELLIS, while the object owned in OWN-1 is SHARE-1. Similarly, the relationship between the stock and New Jersey corporation is described by means of the act of issuing stock. The act ISSUE has an instance ISSUE-1, whose AGENT is NJ and the object issued is STOCK-1.

(ISSUE-1 NAME-INSTANCEOF-TEMPLATE ISSUE)
(OWN-1 OWN-OWNED-PROPERTY SHARE-1)
(OVERN-1 OWN-OWNER-PROPERTY PHELLIS)
(SHARE-1 NAME-INSTANCEOF-TEMPLATE SHARE)
(SHARE-1 NAME-INSTANCEOF-TEMPLATE PROPERTY)
(SHARE-1 SHARE-QUANTITY-PROPERTY 100)
(SHARE-1 SHARE-OFP-INTEREST STOCK-1)
(STOCK-1 NAME-INSTANCEOF-TEMPLATE STOCK)
(STOCK-1 NAME-INSTANCEOF-TEMPLATE SECURITY)
(STOCK-1 NAME-INSTANCEOF-TEMPLATE INTEREST)
(STOCK-1 STOCK-COMMON-YES NO YES)
(STOCK-1 STOCK-NSHARES-PROPERTY NUMBER 2000)
(NJ NAME-INSTANCEOF-TEMPLATE CORPORATION)
(NJ NAME-INSTANCEOF-TEMPLATE ACTOR)
(PHELLIS NAME-INSTANCEOF-TEMPLATE PERSON)
(PHELLIS NAME-INSTANCEOF-TEMPLATE ACTOR)

The above list is a form of internal representation of the situation being described. We have captured knowledge about individuals such as NJ, STOCK-1, SHARE-1, OWN-1, PHELLIS in terms of their properties and their relationships. The list below shows some of the general knowledge about classes. For example, that CASH is a subclass of PROPERTY, that SHARE and CASH are disjoint classes, and so on.

(CASH TEMPLATE-AKO-TEMPLATE PROPERTY)
(SHARE TEMPLATE-DISJOINT-TEMPLATE CASH)
(CASH TEMPLATE-DISJOINT-TEMPLATE SHARE)
(PERSON TEMPLATE-DISJOINT-TEMPLATE CORPORATION)
(CORPORATION TEMPLATE-DISJOINT-TEMPLATE PERSON)
(SHARE TEMPLATE-AKO-TEMPLATE PROPERTY)
(STOCK TEMPLATE-AKO-TEMPLATE SECURITY)
(SEcurity TEMPLATE-AKO-TEMPLATE INTEREST)
(PERSON TEMPLATE-AKO-TEMPLATE ACTOR)
(CORPORATION TEMPLATE-AKO-TEMPLATE ACTOR)

We have illustrated above that using instances, we can describe individuals, classes, abstract relationships, and actions. Among the facts listed above, some are statements that are transient and temporary, whereas some others.
may be viewed as having a longer persistence. AIMDS does not make this distinction. However, it provides a facility for maintaining historical description of changing situations, and for consideration of possible evolution of situations in the future. Historical trace of changing situations is very convenient for undertaking analysis of past events; planning actions is made easier by considering alternative courses of actions. Appropriately, AIMDS provides STATES which is a set of situations linked into a binary tree; each link designates a line of succession. Each STATE contains a situation, and facts not expressly modified from its predecessor states will remain in this state. This makes it convenient to describe a successor state only in terms of the predecessor plus the changes from the predecessor to the successor.

\[\text{CASH} \rightarrow \text{PROPERTY} \quad \text{PERSON} \rightarrow \text{ACTOR}\]

\[\text{STOCK} \rightarrow \text{SECURITY} \rightarrow \text{INTEREST}\]

*Figure 4-2. — Class Inclusion and Exclusion*

Let us consider state S1 to be the situation referred to previously. Let S2 be a state that follows S1, i.e. (S1 STATE-PRECEDES-STATE S2). In the State S2, we modify the owner of OWN-1 and make (OWN-1 OWN-OWNER-PERSON MCDUFFY). After this change, in the State S2, MCDUFFY is the OWNER of OWN-1. After this change, in the State S1, PHELLIS is still the OWNER of OWN-1. In the State S2, all other facts true in S1 will also be true. For example, OWN-1 will have OBJECT to be SHARE-1 which will still have QUANTITY 100.

