MASSACHUSETTS INSTITUTE OF TECHNOLOGY

ARTIFICIAL INTELLIGENCE LABORATORY

Working Paper 206 September 29, 1980

Simulating a Semantic Network in LMS

by

Phyllis A. Koton

Submitted to the Department of
Electrical Engineering and Computer Science
on January 1, 1980 in partial fulfillment of the requirements
for the Degree of Bachelor of Science

Abstract

A semantic network is a collection of nodes and the links between them. The nodes represent concepts, functions and entities, and the links represent the relationships between varoius nodes. Any semantic network must be supplied with a language of conventions for representing knowledge as nodes and links in the network, so that storage and retrieval of knowledge can be carried out effeciently.

This thesis examines two approaches to the problem of representing real-world knowledge in a computer: one designed for use on serial computers, the other designed to run on a parallel network machine. The two formalisms are shown to be nearly identical, and a simulation of the parallel language in the serial language is given.

Disclaimer

Simulating a Semantic Network in LMS

by

Phyllis A. Koton

Submitted to the Department of
Electrical Engineering and Computer Science
on January 1, 1980 in partial fulfillment of the requirements
for the Degree of Bachelor of Science

Abstract

A semantic network is a collection of nodes and the links between them. The nodes represent concepts, functions and entities, and the links represent the relationships between various nodes. Any semantic network must be supplied with a language of conventions for representing knowledge as nodes and links in the network. The language should be compatible with the hardware implementation of the network, so that storage and retrieval of knowledge can be carried out efficiently.

This thesis examines two approaches to the problem of representing real-world knowledge in a computer: one designed for use on serial computers, the other designed to run on a parallel network machine. The two formalisms are shown to be nearly identical, and a simulation of the parallel language in the serial language is given.

Thesis Supervisor: William A. Martin
Title: Associate Professor of Computer Science and Electrical Engineering
and Associate Professor of Management

Acknowledgements

First of all, I would like to thank my thesis supervisor, William A. Martin, for his guidance, encouragement and patience.

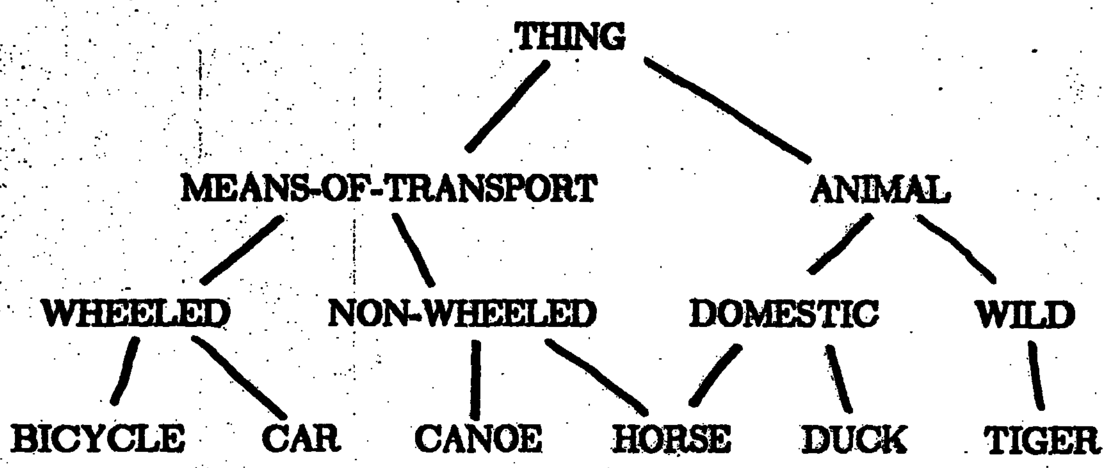
Lowell Hawkinson showed me how to use LMS, and gave me many helpful suggestions regarding my simulation.

I gained a better understanding of my own knowledge, and got some interesting ideas about representation languages from discussions with Carl Hewitt.

Finally, I would like to thank all my friends for their help and support, and I especially thank my parents for encouraging my to get "a good education."

This thesis was typeset with the aid of TEX, a typesetting system developed by Don Knuth. Richard Zippel helped.

Consider the following network:



Suppose we want to find out if there is any animal which is also a means of transport. One way to go about this would be to use a recursive algorithm to search for a connection from MEANS-OF-TRANSPORT and each of its descendants to ANIMAL. The search is complicated since the connection to ANIMAL may be indirect, through any of ANIMAL's descendants. Here we would find that HORSE is connected to ANIMAL via the node DOMESTIC. This method works slowly since it must individually examine each descendant of the MEANS-OF-TRANSPORT node, as well as many of ANIMAL's descendants.

Scott Fahlman, in his 1977 Phd thesis,[1] argued that this problem was essentially one of intersecting the set consisting of the node ANIMAL and its descendants, with the set of MEANS-OF-TRANSPORT and its descendants, and that it could better be solved using a parallel approach. He envisioned his network (as yet unbuilt) as consisting of a large number of small parallel processing elements representing nodes and links. Node elements would be able to store several distinct marker bits, which would be propagated in parallel by the link units from node to node throughout the network.

In order to find the intersection of the two sets, a marker (call it M1) would be attached to the MEANS-OF-TRANSPORT node, then passed down and attached to each of its descendants in the network. Since link units propagate all markers down one level in the hierarchy in a single parallel step, the marking of MEANS-OF-TRANSPORT and its descendants would take time proportional to the longest path from MEANS-OF-TRANSPORT to the leaves of the tree. Next, a second marker, M2, marks the ANIMAL node and is propagated down to mark

[1]Scott E. Fahlman, *A System for Representing and Using Real-World Knowledge*, Massachusetts Institute of Technology AI Technical Report 450, 1977

its descendants (again taking time proportional to the longest path). Finally, the system checks to see if there are any nodes marked by both M1 and M2; these nodes represent the intersection of the two sets. Here, HORSE would be the only node so marked.

NETL is the language Fahlman developed to represent knowledge on the parallel network machine. LMS (Martin[2] and Hawkinson[3]) is a language designed to represent knowledge on a serial machine. In my thesis, I simulated a subset of NETL in LMS. Despite the fact that they were designed for different machines, the two languages are very similar. Most NETL structures were easily transformed into LMS, but others were more difficult. The things which were hard to simulate reflect the different target machines of the two languages. This difference is also reflected in some problems in NETL which don't exist in LMS.

In order to discuss further the differences between LMS and NETL, the notational conventions of each language must be presented.

NETL. Fahlman defines several node and link types in NETL, which are represented in graphical form. There are two basic node types.

INDIVIDUAL nodes represent specific individual entities. For example, we could create a description for an individual, Clyde, who exists in the real world (also represented by an INDIVIDUAL node).

A TYPE node serves as a description whose structure and properties can be copied by INDIVIDUAL nodes. A TYPE node x can be seen as representing the "typical x." Fahlman associates each TYPE node with an INDIVIDUAL node which he claims represents the set of all things of that type; the TYPE node represents the typical member of that set. We could create a TYPE node, ELEPHANT, and an associated set node, ELEPHANT-SET (see figure 1).

There are nine link types defined in NETL. The representation of all relationships between nodes is done using these primitives.

The most important link that Fahlman describes is the *VIRTUAL-COPY (*VC) link. When a *VC link is created, the system behaves exactly as though a portion of the network had been copied, without actually creating a physical copy. For example, if we create a *VC link between CLYDE and ELEPHANT, we cause CLYDE to inherit all the properties, memberships, and restrictions attached to the ELEPHANT node. It is as if we had actually copied the entire elephant description, replacing the ELEPHANT node with CLYDE, except that we have

[2]William A. Martin, *Philosophical Foundations for a Linguistically Oriented Semantic Network* (draft), March 1979

[3]Lowell Hawkinson, "XLMS Guide" (draft), February 1979

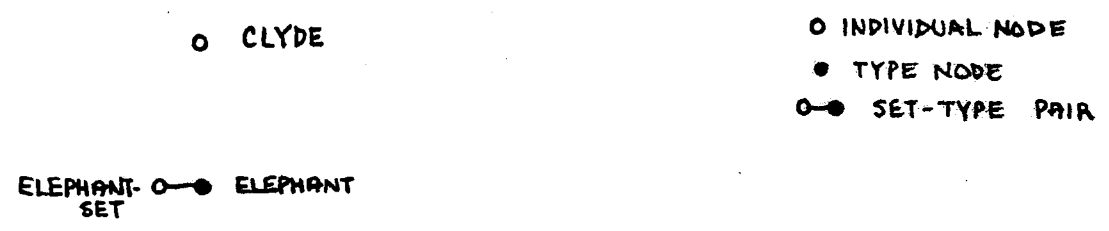


FIGURE 1: NODE TYPES

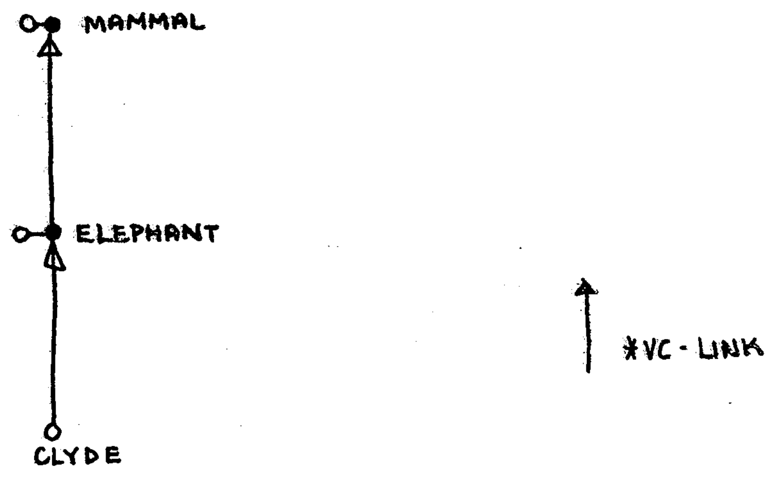


FIGURE 2: CLYDE, an ELEPHANT, a subtype of MAMMAL

only created a single link. We say that the *VC link points upward from CLYDE to ELEPHANT. *VC links are transitive, so that if we create a *VC link between ELEPHANT and MAMMAL, CLYDE will inherit the properties attached to the MAMMAL node as well (see figure 2).

