MASSACHUSETTS INSTITUTE OF TECHNOLOGY

ARTIFICIAL INTELLIGENCE LABORATORY

WORKING PAPER 170 OCTOBER 1978

FRAME-BASED KNOWLEDGE REPRESENTATION
(progress report)

LUC STEELS

The paper introduces a language for representing knowledge in a declarative form. With this language it is possible to define knowledge about a certain domain by introducing a number of concepts and by specifying their interrelations.
The paper is meant to be an informal introduction to the language. We present the available constructs, describe their meaning and present a number of examples.
In other papers (currently in preparation) we will give a formal semantics of the language, introduce the inference theory and discuss a possible procedural embedding.

This report describes research done at the Artificial Intelligence Laboratory of the Massachusetts Insitute of Technology. The author was sponsored by the Institute of International Education on an ITT-fellowship.

CONTENTS

0. PREFACE

WHY WE MAKE LANGUAGES

It is sometimes argued that since Turing invented a formalism to specify every computable process, there is no point in introducing new formalisms for computation because they will not be able to define a process that could not already be defined by a Turing machine.

This argument is invalid because we do not design new languages in order to extend the class of computations they can perform in theory. What we do in making a new language is to perform executional *abstraction* (Dijkstra,1976) over a class of computational processes and to provide a syntactic mechanism for expressing that a certain process is as a member of that class. The resulting abstractions are the new primitives of our formalism. We do this because the appropriate primitives will reduce the complexity of the programs, clarify program structure, improve modularity, etc. Generally speaking they will lead to an increased productivity in programming technology: We will be able to write programs more easily and more effectively and thus do more (in a practical sense).

But in AI the relevance of postulating computational primitives goes beyond these productivity concerns. Because AI languages are advanced as a set of primitive mechanisms that underlie all intelligent processes, they have a theoretical significance: They make an empirical statement about the class of computational processes which can do tasks performed by systems with a natural intelligence. The nature of these statements is different from the introduction of a constraint on the descriptive mechanisms of the formalism thereby effectively reducing the class of possible computational processes (e.g. from the power of a Turing machine to one of a finite state machine). But this does not mean that they have no theoretical value or empirical content.

As with all empirical generalizations it is not possible to prove (from the start) whether the generalizations will be adequate and valid or not. The only thing we can do is try to falsify a proposed set of generalizations. (see Popper,1974). We falsify a language by showing that the primitives are an obstacle to program development, e.g. that a certain mechanism has to be circumvented in order to perform certain tasks, that a certain mechanism with widespread use is missing, that a certain mechanism invites dangerous programming practice, etc. A nice example of such a falsification effort is to be found in McDermott and Sussman (1972).

WHY WE MAKE KNOWLEDGE REPRESENTATION LANGUAGES

Similarly it could be argued that there is no point in developing new knowledge representation languages because in the end they all come down to list structures anyway. A mere repetition of the arguments given in favour of new programming languages invalidates this reasoning.

What we do in constructing a knowledge representation language is to make an abstraction over a class of representational structures and introduce a syntactic mechanism to represent that abstraction. Again there is a practical gain: access functions that are valid for the whole class have to be written only once, modularity is improved, etc. And again, it should be noted that an AI knowledge representation language makes certain generalizations about representational structures used by intelligent processes and is therefore a fundamental and profound contribution to cognitive science.

WHAT THE GOALS OF THIS RESEARCH ARE

What I would like to have is a system where the user provides information about situations, objects, relations, plans, etc., by specifying their aspects and the constraints on those aspects, such that (s)he is then able to confront the system with a particular situation (object, relation, plan) and request that the system recognize this in terms of known prototypical situations. And also that (s)he is able to ask the system to construct particular situations that satisfy a certain number of initial assumptions and the constraints which hold in general for the specified class of situations.

The descriptions of the situations, their parts and constraints must be phrased in a knowledge representation language. In this paper we will introduce a language for that purpose. It is based on a number of abstractions that are generally known as the *frame theory* originally collected in Minsky (1974).