There is yet another concept, CONTEXT, which is a collective name for the set of states that are linked together by means of successor links. There may be several contexts, each of which can be used to store a separate description of a legal case. Thus the current case, the precedents and hypothetical cases can all be stored in separate contexts, without confusion among the facts and states of one case with the facts and states of another case. The use of CONTEXTS and STATES is one way in which the system AIMDS goes beyond the ordinary semantics of the pure predicate logic.
4.2. Classes, Templates and Subclass descriptions

Now let's talk a little bit about classes of objects. When we talk about a class we have to think about extension as well as intension.

<table>
<thead>
<tr>
<th>Table 4-2: Intension and Extension</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extension: of a class is represented as a collection of instances which are members of the class</td>
</tr>
<tr>
<td>Intension: of a class is represented as Templates, Consistency Conditions, Characterizing description Meta-relations among templates</td>
</tr>
</tbody>
</table>

Extension consists of listing the members of the class, intension consists of getting a characteristic description of members of that class. We have shown several class names earlier along with the relations of class exclusion and inclusion, namely STOCK, SECURITY, SHARE, CASH, PROPERTY, INTEREST, PERSON, CORPORATION, ACTOR. Whereas these class relations and the instance relation will allow us to build set theory, we develop the language so that the intensions of classes get emphasized.

What is of interest in describing corporations is not only that some named corporations are members of that class, but that there is a coherent set of attributes and relations which corporations participate in. For example, corporations are issuers of stock, can be agents of various corporate actions and so on. To tell AIMDS about the set of attributes and relations that may be used in building up descriptions of instances we introduce the notion of a template that is associated with the class. Here is an example of a template for describing members of a class.

Every member of the stock class can be described in terms of the dividend rate, whether or not it is a voting stock, and whether it is a common stock or a preferred stock. And we also mention that if it is not a common stock it would have to be preferred stock. There is an expression that says a complement of common stock is preferred stock. Another example, is the record of ownership class; ownership is declared to be a relation, by using REL. The owner of an ownership is an ACTOR; the actor would be the OWNROF the ownership record, indicated by the INVERSE relation. The fact that an ownership record can have one OWNER only is indicated by the marking that OWNER is a function (FN). Similarly, what is OWNED is a property.
Table 4-3: Templates

Templates serve to

(a) provide a vocabulary from which the language descriptions of class members can be constructed;

(b) denote algebraic properties of the relations defined with that class as the domain/range;

(c) introduce information that serves to control the actions that need to be taken when adding/removing/updating descriptions of instances.

(TDN: (STOCK
  ((DIVIDEND-RATE FN NUMBER)
   (VOTING YESNO)
   (COMMON YESNO (COMPLEMENT PREFERRED)))))

(TDN: (OWN REL)
  ((OWNER FN) ACTOR (INVERSE OWNEROF))
  ((OWNED FN) PROPERTY))

4.3. Descriptions built from the vocabulary

This allows us to describe an individual instance of STOCK or of OWN in the following manner.

(STOCK STOCK-1
  (VOTING YES)
  (COMMON YES)
  (DIVIDEND-RATE 5))

(OWN OWN-1
  (OWNER PHILLIS)
  (OWNED SHARE-1))

Table 4-4: Descriptions of instances

We can describe a class or a subclass using a very similar syntax, extended by the introduction of variables.

3. The function (c) of Templates will not receive attention in this report, since this function is concerned with the pragmatics of system usage rather than of knowledge representation, the topic of this paper.
Here P, O, SH, S are all variables. The description characterizes the classes of persons who are shareholders of some common and voting stock. Person P is an owner in an ownership record O. The ownership record indicates that what P owns is a share SH of stock S. Stock S is a common and voting stock. So every person who satisfies that description is such a shareholder.

4.4. Intensional subclasses and subsets

We could compare two descriptions of classes and determine that one describes a subclass of the other. For example,

(PERSON P (OWNEROF (OWN O (SHARE SH))))

is a subclass of

(ACTOR A (OWNEROF (OWN O (PROPERTY P))))).

