LEARNING BY PATTERN INDUCTION

by

Stephen Fraser Zeigler

Computer Sciences Technical Report #319

April 1978

Learning by Pattern Induction

Stephen Fraser Zeigler

Computer Science Department
University of Wisconsin - Madison
April 14, 1978

Abstract

Human infants are able to accumulate considerable knowledge in their first year of life without having well-developed communications skills. Pattern induction is proposed as a mechanism for accomplishing learning in a precommunication environment. This mechanism acts by detecting and describing regularities in memories of past experiences. Recognizer-predictors, structures having some similarities to productions, are proposed to represent and utilize information gathered by pattern induction. In this paper, recognizer-predictor structures and the process by which pattern induction formulates them are described in the context of MUL, an existing TELOS program modelling infant-like development in a very simple environment.

Introduction

This paper describes continuing research on a computer model of some aspects of early intellectual development in humans, focusing particularly on pattern induction, a learning mechanism proposed as underlying that development. "Early intellectual development is taken to be that occurring in infancy, or more precisely the sensorimotor period of Piaget.

Piaget characterizes four periods in the intellectual development of humans [Piaget 1954, Piaget & Inhelder 1969]. The first, and primary subject of this research, is called the sensorimotor period. Lasting until the infant reaches about two years of age, this period is marked by a slow struggle to grasp fundamental environmental properties. Among the more important of these is "object permanence": the infant must learn to recognize that objects will create certain reproducible effects on his sensory input streams. In general, the infant must learn elementary coordination of motor and sensory activities.

Piaget's observations led him to propose three later periods: the pre-operational period (lasting until about age 6), the concrete-operational period (lasting in part for the remainder of life), and the formal-operational period (beginning, with luck, at adolescence). During the pre-operational period the child begins to think about events which are not actually occurring. The child is not capable of sustaining any coordinated or purposeful thinking until the concrete-operational period. During this third

period, the child develops the skills needed for effective understanding of specific (concrete) events. The formal-operational period is characterized by the appearance of capabilities for hypothesis, systematic experimentation, and formalization (abstraction).

From the perspective of the infant, the problem faced immediately after birth is to make sense of a confusing mass of input and output. Sensory input arrives, continually establishing new conditions at various interfaces with the brain. Motor output is produced when certain conditions are detected at the output interfaces. By hypothesis, these interfaces are to higher level constructs than individual muscles or sense organs. Units of motor output might be actions like grasping and reaching. Sensory input might consist in part of outputs from line and motion detectors.

Certainly an external teacher cannot be relied upon at this stage: The infant can't isolate a teacher as an object. the infant must discover from experience that the conditions it observes at these interfaces are not random. Regularities are present. The learning system proposed here, called MUL, acts to detect and describe these regularities, and then to use the descriptions first to recognize events, then to predict unobserved events, and perhaps eventually to hypothesize scenarios for possible event sequences. The name "MUL" is an acronym for Method Underlying Learning and coincidentally describes that method: "mulling" things over.

MUL

In the next section, a version of MUL is described. The algorithms presented are not intended as a final word on MUL, but rather as a sort of existence proof. The design and implementation are continually evolving, even in fundamental ways. MUL does exist as a TELOS program, and it is largely a result of the support that language offers that such design evolution is easily carried out. Appendix 2 presents a brief description of TELOS, as used in the MUL program.

The model for learning systems proposed by Smith, et al. [Smith 1977] provides a framework in which to discuss the current MUL. To begin, learned knowledge is represented by creating symbolic structures rather than by adjusting parameters. The MUL learning element does not learn in direct response to input: it strives to discover and describe regularities among episodic memories (fading records of MUL's physical and mental experiences). The MUL program thus learns by discovery, selecting its own subjects as well as examples and non-examples for them.

Since MUL is not expected to respond to input in any conventional sense, there are no "correct responses" available for performance evaluation. The level of performance is instead measured by estimating the extent to which the environment is understood. Understanding is operationally defined, for the purposes of this research, as the ability to anticipate future events.

The following diagram illustrates the organization of MUL, following the Smith model except in details of information flow and blackboard structure. The next sections of this paper will describe the operation of each functional unit. See Appendix 1 for a brief characterization of MUL in terms of the Smith model.