Of course it is not sufficient to simply propose a language that contains syntactic mechanisms for expressing certain abstractions. That is only the second step of our methodology (the first step is the recognition that there are generalizations to be made and the definition of what their nature is). The third step consists in actually defining the operational semantics of the constructs. That will be the subject of other papers.

1. SLOTS AND FRAMES

As a first approximation we can characterize declarative knowledge as consisting of (i) a number of concepts and (ii) a definition of each of these concepts in terms of other concepts. The concepts will enable us to describe a state of affairs and to recognize that a certain state of affairs holds. A definition of the relationships between the concepts will enable us to make predictions, complete partial descriptions, etc.
In what follows we will represent concepts and their interrelationships in terms of frames.

A *frame* is an information structure with a *frame-name* and a number of *slots*. A slot is the holder of information concerning a particular item called the *slot-filler*. The slot-fillers map onto the objects in the domain of discourse.
A frame expresses a concept involving the various entities that appear as slot-fillers of its slots. There is no limit on the number of slots in a frame.
Each slot has a unique relation to the whole. We call this relation *aspect*. Each aspect has an *aspect-name* which is unique within the frame.

We will often say "a frame for X" meaning a frame with frame-name X, and "the Y-slot of X" meaning a slot related by an aspect with aspect-name Y to a frame with frame-name X.

We now introduce two usages of frames and subsequently two notations.

1.1. FRAME USAGES

1.1.1. FRAMES AS CONCEPT-CARRIERS

When we use a frame to collect all there is known about a certain concept we will call a frame a *concept-carrier*. If so, a frame is represented in terms of a list-structure as follows

```
(<frame-name>
   (WITH <aspect-name>)
   (WITH <other-aspect-name>) ...  )
```

The ordering of the aspect specifications has no meaning.

For example

```
(WALK
   (WITH WALKER)
   (WITH DISTANCE))
```

is a frame with frame-name WALK and with two slots which are related to the whole by aspects with aspect-name WALKER and DISTANCE respectively. This frame is meant to set up a structure for expressing knowledge about the concept of WALK. The structure is such that it provides room for two objects. One that is related to WALK by the WALKER-aspect, one that is related to WALK by the DISTANCE-aspect.

Note that there is no syntactic representation of the slot nor of the slot-filler. Note also that the frame as a whole does NOT have an interpretation in terms of objects in the domain. For example if we would want to include the possibility of talking about the walk-activity itself, i.e. about a member of the class of objects that are walk-activities, we would need another slot. If we use the aspect-name BEING for the aspect by which this slot is related to the rest of the frame, we would have

```
(WALK
  (WITH BEING)
  (WITH WALKER)
  (WITH DISTANCE))
```

Finally note that one could view a concept-carrier as a lambda-expression in the sense of Church (1951). The frame-name corresponds to the function-name from which lambda-abstraction has been made and the aspects correspond to the arguments that have been factored out. Thus the WALK frame would correspond to

```
((LAMBDA BEING WALKER DISTANCE) (WALK BEING WALKER DISTANCE))
```

1.1.2. FRAMES AS DESCRIPTIONS

When we use a frame to specify that a certain slot, which is already known to be functioning as the aspect of another frame, is functioning as one of its own aspects, we call a frame a description and we say that the description *holds for* each of the potential slot-fillers of the slot.

A description is different from a concept-carrier in two respects:

(i) A description can be viewed as an *instantiation* of a concept-carrier in the sense that the class of potential slot-fillers of each of the slots in the description is a subset of the class of potential slot-fillers of their corresponding slots in the concept-carrier.

(ii) One aspect, called the *bridge*, is singled out. The bridge is the aspect from which the frame is approached. Talking in terms of the lambda-calculus we could say that the bridge is the argument still further factored out by a second lambda-abstraction.