The former is specialized to owners who are Persons, and who own Shares. The latter describes owners who are Actors, and who own any Property. Similarly, the description

(PERSON P (OWNEROF (OWN O (SHARE SH (OF (STOCK S (COMMON YES)))))))

characterizes a further subclass of common-stock holders. The system AIMDS has a built-in definition of a relation more specific than which determines intensional inclusion among descriptions. The extension of one description, D1, will be a subset of the extension of another description, D2, if D1 is more specific than D2. In this system, extensional set inclusion relation is kept distinct from intensional class inclusion.

A set of templates given to the system defines an infinite set of nested descriptions, with the associated partial ordering relation, more specific than, creating a space of descriptions. This systematic space of description is very central to the Taxman II research on the representation of concepts by Prototypes-and-Deformations, the development of arguments, and the evolution of concept definitions.
4.5. Algebraic properties of relations

Relations introduced in a template can be indicated to have a combination of algebraic properties. Each property is given by means of a flag associated with the relation in the template definition. The list of available flags is FN, L, IRR, REF, ASYM, SYM.

<table>
<thead>
<tr>
<th>FN</th>
<th>Functional</th>
<th>The given relation is a function.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Thus the relation is one to one.</td>
</tr>
<tr>
<td>L</td>
<td>List</td>
<td>The given relation is not a function.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Thus the relation is one to many.</td>
</tr>
<tr>
<td>IRR</td>
<td>Irreflexive</td>
<td>The relation is irreflexive,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>i.e. ( \sim (X \ R \ X) ) for all ( X ).</td>
</tr>
<tr>
<td>REF</td>
<td>Reflexive</td>
<td>The relation is reflexive,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>i.e. ( (X \ R \ X) ) is true for all ( X ).</td>
</tr>
<tr>
<td>ASYM</td>
<td>Asymmetric</td>
<td>The relation is asymmetric,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>i.e. ( (X \ R \ Y) ) implies ( \sim (Y \ R \ X) ) for distinct ( X, Y ).</td>
</tr>
<tr>
<td>SYM</td>
<td>Symmetric</td>
<td>The relation is symmetric,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>i.e. ( (X \ R \ Y) ) iff ( (Y \ R \ X) ) for distinct ( X, Y ).</td>
</tr>
</tbody>
</table>

Since the flags are pairwise disjoint, any relation can be given 18 combinations, choosing FN or L, IRR or REF or none, ASYM or SYM or none. Some of the combinations are of particular interest. Consider a relation having the combination (FN REF). Such a relation is an identity function, \( (X \ R \ X) \) is true for each \( X \) and \( (X \ R \ Y) \) is false if \( X \) is not equal to \( Y \). The flags in combination with the meta-relations given below provide fairly expressive means of describing properties of relations.

4.6. Meta-relations

We may describe Inverses, Complements and other relations among relations. Thus, we have a set of meta-relations.

5. Instantiation and Retrieval

We introduce named individuals into the current state by using the command verb, MAKE. MAKE accepts a description of an individual instance and adds facts about the individual to the current situation. For example, to say that Jones is an individual who owns 13000 units of cash, we may say:

\[(\text{MAKE} \ (\text{OWN} \ \text{OWN}-3 \ (\text{OWNER} \ \text{JONES}) \ (\text{OWNED} \ \text{CASH}-7)))\]
\[(\text{MAKE} \ (\text{CASH} \ \text{CASH}-7 \ (\text{QUANTITY} \ 13000)))\]
Here, OWN-3 and CASH-7 are arbitrary names introduced to refer to the instances. JONES is inferred from the context to be an ACTOR. The facts added are:

(OWN-3 OWN-OWNER-ACTOR JONES)
(OWN OWN-OWNED-PROPERTY CASH-7)
(CASH-7 CASH.-QUANTITY-NUMBER 13000).