In the usual tree structure, a node may have many descendants but only one immediate superior, with a single strand of ancestors. In NETL, a node can have any number of *VC links entering and leaving it. Because of this, it can have many immediate superiors in the hierarchy as well as many descendants. In addition to being a ELEPHANT, CLYDE might also be a CIRCUS-STAR and a REPUBLICAN; he can inherit from all superior descriptions (see figure 3).

However, the parallel network requires no more time to traverse the hierarchy of ancestors of a node than it would to traverse a single chain of the same length (this is assuming that there are no directed cycles of *VC links which would cause the markers to loop infinitely). A node must have at least one *VC link tying it to some "more general" node; if we know nothing else about it, it can be attached to the most general node, THING.

*VC links can be crossed by markers in either direction (although inheritance is only established in one direction—from above to below, that is, the more specific description inherits from the more general ones.) Sending markers up *VC links from a node marks all descriptions of which the node is supposed to be a virtual copy, either directly or by inheritance. Marking upward from ELEPHANT would find the descriptions of PEANUT-EATER, MAMMAL, ANIMAL and other nodes above it in the hierarchy. Marking downward finds the types and subtypes of a node. Sending markers downward from ELEPHANT would mark the descriptions of CLYDE, AFRICAN-ELEPHANT, INDIAN-ELEPHANT, and their inferiors (see figure 4).

*EXIN ("exists in") links connect an individual to the area (time, location, subject-area) in which it is considered to exist. An area is also an individual. We could connect CLYDE to the node representing REAL-WORLD with an *EXIN link (see figure 5).

A role is defined to be an INDIVIDUAL node which is connected by an *EXIN link to a TYPE node rather than an INDIVIDUAL node representing some area. The TYPE node is called the *owner* of the role. The INDIVIDUAL node NOSE is a role in the MAMMAL description. The node WEIGHT is a role in the PHYSOB ("physical object") description (see figure 6).

*MAP links are used to create an individual version of some role within a virtual copy, in order to say something which does not apply to the original. A MAP node is used to represent the new version. A copy of the NOSE node could be

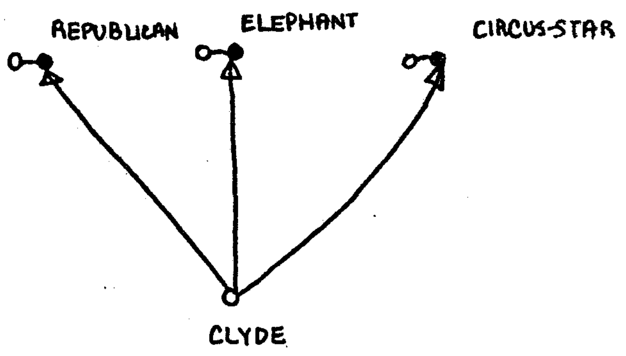


FIGURE 3: A NODE MAY HAVE MANY IMMEDIATE SUPERIORS

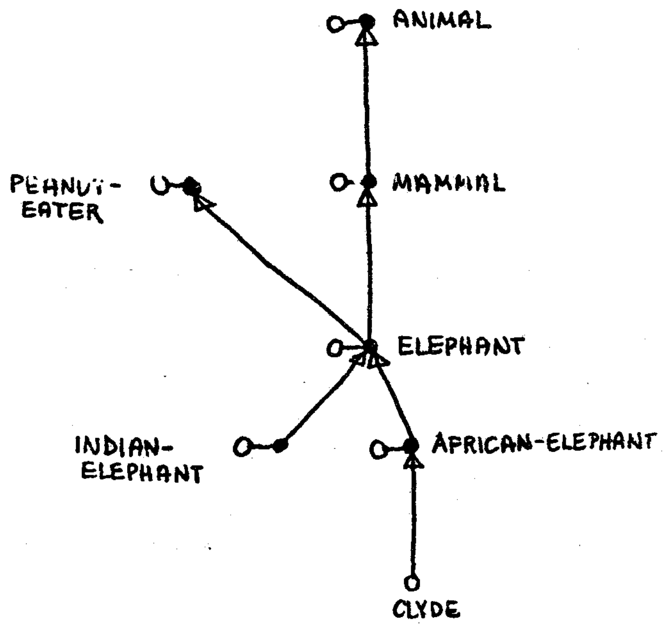


FIGURE 4:

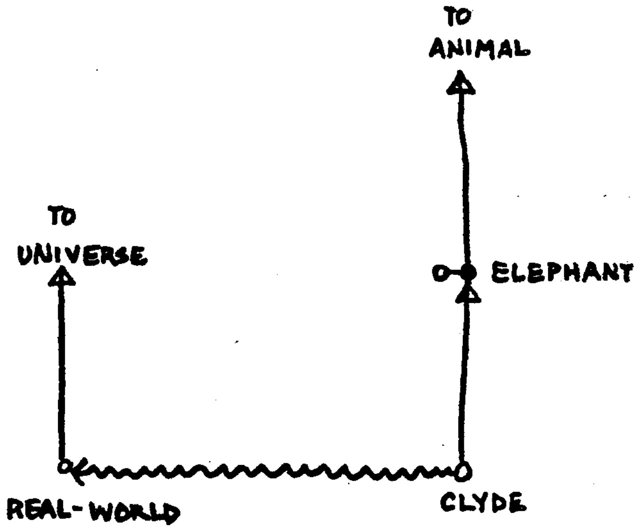


FIGURE 5: CLYDE EXISTS IN THE REAL WORLD

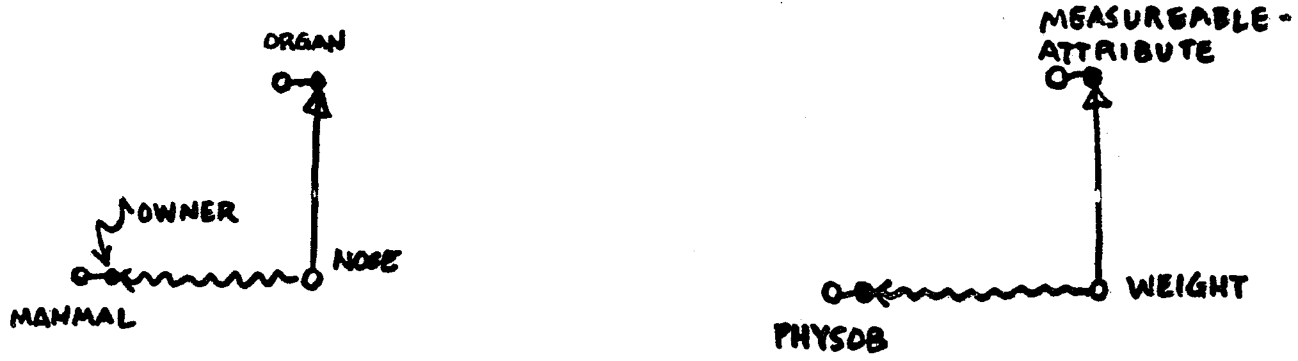


FIGURE 6: ROLE NODES. THE NOSE OF A MAMMAL. THE WEIGHT OF A PHYSOB.

created for the ELEPHANT node. The MAP node, which gets the name TRUNK, is connected to the NOSE node with a *MAP link to indicate the role of which it is a copy, and it is connected to the ELEPHANT node with an *EXIN link to indicate its owner (see figure 7).

The mapped description can be modified or added to without altering the role node being mapped. We would use a MAP node in representing the fact that Clyde's trunk was unusually short, linking the CLYDE'S-TRUNK MAP node to the TRUNK node with a *MAP link, and to the CLYDE node with an *EXIN link, and then attach the additional information to the CLYDE'S TRUNK node. When we look for information about Clyde's trunk, the original TRUNK description as well as CLYDE's copy are used, but the individual copy has precedence in case of a contradiction (see figure 8).

*EQ (equal) links are used to equate two nodes. During marker propagation, any two nodes connected by an *EQ link behave as though they were a single node: anything known about one applies to the other as well. If Clyde is the son of Jumbo, then an equal link can be created linking JUMBO with the node representing the FATHER of CLYDE (see figure 9). The link is also used to assign a value to a property. We could create an *EQ link between the TRUNK-LENGTH node and the node representing 1.3 meters (see figure 10).

In order to examine a mapped role within a description, the description must first be activated with a marker, and then the role can be examined using a different marker. Suppose we want to find the properties associated with Clyde's trunk. We first activate the CLYDE description by placing a marker M1 on the CLYDE node, and propagating it upward in the hierarchy. The marker is propagated up all *VC links and across all *EQ links in either direction. This marks all CLYDE's parents in the network (see figure 11).

We now want to mark the role TRUNK. A second marker, M2, is placed on the TRUNK node, and propagated up all *VC links and across all *EQ links in either direction. It is also propagated up or down any *MAP link pointing to a MAP node whose owner node was activated by M1. If placed on a MAP node, it will be propagated upward to the node above, if placed on a role node, it will be propagated down the *MAP wires of all the nodes beneath it. This marking causes all the M2-marked nodes to function jointly as the description of Clyde's trunk (see figure 12).

A *CANCEL link is used to cancel some role that would otherwise be inherited. If all elephants have hair, but Clyde is bald, we would represent this by extending a cancel link from CLYDE to HAIR (see figure 13).