The general syntactic form of a description is

```
(<description-indicator> <bridge-name> <frame-name>
        (WITH <other-aspect-name> ) ...).
```

What description-indicator is used depends on the type of description.

Descriptions are *attached to* the slots which contain the slot-filler for which the description holds. They specify *constraints* on those slots. Attachment is represented by writing the description after the aspect-name in the frame.

1.1.2.1. DESCRIBERS

The first type of descriptions will be called *describers*. They represent the fact that each of the slot-fillers in the slot to which they are attached is valid as a filler of the bridge of the frame in the description.

We represent describers by using the description-indicator DESCRIBED-AS. For example, given the frame for WALK earlier, we could form a description by making the WALKER aspect the bridge:

```
    (DESCRIBED-AS WALKER WALK
                (WITH DISTANCE)
                (WITH BEING))
```

by making the BEING aspect the bridge:

```
    (DESCRIBED-AS BEING WALK
                 (WITH WALKER)
                 (WITH DISTANCE))
```

or by making the DISTANCE aspect the bridge:

```
    (DESCRIBED-AS DISTANCE WALK
                 (WITH WALKER)
                 (WITH BEING)).
```

Now we construct an example of attachment. Suppose we have in addition to the WALK-frame a frame for PERSON with a slot for the BEING, as in

```
(PERSON
    (WITH BEING ))
```

then we could attach a description to the WALKER-slot in the WALK-frame as follows,

```
(WALK
   (WITH WALKER (DESCRIBED-AS BEING PERSON))
   (WITH DISTANCE )
   (WITH BEING))
```

Because WALKER and BEING are now two aspects coming out of the same slot, each potential slot-filler of the WALKER-slot of WALK will be a potential slot-filler of the BEING-slot.

When we have to extract knowledge from the fact that a certain object plays a certain role in a situation,, i.e. fills a certain aspect in a frame, we may "infer" the descriptions attached to the slot introducing that role as additional properties of the slot-filler. In such an application the descriptions function as the *consequent of an implication* where the antecedent is made up by the rest of the frame.
Returning to our example, if we know that a slot-filler is described-as the walker of a walk, then we know that it is described-as the being of a person.

A consequence of the instantiation relation between a description and its concept-carrier is that all descriptions which are attached to the slots in the concept-carrier are ipso facto attached to the slots of a description. We say that those descriptions are *inherited* by the description.

For example, if the frame of PERSON would have a description attached to the BEING slot, as in

```
  (PERSON
     (WITH BEING
          (DESCRIBED-AS HAVER AGE)))
```
then if we know that a slot-filler is described-as the walker of a walk, we know that it is described-as the haver of an age.

It is allowed to attach descriptions to any kind of slot, whether introduced by a concept-carrier or a description and there is no limit on the degree of embedding.
Thus
```
(JOHN
  (WITH BEING
        (DESCRIBED-AS WALKER WALK
              (WITH DISTANCE
                    (DESCRIBED-AS DISTANCE WALK
                          (WITH WALKER (DESCRIBED-AS BEING JOHN)))))))
```
contains 4 frames embedded into each other.

1.1.2.2. DEFINERS

The second type of descriptions are called *definers*. They are equal to describers in the sense that they represent the fact that every potential slot-filler of the slot to which the description is attached is said to be a filler of the bridge of the description. But they are stronger than describers in the sense that when we know of a slot-filler that all the definers attached to its slot hold, we may conclude that this slot-filler is a filler of the aspect that introduced the slot.

For example suppose we have a frame for HAVING-FOUR-LEGS with an aspect called HAVER for the object having the four legs. Suppose furthermore that we want to attach to the HAVER-slot the information that this object is also a filler of the BEING-slot of a QUADROPED. Then if we do this as follows
```
  (HAVING-FOUR-LEGS
        (WITH HAVER (DEFINED-AS BEING QUADROPED)))
```
we know that whenever an object is known to be described-as the HAVER of a HAVING-FOUR-LEGS we also know that it is the BEING of a QUADROPED. And vice-versa !