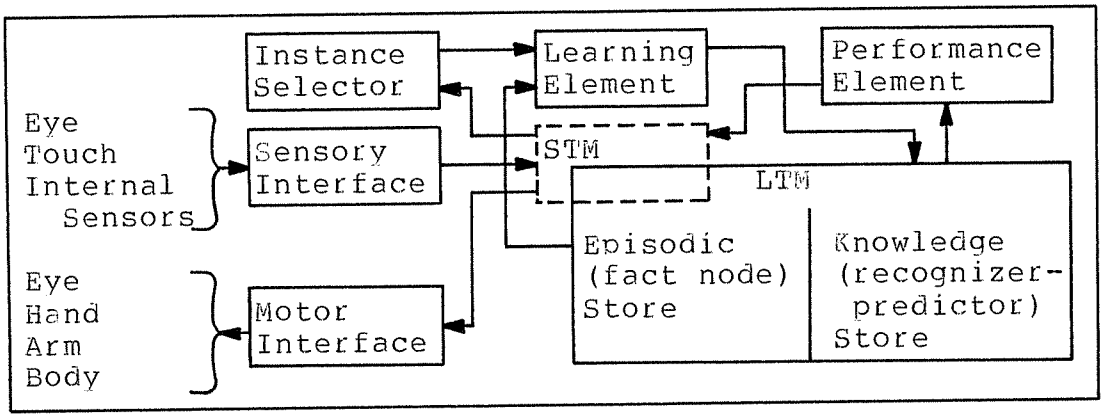

Figure 1. MUL organization

Memory & Performance

Memory for the MUL system is composed of two sections: the episodic (or fact) store and the knowledge store. The fact store is composed of objects called nodes, and serves to record the experiences of MUL, both physical and mental. Several types of nodes will exist for any particular environment: each node type is predefined with a fixed organization of fields, each with a fixed interpretation. New node types are not introduced during the operation of MUL. A node, of any type, is always fully instantiated at the time of its creation, in the sense that each information field must have a definite value, which remains constant

throughout the life of the node. Nodes also have one or more auxiliary fields, distinguished from information fields and not used by the pattern matching process. For instance, the CRITIC procedure maintains an auxiliary field for an estimate of each node's importance. Auxiliary fields may change in value as the importance of the node is altered.

The knowledge store medium is the recognizer/predictor structure (RP for short). An RP is a nonempty set of patterns for fact nodes. Associated with each component pattern is a statistic called the "prediction energy" of that component. RPs are created by the learning element discussed in a later section. The RPs and their component patterns are limited in complexity by the sophistication of that element.

An RP recognizes a group of fact nodes when each component pattern of the RP matches some node in the group. In MUL, successful RP application results in the creation of an RP-success fact node "naming" the recognized node group or at least announcing that the node group has certain properties.

RPs are not applied to all nodes of episodic memory, but only to fact nodes in a short term memory (STM). New nodes have a chance to enter this STM as they enter episodic memory. However, since the STM is of fixed size, it may already be full of nodes. This conflict is resolved by relative node importance: the least important node is forced to relinquish its claim to an STM position, without undergoing RP analysis. Nodes are selected one by one from STM, under-

going an examination in which RP component patterns matching the node are found and "activated". That is, satisfied patterns are flagged with information concerning the date and circumstances of the new match. The RP containing a newly activated pattern as a component is examined. If all its component patterns have been matched within a suitably short time, then the RP "succeeds", and an appropriate RP-success fact node is generated to enter in its turn into episodic memory. The suitably short time mentioned above is represented by a constant called "ACTIVATION_LIFE" in the current MUL. RP-success nodes serve a dual purpose: they allow MUL to examine the progress of its own thought process and, because that examination might result in the creation of new RPs recognizing regularities in RP-success node occurrence, can be used to build hierarchies of recognizers. A "table" RP might recognize occurrences of "flat-surface" RP-successes, "leg" RP-successes, and so forth.

RP use may result in fact node prediction by the following mechanism. Whenever an RP is found to have recent (within ACTIVATION_LIFE) matches for most (currently, all but one) of its component patterns, that RP creates a "pressure" to find matches for its remaining unmatched component(s). This pressure is a function of the importance of the RP, the importance(s) of the nodes used in the matches, and the prediction energy associated with the unmatched component(s). The prediction energy is a measure of the validity of past predictions of that component. When the pressure is great enough MUL will take steps to find or

manufacture (predict) a matching node. This process serves several purposes. It enables MUL to anticipate node occurrence. It can be used to fill in missing information, as with "defaults" for frames [Winston 1975]. Finally, it is used to direct motor behavior: prediction of a motor activity fact node has the side effect of requesting that the appropriate activity be performed.