During marker propagation, marker-bits are used in pairs: the first one to

⇑ *MAP-LINK

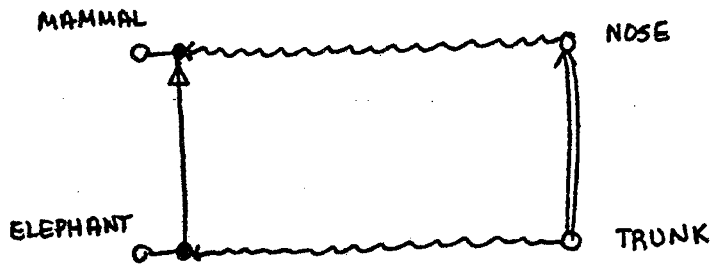


FIGURE 7: THE NOSE ROLE MAPPED FROM MAMMAL TO ELEPHANT

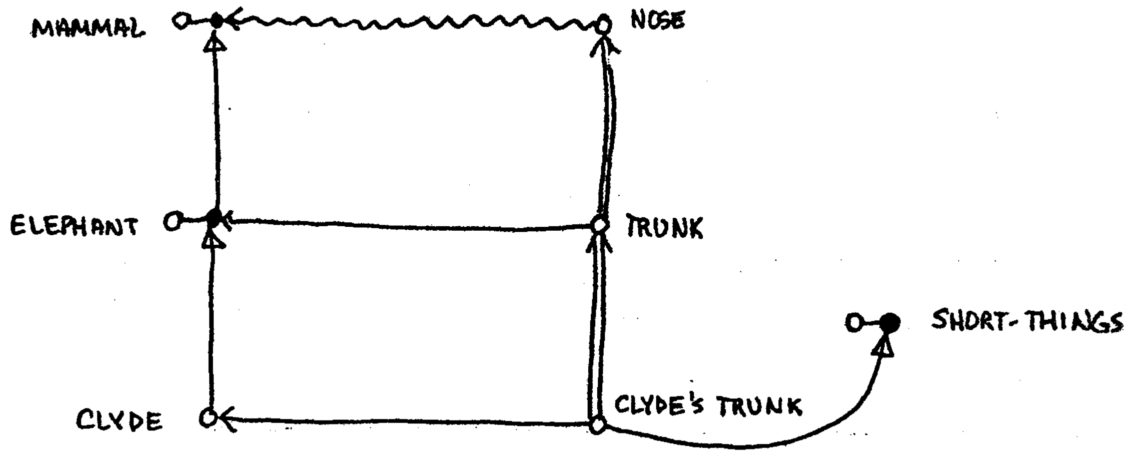
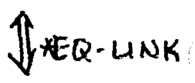


FIGURE 8: **"CLYDE'S TRUNK IS SHORT."**



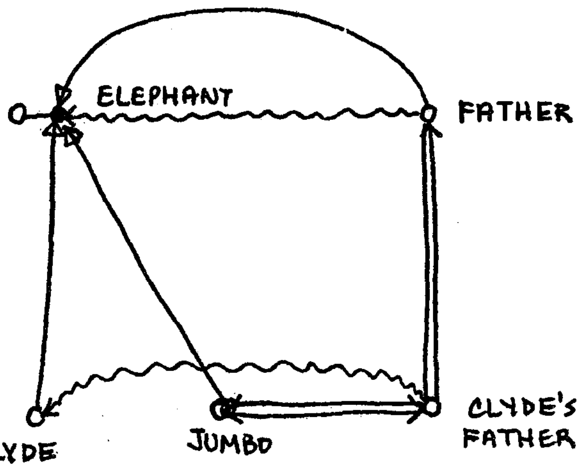


FIGURE 9: JUMBO is the FATHER of CLYDE.

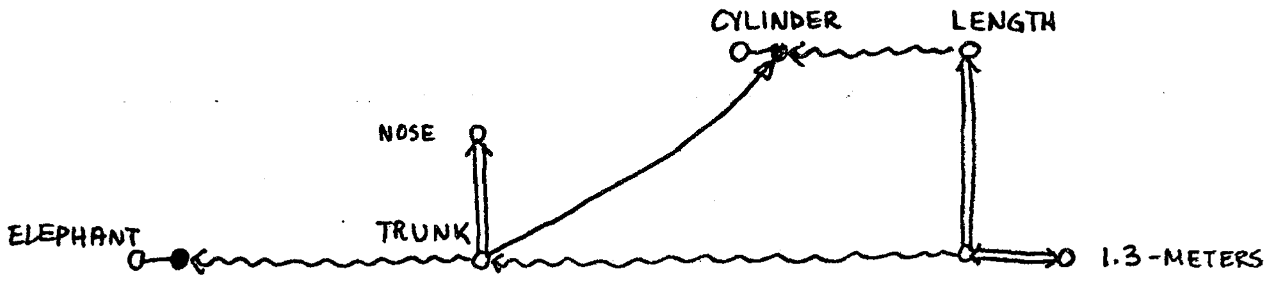
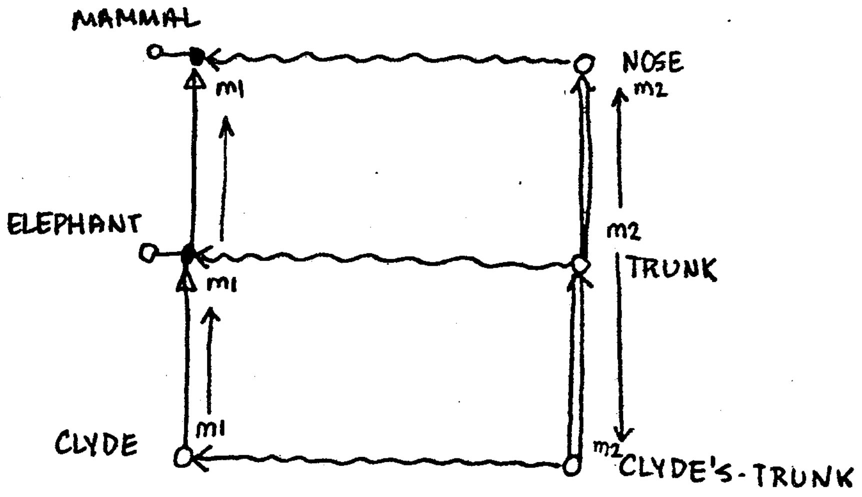


FIGURE 10: A TRUNK is a CYLINDER WITH LENGTH 1.3 METERS.



FIGURES 11, 12 : MARKING TO FIND PROPERTIES ASSOCIATED WITH CLYDE'S TRUNK.

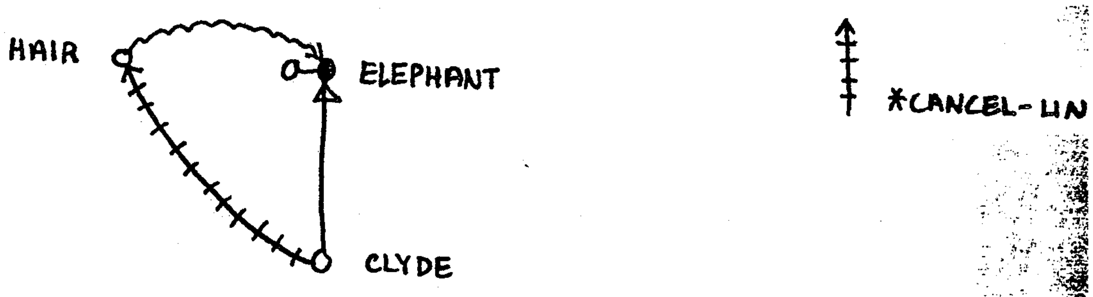


FIGURE 13 : ELEPHANTS HAVE HAIR, BUT CLYDE'S DOESN'T.

activate a copy of a description, then the other to indicate cancellations within that copy. The cancellation marker is placed on any node pointed to by a *CANCEL link whose tail is a node activated by the first marker. The presence of a cancellation marker on a node inhibits the further passage of markers through the node. Any MAP nodes for which it is the owner will not be activated. The cancellation marker is propagated to any other node tied by a *EXIN link to the cancelled node. Nodes marked with the cancellation marker are ignored when the system looks for nodes of its type (see figure 14).

In order to cancel some *identity* (indicated by a *VC or *EQ link) that would otherwise be inherited, a *CANVC link must be used. Since we first activate a description and then send out cancellation markers, cancelling a *VC or *EQ link with a *CANCEL link would cause us to lose track of the activation markers once they had passed the cancelled nodes, since any subsequent markers looking for the activation markers would be stopped at the cancelled node. The *CANVC link causes a cancellation marker to be placed on an identity node *during* an activation scan. The corresponding activation marker will not mark or pass through any node so marked. A race occurs between the activation and cancellation markers on the way to the node to be cancelled, but the cancellation markers always win, since they have a direct route to the node to be cancelled, while the activation markers must always pass through at least one other node. Consider the network fragment in figure 15.

The presence of the *CANVC link from CEPHALOPOD to SHELL-BEARER inhibits the passage of markers to the role node, SHELL, when the system searches for a CEPHALOPOD's SHELL. A race is created between the marker seeking to activate the SHELL-BEARER description and the marker trying to cancel it. The cancellation marker wins since it doesn't have to pass through any intermediate nodes on its way to SHELL-BEARER. When the activation marker arrives at the SHELL-BEARER node, it does not attach itself or pass through to continue marking that path. The reassertion of the *VC link from NAUTILUS to SHELL-BEARER allows the markers to reach that node when searching for a NAUTILUS's SHELL. Here, the activation marker wins since it has a direct route to SHELL-BEARER, while the cancellation marker must pass through CEPHALOPOD. When the cancellation marker arrives at SHELL-BEARER, it does not cancel the activation marker already there; it only prevents new activation markers from entering.

LMS. As in NETL, knowledge in LMS is represented in terms of a set of nodes arranged in a tree-like structure, so that individual nodes inherit most of their properties from their superiors.