It could be argued that there is no need for this second type of descriptions because after all if we have two frames
```
(HAVING-FOUR-LEGS
        (WITH HAVER (DESCRIBED-AS BEING QUADROPED)))
```
and
```
(QUADROPED
        (WITH BEING (DESCRIBED-AS HAVING FOUR-LEGS)))
```
we know just as much.
But this argument bypasses the principle of organizational control: When we have to recognize that a concept is applicable or confirm that the use of a certain concept is valid, we will check whether the definers attached to each of the slots are valid for the respective slot-fillers. If they

all are, we can conclude that the concept is applicable. In contrast, even if we could show that all describers hold, we still would not be able to conclude that the concept is applicable. And because there is *in principle* no way to get from a description attached to a slot to the frame having the slot, we would be unable to perform certain recognition tasks.

So, describers are only useful when we are forward chaining from known information towards a certain goal. Definers are useful when we are forward chaining but even more so when we are backward chaining.

1.3. TYPES OF FRAMES

1.3.1. ABSTRACT AND INDIVIDUAL CONCEPTS

It is widely accepted (see e.g. Strawson, 1972) that there is a distinction between concepts which have a class interpretation and concepts which refer uniquely to a simple object in the domain. These concepts are usually called *individual concepts* (following Carnap,1947) or denoting concepts (following Russell,1940). Where the other ones are called *abstract* or *generic concepts*.

The difference between a frame for arbitrary concepts and a frame for an individual concept lies in the conclusions that follow from a mismatch. Suppose (DESCRIBED-AS BEING 1) is given and we have to show that (DESCRIBED-AS BEING 2), where 1 and 2 represent individual concepts, then we can conclude right away the negation of (DESCRIBED-AS BEING 2) because the frame-names do not match. However if (DESCRIBED-AS BEING 1) is given and we have to show that (DESCRIBED-AS BEING 2) holds, where 1 and 2 are abstract concepts, then nothing can be concluded from the mismatch between 1 and 2 and unless the fact that 1 is different from 2 is stored or can be deduced from explicit facts, we would never be able to detect that 1 and 2 are different. This is the motivation for having a type system for concepts and systems without this construct (such as MERLIN (cf. Moore and Newell,1973)) will lose.

In order to express the fact that a frame deals with an individual or an abstract concept we will introduce a syntactic indicator as first element of a concept-carrier, as in

```
(<type-indicator> <frame-name>
        (WITH <aspect-name> ...)  ....)
```

where type-indicator is one of INDIVIDUAL-CONCEPT, ABSTRACT-CONCEPT.

For example

```
(INDIVIDUAL-CONCEPT 1
  (WITH BEING (DESCRIBED-AS BEING NUMBER)))
```

or

```
(ABSTRACT-CONCEPT HUMAN
  (WITH BEING (DESCRIBED-AS BEING MORTAL)))
```

Note that a frame for an individual-concept does NOT represent an individual. The individual itself is a slot-filler of one of the slots (e.g. the BEING-slot in the frame for 1).

1.3.2. PROTOTYPES AND EXEMPLARS

A further distinction often made in frame-based representation systems has to do with the notion of *prototype* (also called stereotype (Hewitt,1976) or typical member) and *exemplar* (Winograd,1978).

Whereas the abstract concept contains everything that is always true for every possible slot-filler in the range of a concept, the prototypical concept contains everything which is true for the typical slot-filler in the range of the concept. For example we could have a frame for an abstract concept of a lion (in which we store for example that a lion is an animal) and then a frame for the prototypical concept of the typical lion (in which we store for example that a lion has four legs). When reasoning over an individual lion, the descriptions attached to the typical lion would function as default properties, as such attachable to the individual lion.

An exemplar is a concept which has the function of a prototypical concept but is at the same time an individual from the domain of discourse. It is the individual that serves at a given time as THE example, and therefore as a source of information for default properties.