The process begins with the application of the unmatched pattern to episodic memory. If a sufficiently recent node matches then that node is used to satisfy the pattern (the node must have seemed too unimportant to process when it entered memory). If no recent node is found, a new fact node is manufactured (using the most recent node matched as a model) and entered into episodic memory as though it had just occurred normally. This manufacture process becomes complicated only when an RP-success node is to be manufactured. In this case, MUL tries to support the new node by finding or manufacturing nodes to match the components of the predicted RP. That is, each component of the predicted RP is itself predicted. Prediction (particularly of RP-success nodes) is thus a top-down process as opposed to the bottom-up process represented by normal RP application.

The RP structures share some features with the productions of production systems (cf [Davis 1975]). Most important, new RPs can be learned and introduced with as little difficulty as (pure) productions. RPs have a structure similar to productions, although the distinction be-

tween whether a component is in the antecedent or the consequent rests solely on the size of its prediction energy: nodes having non-zero prediction energies may act as consequents during the node manufacture process. Node manufacture is similar to the evaluation of a triggered production's consequent not only in that new nodes are introduced into memory, but also in that side effects may occur. For RPs, the only side effects now possible are motor activities.

Some differences from standard production systems are important, however:

-RPs are matched in parallel: many RPs may exist in a partially matched state at any given time.

-The order in which component patterns are matched is not important, except in creating pressures for component satisfaction.

-The nodes matched by components of an RP need not be present in the STM, or even in episodic memory, when the RP succeeds. The only requirement is that all matches must have occurred within ACTIVATION_LIFE time units.

-RPs are not required to have components with non-zero prediction energies, nor are they restricted from having several. RPs may in this way have an arbitrary number of "consequents".

-an RP produces an RP-success node when all its expected patterns are satisfied.

-memory nodes do not have an associated truth value, only an importance. This is not an oversight, but a result of a

hypothesis that the concepts of truth and falsehood are learned, arising later than the sensorimotor period.

-the use of RPs does not involve "backing up" or "unwinding", as no specific goal is present. The concept of "working towards a goal" is also assumed to be learned, although desires may be intrinsic. Desires are not currently represented in MUL, but the node manufacture process should support the development of goal-directed behavior.

Critic:

The CRITIC is responsible for resource allocation decisions. It is not a monolythic element but a set of algorithms distributed to points of use within other MUL program elements.

Many resource allocation questions are answered on the basis of the fact node importance values maintained by CRITIC algorithms. For example, importance is used in deciding which nodes can enter and remain in STM for RP activity processing, which nodes should be culled from a full episodic memory, and which nodes should be subjected to the MUL learning process. A simple evaluation of the importance of each node is done as the node is created, by a fixed mechanism depending on what process created the node. For nodes introduced by the sensory apparatus, importance is a constant moderate value, as it is for nodes introduced by physical activity. Physical activity node importance is higher. Nodes introduced by prediction have their importance defined as a function of their prediction energy.

RP-success nodes have an importance dependent on the importance of the nodes matched.

The operations of CRITIC algorithms on RPs in the knowledge store are also simple. RP pattern prediction energies are calculated from a count of the number of times the pattern was predicted and a count of how many of these predictions were validated. A prediction is validated when a fact node equivalent to the predicted node is encountered within ACTIVATION_LIFE time units after the prediction. Two nodes are equivalent iff the values in their non-auxiliary fields are equivalent.

The space available in the knowledge store is limited. For this reason RPs must be occasionally culled. Deciding which RP to abandon is another major task of CRITIC algorithms. The importance of maintaining an RP is a function of how often and how well the RP is performing its two tasks, recognition and prediction. An RP's predictive strength is the importance of the most important node it would predict if all its other patterns were matched by nodes of some standard importance. Recognizer strength is measured as the importance of the RP-success node generated given that all component patterns matched nodes of the standard importance.

Learning Element

The learning element is charged with the task of introducing new RPs. The creation of a new RP might be justified either because of trusted communication (ie, being told) or because of experience. Since MUL operates in the pre-communication sensorimotor period, learning must be based on experience. Each new RP, then, should act to recognize that a situation is similar to a previously experienced situation, and suggest information about unobserved features of the recognized situation.