Retrieval of created instances and added facts is accomplished through the command verbs FIND or SEEK. FIND is designed to retrieve a set of instances that satisfy a characteristic description. The result is the empty set, NIL, if no such instance can be found.

\[
\text{\langle FIND (CASH (QUANTITY 13000)) \rangle} \implies \text{(CASH-7)}
\]
\[
\text{\langle FIND (THE ACTOR (OWNEROF OWN-3)) \rangle} \implies \text{JONES}
\]
\[
\text{\langle FIND (ALL OWN (OWNED CASH) (OWNER JONES)) \rangle} \implies \text{(OWN-3)}
\]

It is also useful to have a facility to retrieve not only instances but their related properties, attributes and other instances at once. Such a multiple retrieval facility is SEEK. SEEK accepts a description containing variables and returns bindings for variables 4.

\[
\text{\langle FINDSEEK-DL-N ' ((CASH C (QUANTITY X))) \rangle} \implies \text{((C CASH-7) (X 13000))}
\]
\[
\text{\langle FINDSEEK-DL-N ' ((OWN O (OWNER A))}
\quad \text{(OWNED (CASH C (QUANTITY 13000)))) \rangle}
\quad \implies \text{((O OWN-3) (C CASH-7) (A JONES))}
\]

4. What we call bindings are called substitution in Logic.
6. **Action Knowledge**

Knowledge about actions is given in terms of state changes they induce, as well as by describing compound actions in terms of other actions. The figure below indicates the use of aggregation and abstraction in building up action hierarchies.

![Diagram](image)

*Figure 6-1. — Aggregation and Abstraction of Actions*

6.1. **State Changes**

A state is described by pairing a description of the state before the change and the state after the change. A state change could take place between two successive states or two states separated by other states. Therefore, we allow each description to indicate a state variable and to include a statement of the relationship between the two variables.

\[
\text{(PDN: CHANGE-OWNER)} \quad \text{((STATE T1 (OWN D1 (OWNER A1) (OWNED P1))))} \\
\text{((STATE T2 (OWN D1 (OWNER A2) (OWNED P1)))}} \\
\text{((T1 STATE-PRECEDES-STATE T2))} \\
\text{NIL)}
\]

The above example indicates a state change from state T1 to T2, where T1 precedes T2, and involves a change in owner of an ownership record.\footnote{Such a pairing of descriptions can also be used in Taxman II to represent Permissions and Obligations. The details of this are omitted here.}

214
have named this pairing of state descriptions PDN (Production Definition) following common usage in A1, where a «production» consist of a «condition» and «actions». When a state change is attempted using a PDN, the description of the state before the change serves as a «condition», i.e. the system verifies that the current state satisfies the description. In the example above, if A1 and P1 already have assigned values, it will be verified that the ownership record exists. If, say A1, is unassigned, the verification will assign values for it. The state will then be modified to satisfy the description given for the state after the change.

6.2. Simple Action

Simple actions will have their effects described by means of a state change PDN. For example, by abstraction, we can introduce a DELTAOWN action whose effects are given by the PDN shown in the previous subsection.

```
(DELTAOWN D
 (AGENT (ACTOR A))
 (OBJECT (PROPERTY P))
 (OLDOWNER (ACTOR A1))
 (NEWOWNER (ACTOR A2))
 (TIME1 (STATE T1))
 (TIME2 (STATE T2)))

HAS EXPANSION

(PDN: CHANGE-OWNER
  ((STATE T1 (OWN O1 (OWNER A1) (OWNED P1)))
   (STATE T2 (OWN O1 (OWNER A2) (OWNED P1)))
   ((T1 STATE-PRECEDES-STATE T2))
   NIL))
```

As another illustration, consider change in the amount of ownership, the owner remaining the same. The amount owned increases by N, from old amount Q1 before the change to amount Q2 after the change.
(DELTAQUANT+ D
  (OBJECT (PROPERTY P))
  (OWNER (ACTOR A))
  (AMOUNT (NUMBER N)))

HAS EXPANSION

(PDN: DELTAQUANT+PDN
  ((STATE T1
    (OWN O1
      (OWNER A)
      (OWNED P)
      (QUANTITY Q1)))
   (STATE T2
    (OWN O2
      (OWNER A)
      (OWNED P)
      (QUANTITY Q2)))
   ((EQUAL Q2 (=PLUS N Q1))
    (T1 STATE-PRECEDES-STATE T2))
   NIL))