I hesitate to introduce two new types for prototypical and exemplaric concepts. It seems that many functions of these types can be realized using primitives to be introduced later (such as the viewed-as operator of section 6). But further research is needed to sort out the weaknesses of the possible solutions.

Note that the four types reflect stages of learning a concept. First an individual is recognized as having a number of properties, such as having manes, four legs, being ferocious, etc. Second this individual is promoted to an exemplar by making it a model for other individuals. Third abstraction is made from the individual functioning as exemplar and the model becomes a prototype. Fourth all necessary properties are abstracted away and collected in the abstract concept.

2. THE CONNECTIVES

2.1. THE USUAL CONNECTIVES

It is allowed to construct more complex descriptions by combining descriptions with the logical connectives NOT, AND, OR, and XOR (= exclusive or). These connectives have their usual meaning.

For example

```
(NOT (DESCRIBED-AS BEING HUMAN))
```

means that the potential slot-fillers to which this description will be applied cannot be fillers of the BEING aspect of a HUMAN frame.

```
(INDIVIDUAL-CONCEPT JOHN
  (WITH BEING
       (AND (DESCRIBED-AS PARENT PARENT-CHILD-RELATION
                     (WITH CHILD (DESCRIBED-AS BEING MARY)))
            (DESCRIBED-AS SINGER BAND
                     (WITH BEING (DESCRIBED-AS BEING 'THE-ZOMBIES'))))))
```

means that every filler of the BEING slot in the frame of JOHN is the filler of the PARENT slot in a PARENT-CHILD-RELATION frame and the filler of a SINGER slot in a BAND frame.
Note that with definers, all descriptions must be true for the concept-carrier to hold.
For example

```
(ABSTRACT-CONCEPT BACHELOR
  (WITH BEING
       (AND (DEFINED-AS BEING MALE)
            (DEFINED-AS BEING SINGLE))))
```

implies that we can only conclude that some object is the filler of the BEING-slot in a BACHELOR frame if this object is the filler of the BEING-slot in a MALE frame AND the filler of the BEING-slot in a SINGLE frame.

```
(INDIVIDUAL-CONCEPT JOHN
  (WITH BEING
       (DESCRIBED-AS PARENT PARENT-CHILD-RELATION
            (WITH CHILD (OR (DESCRIBED-AS BEING MARY)
                            (DESCRIBED-AS BEING PETER))))))
```

means that the filler of the child slot introduced by the parent-child-relation is either a filler of the being slot of a MARY frame or a filler of the being slot of a PETER frame or both.

```
(INDIVIDUAL-CONCEPT JOHN
   (WITH BEING
        (DESCRIBED-AS PARENT PARENT-CHILD-RELATION
             (WITH CHILD (XOR (DESCRIBED-AS BEING MARY)
                              (DESCRIBED-AS BEING PETER))))))
```

means that the filler of the child slot introduced by the parent-child-relation is either a filler of the being slot of MARY or a filler of the being slot of PETER but not both.

Now we introduce some new connectives.

2.2. A CONNECTIVE FOR WEAK DESCRIPTIONS.

Suppose we want to say that a certain description holds for a certain slot-filler but not necessarily so. This may be useful in default reasoning. We know that it is usually the case that a certain property holds for a class of individuals and we want to attach to a certain individual the information that it is not unlikely that it has this property

To express this we introduce a connective that is a weaker form of negation. We use the indicator NOT-NECESSARILY, as in

```
(NOT-NECESSARILY (DESCRIBED-AS HAVING 4-LEGS))
```

Note that

```
(NOT-NECESSARILY (NOT-NECESSARILY (DESCRIBED-AS HAVING 4-LEGS)))
```

is identical to

```
(NOT-NECESSARILY (DESCRIBED-AS HAVING 4-LEGS))
```

Note also that if

```
(ABSTRACT-CONCEPT HUMAN
  (WITH BEING (DESCRIBED-AS BEING MORTAL)))
```

and (DESCRIBED-AS BEING MORTAL) is known, we can conclude that

```
(NOT-NECESSARILY (NOT (DESCRIBED-AS BEING MORTAL)))
```

is true.