MUL's process for discovering the regularities of experience justifying RP formation is called "pattern induction". To indicate how pattern induction might be accomplished, this section presents a general framework for a pattern induction mechanism. Its goal is to produce new RPs, by finding component patterns (here called P1,...,Pn and Px) such that when nodes matching these patterns are encountered, the RP-success node generated will be useful. As a simplification, only one pattern (Px) is allowed to develop a non-zero prediction energy.

Ø. Given a fact node X chosen by the instance selector:

[This algorithm will try to create an RP having as a component a pattern (Px) matching X. With luck, the algorithm can assign a non-zero prediction energy to Px, so that future occurrences of X can be foreseen.]

1. From the set {nodes N of episodic memory, where N was entered less than ACTIVATION_LIFE time units before X} choose MAX or fewer nodes, say {A1,...,Am}
   - ACTIVATION_LIFE is currently 1: patterns will remain activated after matching a new fact node for only one environment step. Thus, RPs can succeed only if all their patterns are matched within one environment step of one another.
   - The choice of the Ai is currently random, weighted by node importance. A more sophisticated scheme is being designed, with the aim of defining relatedness between nodes. For example, a retina cell might be related to its nearest neighbors and to its previous values. A node could then be introduced to the set {X,A1,...,Ai} only if it were related to some node already in the set. The effects and justifications of relatedness are similar to those for recognition cones [Uhr 1977].
   - MAX is currently 3: RPs contain at most four component patterns including Px.

   [If a new RP is formed, it will describe some or all nodes in {A1,...,Am,X}, and may possibly have a non-zero prediction energy for the pattern matching X.]
2. For each Ai construct a pattern Gi as a loose generalization of Ai.
   - Loose generalization is currently very loose indeed: the pattern requires only that the node type be the same as Ai's type. This will certainly be changed in

later MUL versions, and is now used only for simplicity.

[MUL will attempt to find other occasions when nodes similar to {A1,...,Am} occurred together. These occasions will be called node groups.]

3. By searching episodic memory, construct a sample S containing some groups of nodes recognized by the patterns Gi. A group is recognized by the Gi if: 1) if nodes n1 and n2 are in the group, then n1 and n2 were entered less than ACTIVATION_LIFE time units apart; 2) each Gi matches some node in the group; 3) each node in the group is matched by some Gi; and 4) the number of nodes in the group is not greater than the number of Gi's.

   [The groups in S represent other occurrences of nodes similar to the nodes in the group {A1,...,Am}. The steps below will attempt to describe the conditions (in terms of internal group properties) present when a node similar to X entered episodic memory just after the occurrence of a group. Ideally, the conditions can be described in terms of patterns, enabling the creation of a predicting RP.]

4. If the size of the sample does not exceed MIN_SAMPLE_SIZE or if the number of groups in S which lead to an X-like node does not exceed MIN_LEADING_TO X then a new group of Ai's is selected and processing resumes at step 2.

   - A group in S leads to a node if that node entered the episodic store not more than ACTIVATION_LIFE time

units after the youngest node in that group, but not before the oldest node in the group.

- MIN_SAMPLE_SIZE is currently 5: RPs are created only if at least 5 groups in S support them.
- MIN_LEADING_TO_X is now a function of the total number of X-equivalent nodes residing in the episodic store. An X-equivalent node is a node whose information fields (as opposed to auxiliary fields) are equivalent to those of X. This definition allows prediction to be based on even a single occurrence of a very important node.
- An X-like node is a node that matches a loose generalization of X.
- Each new group of Ai's is one node smaller than its predecessor, but formed by random choice. This decrease in size terminates looping. If size zero is reached then the learning element returns in failure.

5. By applying some pattern-narrowing method to the Gi, attempt to find patterns Pi such that if the Pi's recognize some group in S, then that group leads to an X-equivalent node. These Pi are precisely the patterns needed to form a predicting RP.
   - The current pattern-narrowing process begins with the Pi equivalent to the Gi. S is divided into S-yes (containing groups in S leading to X-equivalent nodes), and S-no (the remainder of S). While a member of S-no can be found to be recognized by the Pi, some change will be made in one of the Pi so that the

member of S-no will no longer match. The changed Pi may no longer recognize some groups in S-yes: These groups are dropped from S-yes. The change to the Pi is currently a minimal one, in the sense that no other change would result in fewer nodes being dropped from S-yes. If S-yes has fewer than MIN LEADING_TO X members remaining, then continue at step 6.