(PDN: DELTAQUANT+PDNO
  ((STATE T1 (ACTOR A) (PROPERTY P))
   (STATE T2
     (ACTOR A)
     (PROPERTY P)
     (OWN O1
      (OWNER A)
      (OWNED P)
      (QUANTITY N)))
   ((T1 STATE-PRECEDES-STATE T2))
   NIL))

This action consists of two expansions, one for the case there is a prior ownership record property P to the amount Q1, and another for the case there is no ownership record, i.e. the owner A is considered to have 0 amount of the property P. In the latter case, a new ownership record is created in the amount of N, after verifying only that A is an actor and that P is indeed a property. In the former case, the outcome asserted is ownership in the amount of Q2, with the relation stated that Q2 = Q1 + N.

6.3. Compound Actions, Action Hierarchies

We may also define one action by describing it in terms of a composition of other actions. The two simplest methods of action composition are exemplified below, however, there are a variety of such methods possible.
The first method, parallel composition, involves co-action, wherein two more actions take place between the same two states T1 and T2 and a new action is defined to be equivalent to them. We define TRANS:SPLIT to transfer an amount N of a property P from owner A1 to A2. The co-actions are DELTAQUANT+ which adds the amount N to A2, and DELTAQUANT− which reduces the ownership of A1 by the amount N. These two sub-actions are co-actions in the sense that both of these are defined to have their state changes between the same two states.

\[(\text{TRANS:SPLIT} T)\]
\[
\begin{align*}
&\text{(AGENT (ACTOR A0))} \\
&\text{(OBJECT (PROPERTY P))} \\
&\text{(OLDOWNER (ACTOR A1))} \\
&\text{(NEWOWNER (ACTOR A2))} \\
&\text{(AMOUNT (NUMBER N))} \\
&\text{(TIME1 (STATE T1))} \\
&\text{(TIME2 (STATE T2))} \\
\end{align*}
\]

\[\text{HAS EXPANSION}\]

\[(\text{DELTAQUANT+ D1})\]
\[
\begin{align*}
&\text{(OBJECT P)} \\
&\text{(AMOUNT N)} \\
&\text{(OWNER A2)} \\
&\text{(TIME1 T1)} \\
&\text{(TIME2 T2)} \\
\end{align*}
\]

\[(\text{DELTAQUANT− D2})\]
\[
\begin{align*}
&\text{(OBJECT P)} \\
&\text{(AMOUNT N)} \\
&\text{(OWNER A1)} \\
&\text{(TIME1 T1)} \\
&\text{(TIME2 T2)} \\
\end{align*}
\]

For sequential composition, we use actions whose state variables form a chain of precedence relation. An Exchange can be defined to be a pair of Transfers of this sort.

Actions, like that of Distribution of stock dividends to stockholders, are compositions of individual Exchanges with individual stockholders, aggregated over the class of stockholders to whom the distribution is being made.

6.4. Action Hierarchies

Using the mechanisms outlined above for action definition, we have been able to introduce action hierarchies, an example of which is presented be-
low. Change of owner in an ownership is defined as DELTAOWN, which when mediated by an agent gets abstracted to the action TRANS (transfer). An EXCHANGE involves mutual transfer of ownership of two objects between owners. An ACQUISITION is an exchange in which one of the owners is a Corporation. The acquisition by one corporation, in exchange solely for its voting stock, of stock of another corporation is a Type B tax-free reorganization (United States Internal Revenue Code, 368 (a) (1) (B)), providing the acquiring corporation has CONTROL over the acquired corporation immediately thereafter. CONTROL is defined as the ownership of at least 80% of the voting stock of the controlled corporation.

Using these hierarchical definitions, we have created RMATCH and GMAKE procedures for use in Taxman II. For details, the reader is referred to [5]. Given the set of time-ordered states that represents the facts of a case and the effects of the transactions in the case, RMATCH can be used to determine if some compound action such as BREORGANIZATION can be recognized among the states. RMATCH will return a residue describing the best or closest match \(^6\) obtainable via some set of definition expansions. It also returns a metaresidue which describes the set of definitions which were tried and annotation of the nature of success or failure following each of those alternatives. RMATCH is, thus, like FIND, but utilizes the definition hierarchies and therefore returns a more complex result.