The NOT-NECESSARILY connective may also serve as a primitive for building up abstract concepts representing modal notions. (Hughes and Creswell,1973)

2.3. A CONNECTIVE FOR MULTIPLE DESCRIPTIONS

We now add a second new connective. This connective combines a number of descriptions that all hold for the slot to which they are attached but each of them is on its own valid as a trigger for the whole. This is useful if we have a description from *various viewpoints* (cf. Moore and Newell, 1973). It is assumed that each description is in itself a sufficient characterization of the potential slot-fillers.

We call this new connective MULTIPLE-AND. Note that a truth-table of a MULTIPLE-AND would be exactly the same as a truth-table for AND. Only the usage differs. Moreover the difference in usage crops up only with definers. For example,

```
(INDIVIDUAL-CONCEPT JOHN
   (WITH BEING
        (MULTIPLE-AND
            (DEFINED-AS PARENT PARENT-CHILD-RELATION
                 (WITH CHILD (DESCRIBED-AS BEING MARY)))
            (DEFINED-AS SINGER BAND
                 (WITH NAME (DESCRIBED-AS BEING "THE-ZOMBIES"))))))
```

means that the filler of the being slot is defined-as the parent of a parent-child-relation AND the singer of a band. If we are able to show however that this filler is described-as one of the two, the concept in the frame and the other description may be assumed to hold also.

3. THE CO-REFERENTIAL

Sometimes we want to talk about the filler of a slot, in terms of it being the filler of a slot. This happens for example when we want to establish an explicit link between two slots such that all descriptions which are attached to one slot are also known to be attached to the other and vice-versa.

To deal with this we introduce a new language construct called the *co-referential*. It consists of a pair (FILLED-BY <referring-name>), which is attached to the slot which contains the filler by writing it after the aspect-name introducing the slot (i.e. before the constraints itself). When referring to the same slot-filler somewhere else, we simply use the same pair.

Here is an example. Suppose we have a parent-child relation, as

```
(ABSTRACT-CONCEPT PARENT-CHILD-RELATION
      (WITH PARENT)
      (WITH CHILD))
```

and we want to construct a frame for PARENT. Then this could be done with the following frame:

```
(ABSTRACT-CONCEPT PARENT
      (WITH BEING
            (DEFINED-AS PARENT PARENT-CHILD-RELATION
                  (WITH CHILD (FILLED-BY THE-CHILD))))
      (WITH CHILD
            (FILLED-BY THE-CHILD)))
```

In this case we say that the CHILD-slot of the PARENT frame is co-referential with the CHILD-slot of the PARENT-CHILD-RELATION frame.

Now if we say

```
(INDIVIDUAL-CONCEPT JOHN
    (WITH INDIVIDUAL
        (DESCRIBED-AS BEING PARENT
              (WITH CHILD (DESCRIBED-AS BEING MARY)))))
```

then the description attached to the individual slot of JOHN is equivalent to

```
(DESCRIBED-AS PARENT PARENT-CHILD-RELATION
      (WITH CHILD (DESCRIBED-AS BEING MARY))).
```

Notice that because we do not make a distinction between the moment where the referring name gets "bound" to a particular slot-filler, and the moment where it is "used", constraints on the slot-filler may come from any point where a co-referential is attached.
Thus, using the PARENT frame, when given

```
   (DESCRIBED-AS BEING PARENT
         (WITH CHILD (DESCRIBED-AS BEING MARY)))
```

we can infer

```
   (DESCRIBED-AS PARENT PARENT-CHILD-RELATION
         (WITH CHILD (DESCRIBED-AS BEING MARY))).
```

But also, when given

```
  (DESCRIBED-AS PARENT PARENT-CHILD-RELATION
         (WITH CHILD (DESCRIBED-AS BEING MARY)))
```

we can infer

```
  (DESCRIBED-AS BEING PARENT
         (WITH CHILD (DESCRIBED-AS BEING MARY))).
```

As long as it occurs once as first expression on a slot, the co-referential may be used everywhere where a description is used (also with connectives).