6. If no basis for predicting X is found, then an attempt is made to construct a non-predicting RP recognizing cooccurrences of X with important groups in S. At some later time a predicting RP may be learned to predict these cooccurrences by predicting occurrences of RP-success nodes for this RP. For example, it is difficult to successfully predict the occurrence of vertical poles on the basis of low-level local information, but if the concept of a table is known, and a table is expected in some location then table legs can be predicted.
   - In the current MUL, this recognizer creation step is straightforward. The group in S that lead to the X given by the instance selector is used as a model. The Pi are created to specifically recognize groups equivalent to this group, with Px being constructed as for a predicting RP except that its prediction energy is zero.

This particular algorithm, especially in detail, is not the last word on pattern induction implementation. Numerous variations await investigation, each promising their own

benefits. Some general comments can be made on more fundamental aspects of pattern induction, however.

The pattern induction mechanism is not a "reaction to input", but acts on recurrent aspects of experience. Of course, MIN_LEADING_TO X may be very small (even one) if X is of sufficient importance, so that "quick" learning is possible. The cautious approach of (normally) learning only in response to several similar experiences should result in more stable behavior.

Pattern induction acts to define similarity between experiences. That is, two nodes can be regarded as similar if they match the same pattern. Since patterns are associated with RPs, the definition of similarity which they represent is valid only for the purposes of the containing RP. This gradual and domain-specific development of similarity allows exact matches to be required for pattern matching during RP use. Although approximate matching is still done during RP construction, damage done by a bad match is limited to the introduction of a bad RP. Bad RPs are introduced in any case, so that damage from improper matches is corrected by the already required RP culling process.

Pattern induction also acts to define "relatedness" between nodes: an RP implies that nodes matched by its patterns are related to one another, especially if a prediction is involved. While pattern induction does not guarantee the relevance of these relationships, the action of the CRITIC should leave only consistently useful RPs, thus defining relatedness and relevance by experience.

Pattern induction is currently the only mechanism for learning used by MUL. This is not out of a desire for economy, minimality, or elegance, goals rejected eloquently by Minsky [Winston 1975], but rather because complexity obscures behavior. Other mechanisms or improvements will be introduced, but pattern induction and its interaction with the RPs it creates seem to offer unusually comprehensive capabilities. One potential improvement might be applying newly created RPs across episodic memory to more precisely determine their usability and prediction energy. Experience with earlier MUL versions has indicated that improvements are required in the pattern induction mechanism, particularly in the construction of its samples. The relatedness mechanism mentioned may effect that improvement.

Instance Selector

MUL has, as yet, no specific goals. Its level of performance is measured as a function of how well it anticipates its environment. Future development of "desires" will probably not affect this measure other than by emphasizing desire-related experiences. The function of the instance selector is to focus the learning element on nodes which were unpredicted but, because of their importance, should have been.

Environment

The choice of environment is not important to the study of MUL, as long as it is an environment within which learn-

ing from experience is possible. Unfortunately, MUL can not be switched to completely new environments without difficulty, as the nodes of episodic memory, and thus the patterns matching them, are environment-specific.

The current environment is a 3 x n array of colored, textured cells inspired by Becker's environment [Becker 1970]. MUL receives sensory input from a 3 x 3 cell retina (each cell specifies red, black, white and colorless), a touch sensor status (returning the texture in the environment cell occupied by the hand), a hand status (open or closed), and an arm status (raised, centered or lowered). A major departure from Becker is that all sensory information is received for each time interval. Possible motor activity events are a body move (left or right), an arm move (raise or lower), and a hand move (open or close).

The only movements in the environment are generated by MUL: body moves and arm moves. MUL also has the potential to "carry" the colored blocks to new cells in the environment. No change in MUL structure is anticipated in order to handle non-MUL controlled motion, or other environmental changes. The environment may be altered in the future, particularly after "desire" apparatus is added to MUL, so that more interesting learning can be studied. One of the impressive features of MUL is its adaptability to alterations in its environment.

This particular environment was chosen because it is simple enough that a sizable episodic memory for it could be maintained, because a number of simple extensions can in-

crease its complexity to more interesting levels, and because it has been used in previous research.

Final Comments

MUL exists as a TELOS program running on the UNIVAC 1100 at the University of Wisconsin - Madison. The existing version has succeeded in discovering simple relationships. For example, it has discovered that if it can see things, then its eye must be open. MUL's pattern induction is, however, prone to seek relationships between plainly unrelated events. The introduction of relatedness among fact nodes will alleviate this problem.