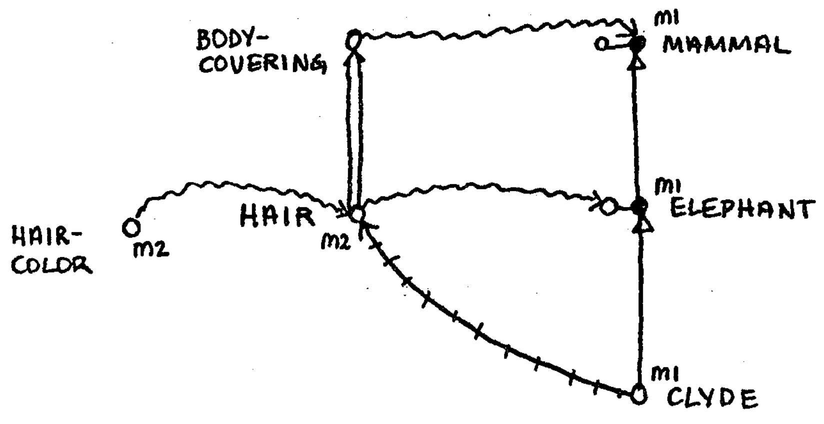


FIGURE 14: CANCELLATION MARKERS

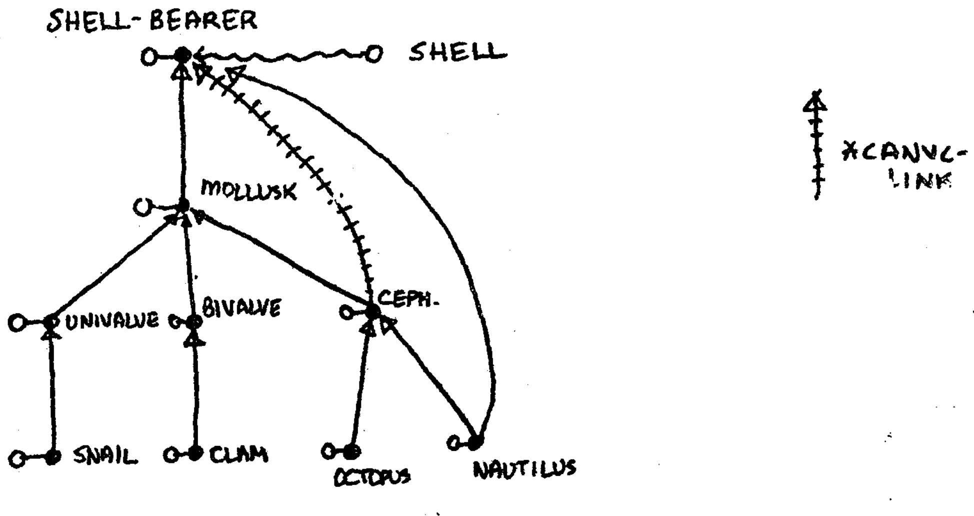


FIGURE 15: *CANVC LINK IS USED TO CANCEL AN IDENTITY

LMS nodes are consructed from the most general node, SUMMUM-GENUS, using the binary operation *specialization*. Nodes have the form *(genus specializer)* where *genus* is a superior node in the tree and *specializer* is another node or an atomic symbol. We say that a node C = (A B) is a specialization of node A, the node in its genus position.

A NETL node is represented in LMS by a kind of node called a concept. A concept is written as a triple of the form (ilk*tie cue). The ilk specifies the fundamental identity of the concept,[4] and must always be another concept (in order to have someplace to start, LMS provides the concept [summum-genus], whose ilk is itself, equivalent to the THING node in NETL). For example, the ilk of the node representing Clyde would be ELEPHANT. The ilk of the node TRUNK could be NOSE.

Concepts must be named uniquely, and (unlike NETL nodes) require a unique name to exist. The cue of a concept is chosen so that the ilk, tie, and cue together constitute a unique identifier. The cue may be an atomic symbol. The cue of the node representing Clyde would typically be (NAME*I "CLYDE"), but could also be the atomic symbol "CLYDE".

The tie is chosen (from a set provided in LMS) depending on what the node represents and the relation between the ilk and cue.

An INDIVIDUAL node is created by setting the tie to *INDIVIDUAL-TIE (abbreviated *I). The LMS representation of the NETL node CLYDE is

(ELEPHANT*I "CLYDE").

To represent the different subclasses of a concept, the *SPECIES-TIE (*S) is used. Any concepts that have the same ilk and *SPECIES-TIE are mutually exclusive (and by inheritance, so are their two sets of inferiors). For example, we could subclassify the LIVING-THING concept by

(LIVING-THING*S "PLANT")

and

(LIVING-THING*S "ANIMAL").

A difficulty arises when a set is to be partitioned into more than one set of mutually exclusive subdivisions. For example, we could partition LIVING-THING by PLANT/ANIMAL and MULTI-CELLED/SINGLE-CELLED. If we attempted to represent this by creating the concepts

(LIVING-THING*S "MULTI-CELLED")

and

(LIVING-THING*S "SINGLE-CELLED")

[4]There is a similar mechanism in NETL, whereby a node is connected to the superior node representing its most fundamental identity with a *parent wire* rather than a *VC link.

(in addition to the PLANT and ANIMAL concepts created previously), it would cause problems such as PLANT and MULTI-CELLED being mutually exclusive (see figure 16).

In order to avoid this problem, we must represent the two divisions in a way that indicates that they are subclasses of LIVING-THING but does not define them to be disjoint. The STEREOTYPE-TIE (*T) is used for this. Concepts created with a *STEREOTYPE-TIE are not disjoint with other concepts having the same ilk. A concept

(LIVING-THING*T CELL-STRUC)

can be created to stereotype LIVING-THINGs by their cell structure, then this can be partitioned using *S into PLANT and ANIMAL. Another node is then created to stereotype LIVING-THINGs by their cell-number,

(LIVING-THING*T "CELL-NUMBER").

The CELL-NUMBER concept is then partitioned into

(CELL-NUMBER*S "MULTI-CELLED")

and

(CELL-NUMBER*S "SINGLE-CELLED").

(See figure 17).

We can reduce the representation by one concept by directly subclassifying LIVING-THING by PLANT and ANIMAL, then creating the CELL-NUMBER concept as described above. This seems to mean that LIVING-THINGs are either PLANTS or ANIMALS, and in addition they can be classified by their cell number. Some find the symmetrical structure described above more "impartial". However, Martin[5] points out that some philosophers distinguish between "natural kinds" and abstractions such as grouping by the number of cells.

Role and MAP nodes are created using the *ROLE-IN-TIE (*R). For example, if the NETL node NOSE is represented as (PHYSICAL-OBJECT*I "NOSE"), TRUNK would be represented as (NOSE*R ELEPHANT) and Clyde's trunk would be represented as (TRUNK*R CLYDE). This notation is enough to completely specify the node. It is not necessary to simulate the *MAP and *EXIN links in LMS since the role node being mapped and the owner node can be found through the ilk and cue, respectively.

The *FUNCTION-TIE (*F) indicates that the cue of the concept is to be applied as a function to the ilk. The concept BOY could be represented as

(CHILD*F MALE)

There is no real equivalent to this in NETL.

[5]personal communication.

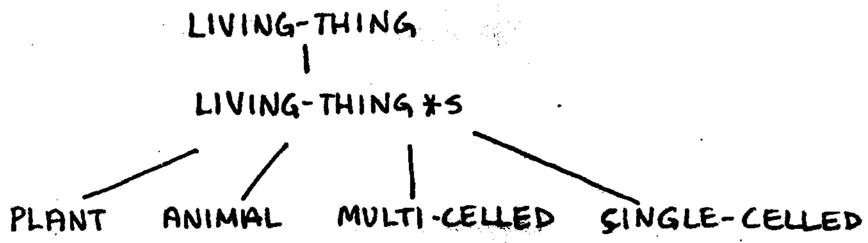


FIGURE 16:

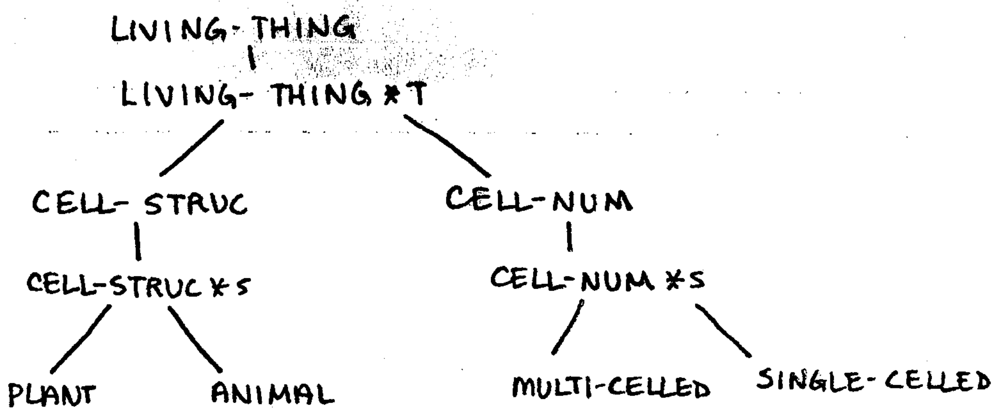


FIGURE 17:

Because of their structure, LMS concepts form a hierarchy in which the ilk points "upward". We say that (LIVING-THING*S PLANT) and (LIVING-THING*S ANIMAL) are inferiors of LIVING-THING, while (MAMMAL*S ELEPHANT) and, by transitivity, (ANIMAL*S MAMMAL) are the superiors of (ELEPHANT*I CLYDE).

This "built-in" framework defines one type of inheritance in the network: a concept inherits all the ROLE's and FUNCTION's of its genus (ilk*tie).[6] However, it is not sufficient to represent a NETL node since each concept in this LMS taxonomy has only one immediate superior. In NETL, a node may have many immediate superiors because it may have many *VC links emanating from it. Attachments[7] are used to create this condition in LMS. Attachments create a directed link between two concepts.

There are nine attachment relations defined in LMS. Several are particularly prominent in the creation of more complex relations. Non-coincidentally, they also correspond almost exactly in meaning and use to NETL links.

In order to say that an entity is of a particular kind[8] the ∤CHARACTERIZATION (∤C) attachment is used. [A ∤CHARACTERIZATION B] means that anything that can be described as A can be described as B. Informally, this implies that A's inherit the properties of B's, in the same way as a *VC link from A to B signifies that A inherits from B. To represent the fact that Clyde is an elephant, we would write

[CLYDE ∤C ELEPHANT]

and say that ELEPHANT is (attached as) a characterization of CLYDE.