4. DECLARATIVE CONDITIONALS

4.1. THE WHEN-CONSTRUCT

We may want to specify constraints on possible slot-fillers of a frame inside (i.e. under the scope of) another slot. This happens for example if we are attaching descriptions to a certain slot and we want to make a distinction based on different properties of another slot.

To express this we introduce a *declarative conditional.* The syntax of this construct is as follows. First we write the syntactic indicator WHEN, then the referring-name of the slot-filler that is being re-introduced and then we construct a list of description pairs with the left pair the condition and the right pair the description which holds for the slot-filler of the slot to which the WHEN-construct is attached. In other words

```
(WHEN <referring-name>
   (<condition-1> <description-1>)
   ...
   (<condition-n> <description-n>))
```

The meaning is such that as soon as a condition holds for the potential slot-fillers in the slot referred to by the referring-name, the corresponding description holds for the potential slot-fillers of the slot to which the WHEN-construct is attached.

For example suppose we have frames for LIST and ATOM as follows

```
(ABSTRACT-CONCEPT LIST
  (WITH CONS)
  (WITH CAR)
  (WITH CDR))
(ABSTRACT-CONCEPT ATOM
  (WITH BEING))
```

then we can give a declarative definition of append as follows

```
(ABSTRACT-CONCEPT APPEND
  (WITH FIRST (FILLED-BY THE-FIRST))
  (WITH SECOND (FILLED-BY THE-SECOND))
  (WITH RESULT
        (WHEN THE-FIRST
              ((DESCRIBED-AS BEING NIL) (FILLED-BY THE-SECOND))
              ((NOT (DESCRIBED-AS BEING NIL))
                     (DEFINED-AS CONS LIST
                          (WITH CAR
                                (DESCRIBED-AS CAR LIST
                                      (WITH CONS (FILLED-BY THE-FIRST))))
                          (WITH CDR
                                (DESCRIBED-AS RESULT APPEND
                                      (WITH FIRST
                                            (DESCRIBED-AS CDR LIST
                                                  (WITH CONS (FILLED-BY THE-FIRST))))
                                      (WITH SECOND
                                            (FILLED-BY THE-SECOND)))))))))
```

which can be compared with the procedural definition

```
(DEFUN APPEND (X Y)
   (COND ((NULL X) Y)
         (T (CONS (CAR X)
                  (APPEND (CDR X) Y)))))
```

A further refinement consists in allowing as condition the indicator ELSE which is equivalent to the conjunction of the negation of all the other conditions. For the append-example this leads to

```
(ABSTRACT-CONCEPT APPEND
  (WITH FIRST (FILLED-BY THE-FIRST))
  (WITH SECOND (FILLED-BY THE-SECOND))
  (WITH RESULT
        (WHEN THE-FIRST
              ((DESCRIBED-AS BEING NIL) (FILLED-BY THE-SECOND))
              (ELSE
                    (DESCRIBED-AS CONS LIST
                         (WITH CAR
                              (DESCRIBED-AS CAR LIST
                                   (WITH CONS (FILLED-BY THE-FIRST))))
                         (WITH CDR
                              (DESCRIBED-AS RESULT APPEND
                                   (WITH FIRST
                                        (DESCRIBED-AS CDR LIST
                                             (WITH CONS (FILLED-BY THE-FIRST))))
                                   (WITH SECOND
                                        (FILLED-BY THE-SECOND)))))))))
```

4.2. THE REPRESENTATION OF DEFAULTS

It may happen during reasoning that nothing is known about a certain condition in a conditional, i.e. we cannot show whether it is valid or not. In such a case the retrieval of the description for the slot would come to an unfortunate halt. However in many cases it is possible to specify a so called *default* that would function as the normal case in such moments of uncertainty. When one of the other conditions would become satisfied, we could retract the original assumption and all facts depending on it (as in Doyle,1978). An alternative implementation would make use of the NOT-NECESSARY connective introduced earlier.