While MUL is somewhat primitive to be called a cognitive model, and has yet to be observed in realistic operation, some observations on its expected behavior can be made.

- MUL seems to face problems similar to those hypothesized to dominate the human sensorimotor period. In particular, the discovery of object permanence is considered a landmark goal for MUL.
- Fortunately, while psychologists must guess about an infant's thoughts from indirect cues such as eye movements and facial expressions, MUL "psychologists" can examine MUL's knowledge directly.
- As with an infant, no single "rule" will appear to describe a concept like object permanence. Rather, it seems likely that each object will be described by an essentially independent set of RPs . Object permanence will be

reflected as an overall property of the predictions made by these RPs.

- As with an infant, MUL is expected to learn slowly. An idea of the real-time resources needed for a productive MUL run can be estimated from an approximation of the number of database retrieval operations done in processing each environment step and the number of environment steps to be processed. By design the environment moves to a new state when MUL has done some constant number of data base accesses (currently 40 of the inverse pattern retrieval operations). The real time required for these operations rises with the log of the number of items in episodic memory and is thus bounded by memory capacity. Preliminary trials indicate that MUL processing should not exceed 5 seconds per environment step and on the average should be about two seconds. Since in the simpler environment there is less to be learned, MUL is expected to learn interesting things within 500 environment steps, or about 1,000 seconds of execution time. For comparison, infants may not exhibit signs of having learned for as long as 1,000,000 seconds (about 10 days).
- Pattern induction requires a large amount of memory. The needs of episodic memory dominate: given that fact nodes require 30 bytes of storage, and appear at the rate of 50 nodes per environment step, storage demands could approach a megabyte after the 500 steps planned.

These observations should make clear that no HAL 2000 will spring full-grown from the core of the UNIVAC. MUL is not primarily an attempt to construct an artificial intelligence. The aim of MUL research is to develop an existence proof that learning by pattern induction can solve problems analogous to those faced by infants during their sensorimotor period. The MUL design will be vindicated if its learning by detection and description of regularity can help elucidate human intellectual development.

Appendix 1

A characterization of the MUL system after the fashion of the model for learning systems proposed in [Smith 1977].

Purpose: To learn to understand an environment, given that understanding is measured by the accuracy of predictions of future or otherwise unobserved events.

Environment: A matrix, each element containing a color and a texture. Program recieves such input as retina cell values, touch sensor value, arm status, etc. Output of program controls arm, body, hand, etc. Noise-free, with no externally generated motion (at present).

Performance Element: Applies recognizer/predictor structures (provided by the learning element) to current "important" memory nodes, producing new memory nodes if the applications are successful either in recognition or in prediction.

Learning Element: Employs pattern induction to create new recognizer/predictor structures. Pattern induction is a learning paradigm: episodic memory is searched for regularities, particularly regularities that suggest that the occurrence of one class of memory node seems dependent on the previous occurrence of some other class(es) of memory node(s). A recognizer structure describes any regularity found. It will be given a predictive role if an antecedent/consequent relation can be established.

Critic: sets the importance of each memory node as it is created, and each recognizer/predictor when necessary. These importance values are used to decide space and time allocation in the other functional units.

Instance Selector: selects the memory node upon which the attentions of the learning element should be focussed. Its choice is a function of importance of each memory node and to what extent the memory node was predicted.

Appendix 2

The programming language TELOS is an extension of PASCAL designed specifically for the high-level applications common in artificial intelligence [Travis 1977, LeBlanc 1977]. TELOS incorporates facilities for modularization (by data and control abstraction), for psuedo-parallelism (by coroutines and generators), for inter- and intra- process communication (by an EVENT mechanism unifying messages and software interrupts), and for an associative, context-dependent data base. Only the data base related facilities have been implemented so far, since only those facilities were absolutely necessary for MUL construction.

The TELOS data base is composed of data base objects (DBOs). DBOs are dynamically created by user calls to the system STORE function. The call STORE(<ptr>) will cause the creation of a new DBO having as its value a transcription (into the data base) of the object referenced

by <ptr>. The only constraint on <ptr> is that it must be a pointer-valued expression: only objects referenced by pointers may be transcribed into the data base. STORE returns a reference called a DBOP (for DBO pointer) to the new DBO. User programs may use DBOPs as they would pointers, to obtain access to DBO values. DBO values may also be accessed by associative lookup, as will be explained later.