---

6. This obviously rests on a system defined criterion on what is meant by best or closest. We have attempted to refrain from injecting criteria that reflect any knowledge of the corporate tax domain.
GMAKE is like MAKE, but relies on the results of doing an RMATCH. GMAKE attempts to produce the necessary state changes to simulate the effects of a compound action. The closest match produced by RMATCH identifies a small set of changes that would provide the necessary state changes. GMAKE subsequently effects the changes, producing new or modified states as appropriate. After GMAKE produces its changes, a subsequent RMATCH will surely succeed in finding a match!

7. SUMMARY OF REPRESENTATION FACILITIES

Our view of knowledge representation covers the behavioural manifestations of having and using knowledge as well the systematic expression of knowledge as symbol structures. Knowledge is not being viewed as something static, but one possessed with dynamic qualities. A model of states is introduced to compartmentalize knowledge and also to describe its dynamics. Association and Aggregation are seen as ways of building up complex symbol structures. We built a definition hierarchy of actions to illustrate this. The processes that we run on symbol structures are developed from the abilities of: Access, which is following associations; Matching, which is the ability to decompose and test components of aggregates; Copy/Transform, which is the ability to replicate symbol structures and to transform them systematically. Meta-knowledge, in this scheme of things is knowledge

<table>
<thead>
<tr>
<th>Two basic structural elements of Representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASSOCIATION</td>
</tr>
<tr>
<td>AGGREGATION</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Three basic procedural elements of Representation,</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACCESS</td>
</tr>
<tr>
<td>MATCHING</td>
</tr>
<tr>
<td>COPY/TRANSFORM</td>
</tr>
</tbody>
</table>

Meta-knowledge is knowledge

about Association and Aggregation

about Access, Matching and Transforms

Table 7-1: Basic Representation Tools
about association and knowledge about aggregation; knowledge about how the definitions are structured, knowledge about what definitions there are, how they can be used, knowledge about how definitions can be changed, how definitions can be constructed.

There are facilities for representing classes and for representing instances. There is a notion of the state and we can model time which can be linear time, a branching time. We can transform one state to another. Actions are built up from state changes by association and aggregation yielding a definition hierarchy. Although I have not illustrated for you, it is possible to model notions of expectation, intention [7] obligation and permission.

<table>
<thead>
<tr>
<th>Table 7-2: AIMDS facilities summary</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CLASSES</strong></td>
</tr>
<tr>
<td>their extension</td>
</tr>
<tr>
<td>their properties</td>
</tr>
<tr>
<td>their relation to other classes</td>
</tr>
<tr>
<td>their characterizing description</td>
</tr>
<tr>
<td>their structural expansions, procedural expansions</td>
</tr>
<tr>
<td><strong>INSTANCES</strong></td>
</tr>
<tr>
<td>their properties</td>
</tr>
<tr>
<td>their relation to other instances</td>
</tr>
<tr>
<td>their relation to classes</td>
</tr>
<tr>
<td>their parts</td>
</tr>
<tr>
<td><strong>STATES</strong></td>
</tr>
<tr>
<td>aggregation of assertions about instances</td>
</tr>
<tr>
<td>relationship to other states</td>
</tr>
<tr>
<td>inheritance from parent states</td>
</tr>
<tr>
<td><strong>ACTION</strong></td>
</tr>
<tr>
<td>state change</td>
</tr>
<tr>
<td>action with agents</td>
</tr>
<tr>
<td>action and their purposes, preconditions, outcomes</td>
</tr>
<tr>
<td>action information about expectation, desirability and so on</td>
</tr>
<tr>
<td><strong>DEFINITION</strong></td>
</tr>
<tr>
<td>Association</td>
</tr>
<tr>
<td>between Abstractions (descriptions)</td>
</tr>
<tr>
<td>and Expansions (aggregations)</td>
</tr>
</tbody>
</table>

Acknowledgement. Research on Taxman II is supported financially by National Science Foundation Grant# NSF-MCS79-21471. The design and construction of AIMDS was made possible by a Grant from the Division of

REFERENCES