At this point one might wonder what is the difference between the inheritance implied by characterization (for example, [ELEPHANT ∤C MAMMAL]) and that implied by specialization (as in (MAMMAL*S ELEPHANT)). The difference is, according to Martin,[9] that "in the case of specialization we are not guaranteed" that anything which is an ELEPHANT is also a MAMMAL. Consider the concept DOG-HOUSE=(HOUSE*S DOG), where the prototypical HOUSE is one for humans. In this case we do not want to say that a DOG-HOUSE is a HOUSE, since it differs too much from the typical HOUSE. Rather, specialization indicates that the concept HOUSE has been modified to be appropriate for dogs.[10] The

[6]Martin, p. 85

[7]Also called *zone relations*.

[8]Martin, p. 49

[9]p. 86

[10]p. 77

ilk of a concept provides a basis for the construction of the concept, but does not ensure that "a criterial description is inherited."[11]

The ∤ROLE-IN attachment is used to represent an entity which is part of the structure of another description. We say

[A ∤ROLE-IN B]

to state that A exists in the structure or area defined by B. In this respect, the *EXIN links and ∤ROLE-IN attachments are identical.In LMS, we would indicate that mammals have noses by saying

[NOSE ∤R MAMMAL].

It is often the case that we are at some node B, where B has been attached as a characterization to several nodes, and we wish to access those nodes. In NETL this could be done by sending markers *down* the *VC links pointing to B (remember, in NETL markers could cross links in either direction). In LMS, we achieve the same effect using *complimentary* attachment relations. For example, the ∤EXEMPLAR (∤E) attachment is a complimentary relation for ∤CHARACTERIZATION. This is written

[A ∤EXEMPLAR B].

We can say that B is an exemplar of A whenever we say that A is a characterization of B. The LMS statements[12]

[A ∤CHARACTERIZATION B]

[B ∤EXEMPLAR A]

indicate that B is attached to A by the attachment relation ∤C, and also that A is attached to B by attachment realtion ∤E.[13]

[CLYDE ∤CE ELEPHANT CIRCUS-STAR] states that ELEPHANT and CIRCUS-STAR are attached as characterizations of CLYDE, and also that CLYDE is attached as an exemplar to both ELEPHANT and CIRCUS-STAR.

The LMS statement [A ∤CC B] indicates that the concepts A and B are equal, since it is equivalent to saying $A \subset B$ and $B \subset A$ in predicate calculus.[14] This is the same as the *EQ link in NETL.

From the above descriptions of NETL and LMS, it is apparent that the two languages are very similar. One way in which they both differ from other knowledge representation languages is that they use a small number of links (each has about

[11]Martin, p. 87

[12]In an abbreviated form, [A ∤CE B]

[13]Of course, the converse relation always holds whether or not it is explicitly stated. It would take a lot of room to store them all explicitly, though.

[14]Martin, p. 52

nine) to represent the relationships between nodes. More complicated relationships are constructed using these links. Other representation languages use a separate link type for each type of relationship between the nodes.[15]

For example, to represent "John loves Mary" in the typical representation language, one would merely have a "LOVES" link running from the node representing John to the node representing Mary. In NETL or LMS, a structure is created to represent the relationship "LOVE," and an instance of this is created for the specific case of John and Mary (see figure 18).

The advantage to the method used by NETL and LMS is that since the relationships are all explicitly stated (here, LOVES, LOVER, and LOVED), it is easy to describe them just as if they were objects. For example, to indicate that the state of John's love for Mary is fleeting we could attach a link from the node representing their love to the node representing fleeting-things. We could also attach properties to, say, the LOVER node, which would then be inherited by each of its inferiors.

Also, in the case of NETL, each link type is a separate functional unit, so having a small number of link types is an obvious advantage.

Meta-description is used to construct descriptions which talk about the representational structure itself, "to describe the correspondence between one's conceptual structure and the world". It is indicated in LMS by the ∤METACHARACTERIZATION (∤M) attachment. To indicate that Fido does not have a tail we could say

```
                [(TAIL*R (DOG*I FIDO)) ∤M NON-EXISTENT]
```

Fahlman does metacharacterization by replacing the usual link element with an *individual statement* (*IST) node. In this way a statement can be described in exactly the same way as an individual object.

There is an equivalent way of representing this in LMS: that is to explicitly state the relationship and then describe it.[16] For example, since [A ∤CHARACTERIZATION B] can be thought of as "A is-a B," the relationship between A and B is "BE" and can be expressed as

```
[BE-1
    [(SUBJECT*R BE-1) ∤C A]
    [(OBJECT*R BE-1) ∤C B]]
```

[15]The OMEGA system ("Knowledge Embedding in the Description System OMEGA," Carl Hewitt, Guiseppe Attardi, Maria Simi, MIT AI Lab Working Paper (Draft), December 1979) however, uses a minimal number of primitives—two: the equivalents of ∤CHARACTERIZATION and ∤ROLE-IN.

[16]Martin, p.128

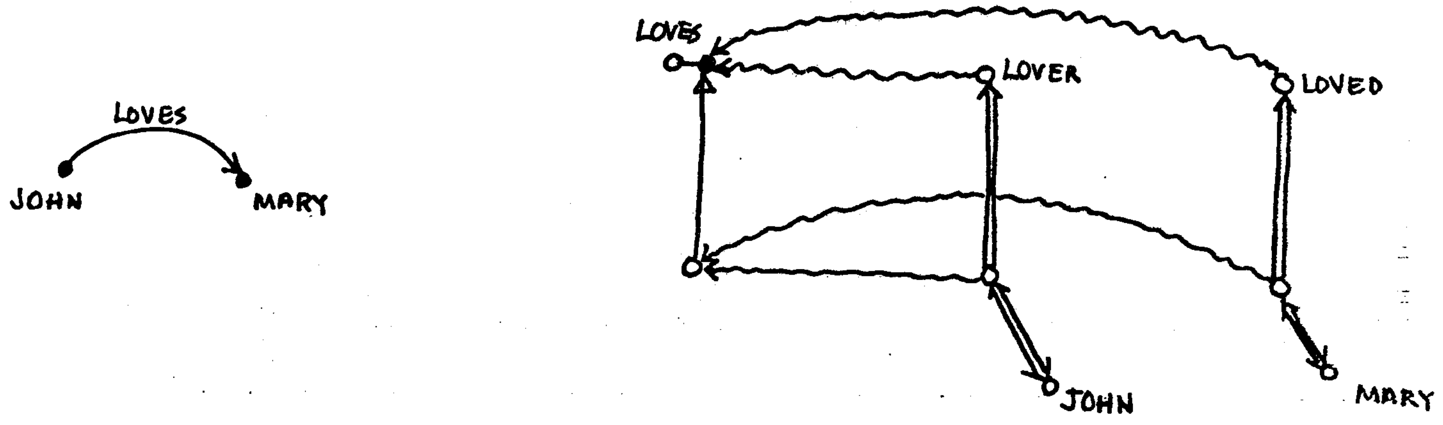


FIGURE 18: "JOHN LOVES MARY" REPRESENTED IN A TYPICAL FORM (LEFT) AND IN NETL (RIGHT).

The explicit state can then be further described, e.g. "an elephant is probably a mammal" could be represented as[17]

```
[BE-2
    [(SUBJECT*R BE-2) ∤C ELEPHANT]
    [(OBJECT*R BE-2) ∤C MAMMAL]
    ∤F PROBABLY]
```

Note that in both LMS and NETL, in order to describe a primitive, it is necessary to rise one level of meta-description.

Another way in which Fahlman does meta-characterization is to attach modifers directly to the links. The **LIKE modifier, when attached to a *VC link, causes a virtual copy to be created which can be modified in the usual way, but does not impart formal class membership.[18] Suppose that we want to say that a toy gun is like a gun. It may look like a gun, and shoot little plastic bullets, but it is not a "real" gun. Attaching a **LIKE modifier to the *VC link between TOY-GUN and GUN causes a virtual copy of gun to be created for the toy gun node, which can then be modified to indicate that it is not a weapon. However, during marker scans to answer "is-a" questions, the **LIKE modifier disables the *VC link to which it is attached.

In LMS, "toy gun" can be represented by the *FUNCTION tie. The concept "TOY" is applied as a function to the concept "GUN". We can then characterize TOY-GUN=(GUN*F TOY) as being a toy, from which it will inherit, but it will not inherit from gun.[19]

The *CANCEL link is one NETL construct that was difficult to translate to LMS. The purpose of the *CANCEL link is to prevent the inheritance of a description by inhibiting the passage of markers during the activation of the description. Since my simulation does not use any markers, creating a "cancel" attachment in LMS and treating it as an indication to ignore the attached information was awkward. Let us consider again the example of cancellation given in the previous section.

[17]Martin, p. 128

[18]The action of the **LIKE modifier is thus similar to specialization, which creates a subtype of the genus modified by the specializer, but does not guarantee class inclusion (pointed out by W. A. Martin).

[19]Another representation which could be used to represent this in both NETL and LMS would be to have a concept representing the general concept "GUN", and then subclassify it by "WEAPON-GUN", "TOY-GUN", etc. We could then attach to "GUN" all the attributes that we wanted to be inherited by all guns, and attach to the individuals their distinguishing characteristics and roles (for example, we might attach "FATAL BULLET" as a role in "WEAPON-GUN"). This seems to remove it from the realm of metacharacterization, however.

Here is the same structure represented in LMS[20] :

[SHELL ‡R SHELL-BEARER]
[MOLLUSK ‡C SHELL-BEARER]
[UNIVALVE ‡C MOLLUSK]
[BIVALVE ‡C MOLLUSK]
[CEPHALOPOD ‡C MOLLUSK]
[CEPHALOPOD ‡CANCEL SHELL-BEARER]
[NAUTILUS ‡C SHELL-BEARER CEPHALOPOD]
[OCTOPUS ‡C CEPHALOPOD]

Suppose we now present the system with the questions

(1) Does a cephalopod have a shell?
(2) Does a nautilus have a shell?