The following annotated excerpt from the declaration of the fact_node type should give the flavor of episodic memory contents (as well as a glimpse of the TELOS database facility). Numbers in parentheses () refer to explanatory notes found below.

```
fact_node = PACKED RECORD (1)
   importance : INTEGER SEQUENCER; (2)
   birth_date : age_type INDEXED; (3)
   CASE class : memory_node_classes INDEXED OF
     retina_event:
       ( pos : retina_positions;   color : colors ); (4)
     rp_success_event:
       ( rp : DB-> rp_node; (5)
         nodes_matched : -> nodes_matched_listcell ); (6)
             .
             .
             .
END; { of fact_node }
```

(1) TELOS records have the same structure as PASCAL records, and may be PACKED to conserve storage. TELOS keywords are capitalized for readablity only.

(2) When record objects are STOREd into the TELOS database, keywords in the field declarations direct the database routines. A field marked 'SEQUENCER' will determine the order in which database objects (DBOs) of this type will be returned by associative retrieval operations. A field marked 'INDEXED' will have an inverted index list constructed for values found in that field (used to meet REQUIREMENTS: see below). A field marked 'UNSTORED' will not be placed in the database, and its value will be ignored.

(3) 'birth_date' records the time of creation for this fact node.

(4) The retina_event node has two information fields, in addition to the universal auxilliary fields birth_date and importance, and the universal information field 'class'. The pos field provides a unique identifier of the retina cell represented, while the color field records the "color" seen by that cell at the time this node was created. Other fields might have been included (eg. intensity).

(5) With the dbop (for database object pointer) TELOS allows direct access to specific objects in its database. The field 'rp' will provide a cross reference from an rp_success fact node to an rp_node defining the recognizer/predictor (RP) that succeeded. The dbop fa-

cility also allows direct read access to the database, and write access via a CHANGE function.

(6) The rp_success_event preserves not only a reference to the rp which succeeded, but also links to the particular memory nodes matched. (This information is not yet used by MUL, but is helpful in tracing MUL activity.) "nodes_matched_listcell" has this declaration:

```
nodes_matched_listcell = RECORD
    matched_node : DB-> fact_node;
    next : -> nodes_matched_listcell;
    END;
```

Associative retrieval of DBO values is based on a facility for matching an object with a pattern. TELOS patterns are not template affairs to be fitted against some object. Rather, patterns are data structures interpretable as a routine call with arguments (cf a LISP list). As data structures, they behave as records, with a structure defined as a field ROUTINE, referencing an appropriate MatchRoutine, and other fields corresponding to the formal parameter structure of the MatchRoutine. As an example, the following MatchRoutine will match any retina_event fact node whose pos field value is in pos_set, and whose color is in color_set:

```
MatchRoutine match_retina ( pos_set:retina_pos_set;
    color_set:color_sets )   MATCHING e : DB-> fact_node;
    BEGIN
    match_retina := (e->.pos IN pos_set)
        AND (e->.color IN color_set);
```

```
    END;
  REQUIREMENTS
    BEGIN   REQUIRE( retina_event )   END;
```

Both the REQUIREMENTS and MATCHING parts of the above routine are procedure definitions sharing the same formal parameter structure. Each may define its own constants, variables, even procedures and functions. The MATCHING procedure part is executed by the TELOS system when an object is to be matched. If the routine returns TRUE, a match has occurred. The REQUIREMENTS part is executed to reduce the number of candidates for which the associated MATCHING part might return TRUE. The call "REQUIRE(retina_event)", for instance, indicates that any object to be matched must contain the constant "retina_event".

As an example, suppose the variable "pat" is defined:

```
    pat : PATTERN MATCHING DB-> fact_node;
```

Then pat may be assigned a pattern:

```
    pat := -> match_retina[!
                ROUTINEREF( match_retina ),
                [lower_left], [red, black] !];
```

This pattern consists of a call to the above MatchRoutine, and if it were employed for associative retrieval, say by:

```
    ... Find( pat ) ...
```

its REQUIREMENTS would specify that any fact node its MATCHING part might be applied against should have the constant "retina_event" within it, and from those candidates it would match only those whose pos is lower_left and whose color is

either red or black. Find is a generator returning each object matched, the order in this case controlled by the SEQUENCER field 'importance' of fact_nodes.

Besides this rather ordinary mechanism for associative retrieval, TELOS provides an inverse operation: instead of returning objects that match a given pattern (as above), this inverse operation returns objects containing patterns matching a given object. Inverse pattern retieval is very useful in MUL, particularly in determining which RPs are triggered by the nodes in STM.

In exploring new and involved ideas, the exploration process does not flow smoothly from idea through design and implementation to final testing and results. At each stage, new information and new ideas arise, suggesting or even forcing changes in the implementation, the design, and sometimes in the basic ideas themselves. The primary contribution of TELOS is that it encourages, rather than impedes, this process of evolutionary design. TELOS hides the details of data base implementation, so that changes in data structures are automatically reflected in alterations in data base activities. In fact, by using mechanisms like type-checking, TELOS can even point out where further changes must be made. TELOS also places no real constraints on choice of data structures. Users are free to choose data structures on the basis of their needs rather than the needs of the system. Because the details of the database activities are hidden within TELOS, users can construct more read-

able and comprehensible programs, uncluttered with auxilliary code and data needed for data base activities. Finally, and perhaps most important, TELOS benefits not just MUL research, but the whole community of AI researchers.

Considerable effort has been devoted to the design and development of TELOS, effort that might otherwise have gone to MUL implementation in some existing language. Experience with MUL has reinforced feelings that this division of effort was justified.

References:

Becker, J.D., An Information processing Model of Intermediate Level Cognition, Unpbl. Ph.D. Diss., Stanford, 1970.

Davis, R., and J. King, An Overview of Production Systems. Stanford University Memo AIM-271, October, 1975.

LeBlanc, R. J., L. E. Travis, M. Honda and S. F. Zeigler. "TELOS Specifications", UW-Madison Academic Computing Center Tech Rpt 48, November 1977.

LeBlanc, Richard. "The Design and Rationale for TELOS, a PASCAL-based AI Language", PhD Diss. UW-Madison Academic Computing Center Tech Rpt 50, December 1977.

Moore, J., and Newell, A. How Can Merlin Understand? Tech Rep, Carnegie- Mellon University, Nov. 1973.

Piaget, Jean. The Construction of Reality in the Child, New York : Basic Books, 1954.

Piaget, Jean, and Baerbel Inhelder. The Psychology of the Child, New York : Basic Books, 1969.

Popplestone, R.J. An Experiment in Automatic Induction. Machine Intelligence 5, Edinburgh: Edinburgh University Press (1970) 203-15.

Rumelhart, D.E. and Norman, D.A., Active semantic Networks as a Model of Human Memory. Proc. 3d IJCAI. 1973, 450-7.

Rumelhart, D.E., and Norman, D.A. Accretion, Tuning, and Restructuring: Three Modes of Learning. Tech Rep 63, Center for Human Information Processing, University of California, San Diego, Aug. 1976.

Travis, L. E., R. LeBlanc, M. Honda and S. Zeigler. "Design Rationale For TELOS, a PASCAL-based AI Language", Proc. of the Symposium on AI and Programming Languages, New York : ACM, 1977.

Uhr, L. "Describing, Using Recognition Cones", University of Wisconsin - Madison Computer Science Dept. Tech. Rpt. 176, March 1973.

Uhr, L. "A parallel-Serial Recognition Cone System For Perception: Some Test Results", UW-Madison Computer Science Tech Rpt 292, March 1977.

Uhr, L., and Jordan, S. The Learning of Parameters for Generating Compound Characterizers for Pattern Recognition. 1st IJCAI. (1969) 381-415.

Uhr., L. and Vossler, C. A Pattern Recognition Program That Generates, Evaluates, and Adjusts its own Operators. Proc. West. Joint Comput. Conf., 1961, 19, 571-8.

Waterman, D.A. Adaptive Production Systems. 4th IJCAI, 1975.

Williams, H.A. A Net-Structure for Learning and Describing Patterns in Terms of Sub-Patterns. Pattern Recognition, Dec. !976.

Winston, P.H. Learning Structural Descriptions from Examples. Unpbl. Ph.D. Diss., MIT, 1970.

Winston, Patrick. The Psychology of Computer Vision, New York : McGraw-Hill, 1975.

```
253
d 253
DELETE: 253:
OK?y
252
v 252
   252: nodes of the standard importance.
pjh;;;pages=1:1
SIGNAL WHEN READY
```