Let us examine a few proposed techniques for extracting answers to these questions from the LMS world.

One method would be to search for a ‡CHARACTERIZATION attachment from SHELL-BEARER to a concept, and then check whether it has been cancelled. This would correctly find SHELL-BEARER attached as a ‡CHARACTERIZATION of CEPHALOPOD and its cancellation, however it would also mistakenly find the cancellation for NAUTILUS. Similarly, we could check to see whether the SHELL-BEARER concept could be reached by *any* route. Although this would find the link from NAUTILUS to SHELL-BEARER, it loses because it would also find the path from OCTOPUS to SHELL-BEARER via MOLLUSK.

Any solution that simply searches for assertion and cancellation links will not work, because there is an ambiguity in the way we represent the information. We say, for example, that a cephalopod is a MOLLUSK and therefore a SHELL-BEARER, yet we also say that a CEPHALOPOD is not a SHELL-BEARER.[21] In order to get the right answer using this representation, we have to introduce a technique which uses more information than just the presence of a certain type of link.

We could check which information was the most "recent," that is, the most directly linked to a node. One way to do this would be to count the number of

[20]LMS allows the creation of new attachment relations; here I have hypothetically created a ‡CANCEL attachment which indicates that the concept which it is attaching is no longer to be inherited.

[21]This is also true in the NETL representation. Fahlman has suggested (in personal communication) an alternate representation in which the *CANVC link from NAUTILUS cancels the *CANVC from CEPHALOPOD to SHELL-BEARER, which seems cleaner.

links between two concepts.[22] In this example, a CEPHALOPOD would not be found to have a shell because SHELL-BEARER is cancelled at a distance of one link while asserted at a distance of two. A NAUTILUS has a shell because it is asserted to be a shell bearer at distance one, but cancelled at a distance of two. Descriptions which are asserted and cancelled at the same distance need further investigation. (Of course, as Fahlman points out, in the case of descriptions that are cancelled and asserted at a distance of one, both links can just be eliminated.) This technique is isomorphic to Fahlman's cancelling scheme, where the marker (activation or cancellation) that wins the race is the one with the shortest route to the node. The problem with this method is that counting links seems inelegant, just as the race in NETL seems to be.

Other solutions involve altering the LMS representation of this fragment.

Hewitt suggests that the statement "mollusks have shells" is erroneous and should not be stated. He suggests creating a subclass of mollusk (MOLLUSK*T SHELLED-MOLLUSK) which are SHELL-BEARERs. Univalves and bivalves are characterized as SHELLED-MOLLUSKs and CEPHALOPODS are characterized as just plain MOLLUSKs. NAUTILUS is then characterized as being a SHELLED-MOLLUSK (see figure 19).

This representation is unambiguous, but it has a big problem—we just can't go around creating a new classification for a concept every time there is an instance of that concept for which the inheritance of some description would be inappropriate. Cancellations of parts of descriptions are very common and if you had to create a node for everything that something wasn't, the network would get impossibly large. Consider the statement "Mammals are hairy" and the associated hierarchy (see figure 20).

Now suppose we wanted to represent the fact that Clyde is bald. There is no easy way to do it. A large number of new nodes must be created, and links re-arranged. The resulting network looks something like figure 21. This is the advantage of the NETL *CANCEL link—it allows you to cancel only part of a description while retaining all the other characteristics.

Martin suggested creating a NON-SHELL-BEARER concept and somehow indicating that the state of being a NON-SHELL-BEARER is mutually exclusive with the state of having a shell. This representation could be used if the attachment-counting algorithm was used. He also points out a problem in this representation: one should be able to think about a cephalopod not being a SHELL-BEARER

[22]Kent Pitman suggested an algorithm for this.

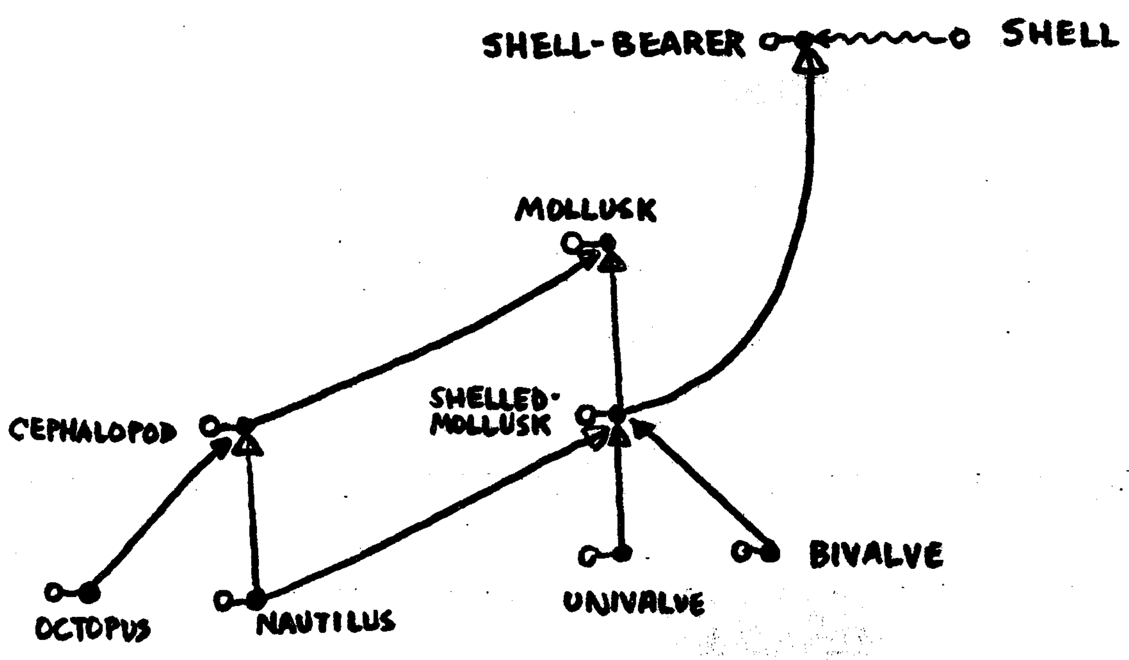


FIGURE 19.

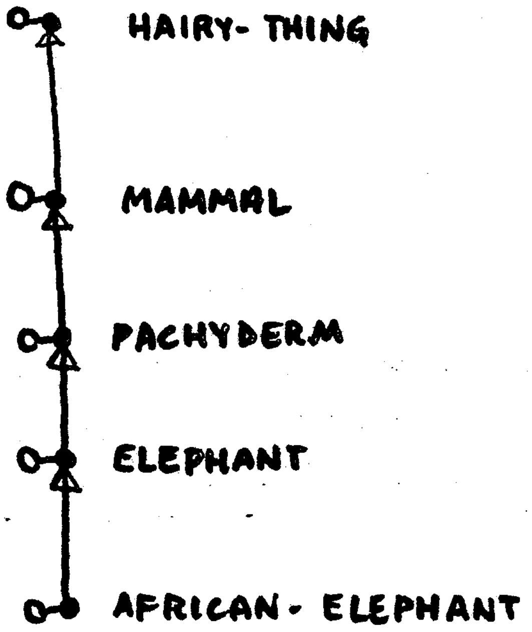


FIGURE 20.

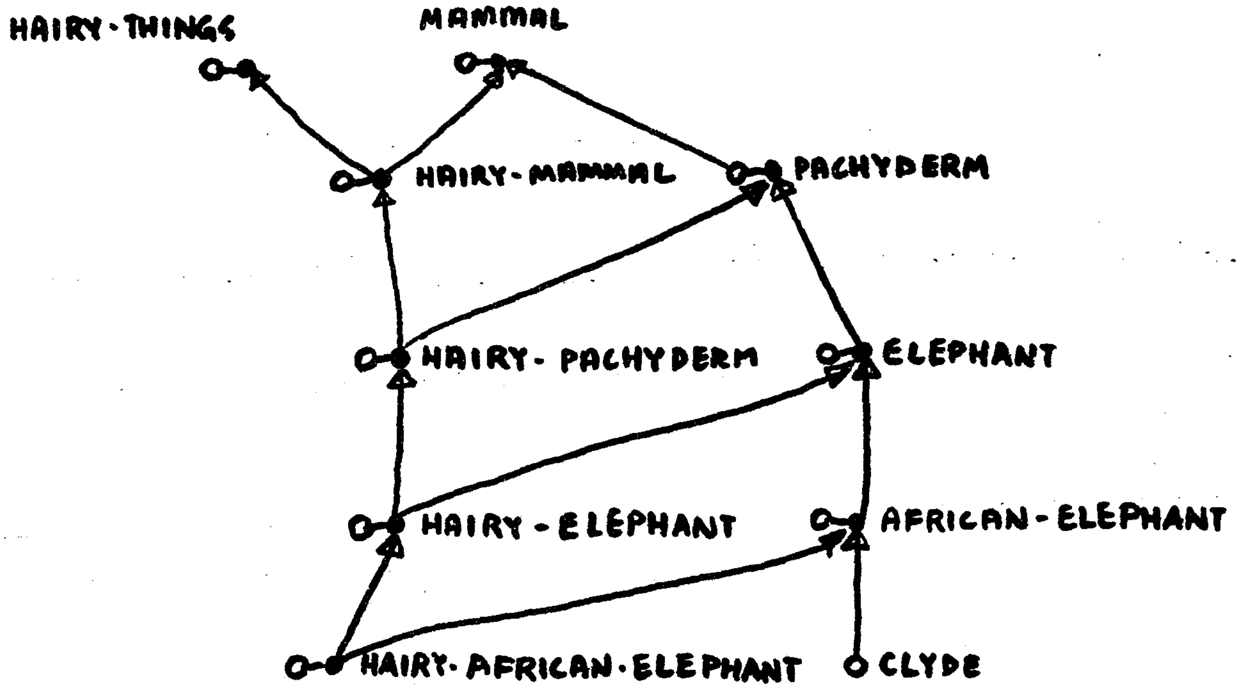

HAIRY-THINGS
MAMMAL
HAIRY-MAMMAL
PACHYDERM
HAIRY-PACHYDERM
ELEPHANT
HAIRY-ELEPHANT
AFRICAN-ELEPHANT
HAIRY-AFRICAN-ELEPHANT
CLYDE

FIGURE 21.

without having to think of what other things a cephalopod might be.[23]

The following approach was also suggested by Martin:[24] rather than preventing an undesired description from being inherited, a mapping of the description is created and it is then indicated that this description has no match in the real world. Since we are now describing the correspondence between the description and the real-world, we use meta-description. For example, a new concept (SHELL*R CEPHALOPOD) is created and meta-characterized (using the ‡METACHARACTERIZE attachment) as being non-existent. Since NAUTILUS inherits this description, a new concept (SHELL*R NAUTILUS) is created for nautilus which is *not* meta-characterized as being non-existent. This is the representation I used although it also has a problem—mainly that we are describing a cephalopod's shell when at the same time we are trying to say is that it has none,[25] although Martin claims that this is unavoidable.[26]

Hewitt has suggested a method for representing this in the OMEGA formalism which is unambiguous and does not require the creation of any new nodes.

```
(((a Mollusk) except (a Cephalopod)) is (a Shell-bearer))
((a Shell-bearer) has (a Shell))
((an Octopus) is (a Cephalopod))
(((a Cephalopod) except (a Nautilus)) is (not (a Shell-bearer)))
((a Cephalopod) is (a Mollusk))
((a Nautilus) is (a Cephalopod))
((a Nautilus) is (a Shell-bearer))
```

The *except* clause signifies that nothing can be inferred about its referent from the statement. Thus from the first statement we cannot deduce that a cephalopod is not a shell-bearer. How to represent something like this in NETL or LMS is a problem for future study.

We have already seen that a parallel approach allows the quick intersection of sets, while a serial machine does this very slowly. There are some problems in NETL due to the parallel implementation, which therefore do not occur in LMS.

[23]Martin, p. 121

[24]p. 121

[25]Brian C. Smith, "Levels. Layers, and Planes: The Framework of a System of Knowledge Representation Semantics," M.S. Thesis,Massachusetts Institute of Technology , 1978, p. 95

[26]Martin, p. 120

The most serious is the difficulty in preventing markers from reaching undesired nodes while propagating through the network.

This problem, which Fahlman calls "copy confusion," is caused by trying to look at two distinct copies of the same individual at once. The most common occurence of this is when an node and any of its role-nodes are described as instances of the same superior type-node. Fahlman gives examples: a PERSON has a MOTHER who is also a PERSON; an ELEPHANT is a PHYSOB with a TRUNK that is a PHYSOB. In the marker-passing scheme, there is a problem of confusing the two copies of the shared description. For example, consider the node WEIGHT, a role in the PHYSOB description. If we try to find the WEIGHT of the TRUNK of CLYDE, we first activate the CLYDE description, then the TRUNK description (see figure 22).

When we activate the WEIGHT description, however, the markers will traverse map links activated by both of the first two markers. Thus they find their way to both the node representing CLYDE's WEIGHT and the node representing CLYDE's TRUNK-WEIGHT (see figure 23).

In order to avoid this, the usual rules of marker propagation must be altered, so that the ELEPHANT description and the TRUNK description use their copies of the PHYSOB description at different times. This involves propagating a marker from the weight node to only TRUNK-activated descriptions, then pausing to erase the markers from certain nodes, and then continuing to propagate the marker to only CLYDE-activated descriptions.

There is no analogous problem in LMS. CLYDE's TRUNK's WEIGHT, represented as

[((WEIGHT*ROLE-IN TRUNK)*ROLE-IN CLYDE)]

is attached to the CLYDE node with a ∦ROLE-IN tie. The system checks whether CLYDE or any of its characterizations has as a role anything which is a member of the class (WEIGHT*ROLE-IN TRUNK).[27] In the process of searching for CLYDE's TRUNK's WEIGHT, my scheme looks at the node representing CLYDE's WEIGHT but since it has no role in (TRUNK*R CLYDE) and is not characterized as being of the class (WEIGHT*R TRUNK) it is eliminated as a possibility. The parallel scheme forces NETL to consider CLYDE's WEIGHT in the search, since it satisfies the criteria `(or (markedp M1) (markedp M3))`. The problem here is one of implementing a correct semantic notation (for it works properly in LMS)

[27]CLYDE's TRUNK's WEIGHT can also be represented as [(WEIGHT*ROLE-IN (TRUNK*ROLE-IN CLYDE))]. These two concepts can be equated (using mutual ∦CHARACTERIZATION attachments) and information attached to either will be inherited by the other.

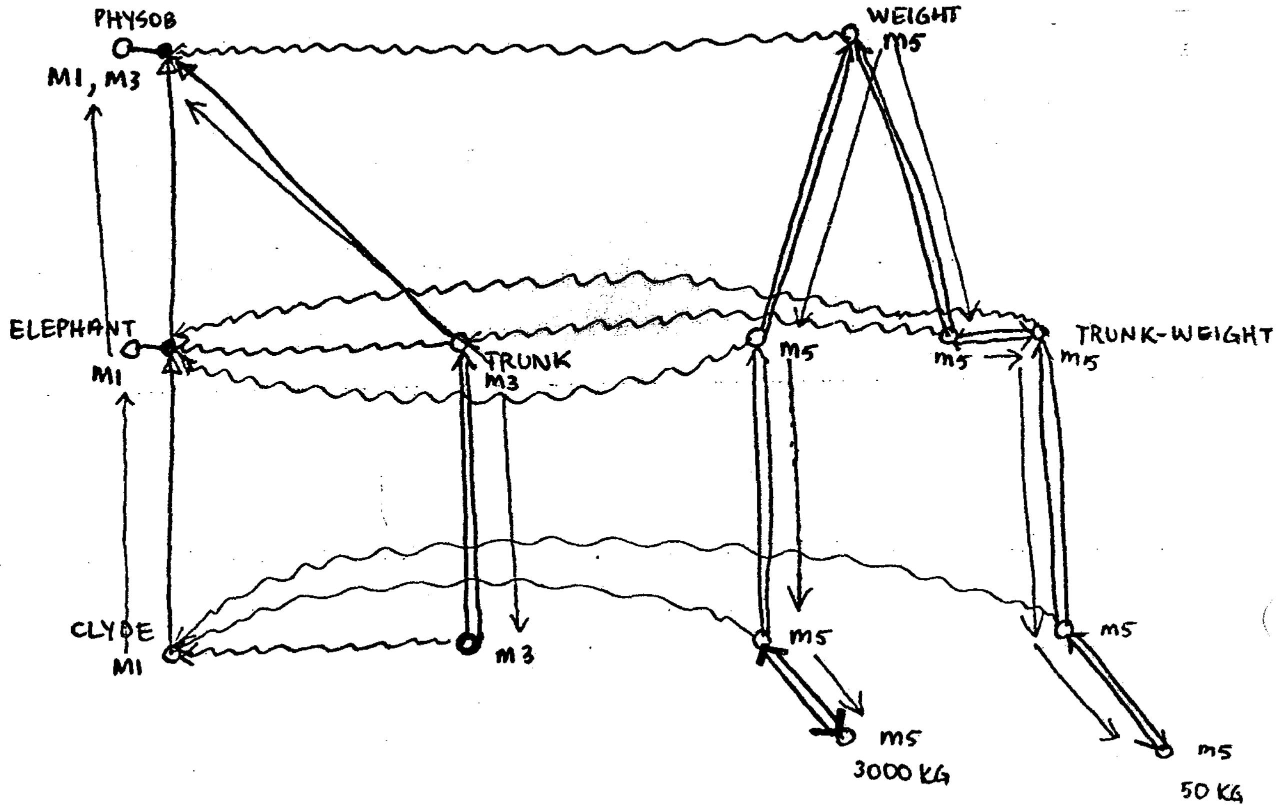


FIGURE 22, 23: COPY CONFUSION IN ACTION.

in a parallel manner.[28] Fahlman was unable to solve it in an elegantly. In fact, since the conditions which cause copy confusion occur so frequently, one wonders whether marker-passing is indeed the panacea which Fahlman claims it to be.

[28]Fahlman, p. 111

We have seen that despite the fact that NETL was designed to run on a parallel network machine, and LMS on a serial machine, the two languages are very similar. The form of the representation of even very complex kinds of knowledge is nearly identical in the two formalisms. The similarities are most evident in the discussion of link types. LMS and NETL both use a small number of primitive links from which they construct all other relationships between nodes. In addition, the (semantic) function of these links are, for the most part, identical in the two systems.

The two languages differ in some ways. The operation of specialization, which plays a key role in the representation of concepts in LMS, has no real counterpart in NETL. Each language contains smaller constructs (e.g. *EVERY and *OTHER nodes in NETL, *FUNCTION-TIEs and *SPECIES-TIEs in LMS) for which there is no corresponding convention in the other. The representation of cancellation, especially, differs greatly between the two.

Despite these differences, the set of conventions for representing knowledge in either system could easily be transported to a different type of machine with no loss in power of expression.

ANIMAL-WORLD DATA BASE

```
(declare-attachment-relation 11 #has #h nil)

[world=(summum-genus*i ")]
[real-world=(world*i ")]
[state-of-existence=(summum-genus*i ")]
[non-existent=(state-of-existence*s ")]
[physob=(summum-genus*i ") #c summum-genus  #rh real-world]
  [non-living-thing=(physob*s ") #c physob]
  [living-thing=(physob*s ") #c physob]
    [plant= (living-thing*s ")  #c living-thing]
    [animal=(living-thing*s ")  #c living-thing]
    [virus=(living-thing*s ")  #c living-thing]
    [bacterium=(living-thing*s ") #c living-thing]
[cell-num=(living-thing*t ")]
[(cell-num*s "multi-celled-lt") #c living-thing]
[(cell-num*s "single-celled-lt") #c living-thing]
[(cell-num*s "subcellular-lt") #c living-thing]
[cell=(physob*i ") #c physob #rh real-world]
[nucleus=(physob*i ") #c physob #rh cell]
[(cell*t "body-cell") #c cell #RH (cell-num*s multi-celled-lt)]
[living-place=(living-thing*t ")]
  [water-dwelling-lt=(living-place*s ") #c living-thing]
  [land-dwelling-lt=(living-place*s ") #c living-thing]
  [amphibious-lt=(living-place*s ") #c living-thing]
    [marine=(water-dwelling-lt*s ") #c water-dwelling-lt]
    [aquatic=(water-dwelling-lt*s ") #c water-dwelling-lt]
[flyer=(living-thing*i ") #c living-thing]
[spatial-area=(summum-genus*i ") #rh real-world #c summum-genus]
  [habitat=(spatial-area*i ") #c spatial-area #rh living-thing]
    [coastal-area=(spatial-area*s ") #c spatial-area]
    [land-area=(spatial-area*s ") #c spatial-area]
    [water-area=(spatial-area*s ") #c spatial-area]
    [air-area=(spatial-area*s ") #c spatial-area]
    [vacuum-area=(spatial-area*s ") #c spatial-area]
    [fresh-water-area=(water-area*s ") #c water-area]
    [ocean-area=(water-area*s ") #c water-area]
  [(habitat*r land-dwelling-lt) #rh land-dwelling-lt  #c land-area]
  [(habitat*r water-dwelling-lt) #rh water-dwelling-lt  #c water-area]
  [(habitat*r marine) #rh marine  #c ocean-area]
  [(habitat*r aquatic) #rh aquatic  #c fresh-water-area]

[virus #c (cell-num*s "subcellular-lt")]
  [(virus*s "herpes")]
[bacterium #c (cell-num*s "subcellular-lt")]
  [(bacterium*s "e-coli")]
[green-plant=(plant*s ") #c plant (fset*i green-plant)]
  [fungus=(plant*s ") #c plant (fset*i fungus)]
    [(fungus*s "slime-mold")]
[(physob*i "chloroplast") #c physob #rh green-plant]
  [cabbage=(green-plant*s ") #c green-plant #c multi-celled]
[single-celled-animal=(animal*s ") #c animal single-celled-lt]
[protozoan = single-celled-animal]
  [amoeba=(protozoan*s ") #c protozoan]
  [paramecium=(single-celled-animal*s ") #c single-celled-animal]
[multi-celled-animal=(animal*s ") #c animal multi-celled-lt]
[metazoan = multi-celled-animal]
  [(metazoan*s "coelenterate") #c metazoan]
  [worm=(metazoan*s ") #c metazoan]
  [mollusc=(metazoan*s ") #c metazoan]
  [arthropod=(metazoan*s ") #c metazoan]
  [echinoderm=(metazoan*s ") #c metazoan]
  [chordate=(metazoan*s ") #c metazoan]
    [(worm*s "platyhelminth")]
    [(worm*s "nematode")]
    [(worm*s "annelid")]
    [univalve=(mollusc*s ") #c mollusc]
    [bivalve=(mollusc*s ") #c mollusc]
    [cephalopod=(mollusc*s ") #c mollusc]
        [snail=(univalve*s ") #c univalve]
          [land-snail=(snail*s ") #c snail land-dwelling-lt]
          [water-snail=(snail*s ") #c snail]
  [mollusc #c water-dwelling-lt]
        [(bivalve*s "clam")]
        [(bivalve*s "oyster")]
        [(bivalve*s "scallop")]
        [(bivalve*s "mussel")]
        [octopus=(cephalopod*s ") #c cephalopod]
        [squid=(cephalopod*s ") #c cephalopod]
        [nautilus=(cephalopod*s ") #c cephalopod]
  [shell-bearer=(metazoan*s ") #c metazoan]
[shell=(non-living-thing*i ") #c non-living-thing #rh shell-bearer]
  [mollusc #c shell-bearer]
[(shell*r cephalopod) #c shell #m non-existent #rh cephalopod]
  [nautilus #c shell-bearer]
[(shell*r nautilus) #c shell #rh nautilus]

[vertebrate=(chordate*s ") #c chordate]
[mammal=(vertebrate*s ") #c vertebrate]
[elephant=(mammal*s ") #c mammal]
```

```
[clyde = (elephant*i "clyde") #c elephant]               ;create a node for
[CLYDE = (ELEPHANT*I ") #C ELEPHANT]                      ; the elephant Clyde

(characterize [clyde] [cabbage])                          ;try to tell it that
                                                          ;Clyde is a cabbage

[CLYDE] cannot be a [CABBAGE]
[ANIMAL] and [PLANT] are disjoint classes

(can-be [clyde] [mollusc])                                ;can Clyde be a Mollusc?

 NO -
[CHORDATE] and [MOLLUSC] are disjoint classes

(has-any [clyde] [shell])                                 ;does Clyde have a shell?

NIL

(has-any [snail] [shell])                                 ;does a snail have a shell?

T

(has-any [octopus] [shell])                               ;does an octopus have one?

NIL                                                       ;the cancellation worked.

(has-any [nautilus] [shell])                              ;does a nautilus have one?

T                                                         ;the reassertion worked.

(can-be [paramecium] [protozoan])                         ;is a paramecium a protozoan?

 YES                                                      ;the intersection worked.
```

```
;---------------------------------------------------------------------------
        ;CLASHP returns T if (node1) and (node2) are members of mutually
        ;exclusive classes, else returns NIL.

(defun clashp (node1 node2)
       (and (member (specializer (least-common-superior node1 node2))
                    '([[*S] = (PRIMITIVE-TIE "SPECIES")]
                      [[*I] = (PRIMITIVE-TIE "INDIVIDUAL")]))
            T))

;---------------------------------------------------------------------------
        ;CHARACTERIZE attaches (node2) as a characterization of (node1),
        ;after first checking that the proposed link does not
        ;       -create a clash
        ;       -create a directed loop of #CHARACTERIZATION links.
        ;If the attachment is made, (node2) is returned, else an error
        ;message is printed.

(defun characterize (node1 node2)
        (cond ((not (clashp node1 node2))
               (cond ((member-of-class node2 node1)
                      (terpri)
                      (princ '|error: creates directed loop of #C links|)
                      (terpri))
                     (t (make-attachment [#C] node2 node1))))
              (t (terpri)
                 (princ node1)
                 (princ '| cannot be a |)
                 (princ node2)
                 (terpri)
                 (clash-errmsg node1 node2))))
;---------------------------------------------------------------------------
        ;CAN-BE returns T if (node1) can be characterized by (node2),
        ;else an error message is printed.

(defun can-be (node1 node2)
        (cond ((not (clashp node1 node2))
               (terpri)
               (princ '| YES|)
               (terpri))
              (t
               (terpri)
               (princ '| NO -|)
               (terpri)
               (clash-errmsg node1 node2))))
```

```
;-----------------------------------------------------------------
        ;FIND-SPLIT is used when a clash has been detected, to find
        ;the point at which the error occurred, i.e. the disjoint
        ;classes of which (node1) and (node2) are members.

(defun find-split (node1 node2)
       (let* ((split-set (for x being immediate-inferiors of
                              (least-common-superior node1 node2)
                                   collect x))
              (int1 (intersect split-set node1))
              (int2 (intersect split-set node2)))
             (list int1 int2)))

;-------------------------------------------------------------------
        ;CLASH-ERRMSG prints an error message at the terminal indicating
        ;the disjoint classes which caused the error.

(defun clash-errmsg (node1 node2)
       (let ((intersections (find-split node1 node2)))
            (terpri)
            (princ (car intersections))
            (princ '| and |)
            (princ (cadr intersections))
            (princ '| are disjoint classes |)
            (terpri)))

;-------------------------------------------------------------------
        ;INTERSECT finds the intersection between (set) and the set consisting
        ;of (node) and its superiors. The intersection always contains
        ;exactly one member, which is returned.

(defun intersect (set node)
        (for n being node and its superiors do
            (cond ((member n set) (return n)))))
```

```
;-----------------------------------------------------------------------
        ;MEMBER-OF-CLASS determines if (object) is a member of the class
        ;(class), and returns T or NIL accordingly.

(defun member-of-class (object class)
   (cond ((equal object class) T)
         (t (for x being each [#C] of object first-time
                 (member-of-class x class)))))

;-----------------------------------------------------------------------
        ;HAS-ANY returns T if (node) has (object), or anything of (object)'s
        ;class, attached to it as a role, else returns NIL.

(defun has-any (node object)
   (for x being each [#HAS] of node
     do
       (cond ((member-of-class x object)
              (cond ((null (look-for-attachment [#M] [non-existent] x))
                     (return 'T))
                    (t (return 'NIL)))))
     finally                                  ; If we fall through:
       (return (for y being each [#C] of node first-time
                  (has-any y object)))))
```

Bibliography

Fahlman, Scott E. *A System for Representing and Using Real-World Knowledge*, Massachusetts Institute of Technology AI Lab Technical Report 450, December 1977.

Hawkinson, Lowell. "XLMS Guide" (draft), February 1979.

Hawkinson, Lowell. *The Representation of Concepts in OWL*, Massachusetts Institute of Technology Project MAC Automatic Programming Group Internal Memo 17, July 1975.

Hewitt, Carl, Giuseppe Attardi, and Maria Simi. "Knowledge Embedding in the Description System OMEGA" (draft),Massachusetts Institute of Technology AI Lab Working Paper, 1979.

Martin, William A. *Philosophical Foundations for a Linguistically Oriented Semantic Network* (draft), March 1979.

Smith, Brian C. "Levels, Layers, and Planes: The Framework of a System of Knowledge Representation Semantics," M.S. Thesis, Massachusetts Institute of Technology 1978.