Syntactically speaking we will represent all this by using the indicator UNLESS instead of WHEN and retain completely the rest of the syntax of a conditional expression.

Here is an example

```
(INDIVIDUAL-CONCEPT JOHN
  (WITH BEING (FILLED-BY MYSELF)
        (UNLESS MYSELF
           ((DESCRIBED-AS LOCATED AT-WORK) (DESCRIBED-AS LOCATED AT-WORK))
           (ELSE (DESCRIBED-AS LOCATED AT-HOME)))))
```

The description attached to the BEING-slot says that unless John is known to be located at his work, it may be assumed that he is located at home.

5. DECLARATIVE QUOTE AND UNQUOTE.

Suppose we want to set up a description for an act of predication, as in "John is a person". We could introduce a frame for predications with (at least) three aspects: one for the subject (i.c. 'John'), one for the predicate or descriptor of the predication (i.c. 'being a person'), and for the truthvalue (i.c. 'being true') , as in

```
(INDIVIDUAL-CONCEPT PREDICATION-1
  (WITH BEING
        (DESCRIBED-AS BEING PREDICATION
              (WITH SUBJECT (DESCRIBED-AS BEING JOHN))
              (WITH DESCRIPTOR (DESCRIBED-AS BEING PERSON))
              (WITH TRUTHVALUE (DESCRIBED-AS BEING TRUE)))))
```

However we see immediately that this does not mean what we want to say. In particular the description attached to the descriptor-slot has to describe the fillers of that slot and because those fillers are to be descriptions and not persons, as this representation says, the frame is inappropriate.

What we need is a construct to prevent a description from performing its descriptive function. In other words we need a declarative quote. As syntactic representation we will use the pair ^<description>. A description consisting of the ^-sign followed by a description yields a new description whose extension (i.e. whose set of slot-fillers) contains the description following the sign.

Here is then a representation that correctly represents what we wanted to say

```
(INDIVIDUAL-CONCEPT PREDICATION-1
  (WITH BEING
        (DESCRIBED-AS BEING PREDICATION
              (WITH SUBJECT (DESCRIBED-AS BEING JOHN))
              (WITH DESCRIPTOR ^(DESCRIBED-AS BEING PERSON))
              (WITH TRUTHVALUE (DESCRIBED-AS BEING TRUE)))))
```

Whenever there is a way of quoting things, there may be a need for the reverse operation: unquote. For example, if we want to define the relation between the various slots in the predication frame in such a way that the truthvalue is true if the descriptor holds for the slot. In order to force a quoted description to be unquoted again we use the pair ˅<description>. Obviously ˅ ^ <description> is identical to <description>.

A frame for predication itself could then be defined as follows

```
(ABSTRACT-CONCEPT PREDICATION
   (WITH BEING)
   (WITH SUBJECT (FILLED-BY THE-SUBJECT))
   (WITH DESCRIPTOR (FILLED-BY THE-DESCRIPTOR))
   (WITH TRUTHVALUE
        (WHEN THE-SUBJECT
             (v(FILLED-BY THE-DESCRIPTOR) (DESCRIBED-AS BEING TRUE))
             (ELSE (DESCRIBED-AS BEING FALSE)))))
```

The declarative quote and unquote mechanism introduced here solves a number of representational puzzles having to do with opaque contexts (such as 'Pat knows the number of a certain safe'(McCarthy(1978), Martin(1978))), i.e.

```
(INDIVIDUAL-CONCEPT PAT
   (WITH BEING
        (DESCRIBED-AS HAVING KNOWLEDGE
             (WITH CONTENT
                  ^ (DESCRIBED-AS VALUE NUMBER
                           (WITH HAVING (DESCRIBED-AS BEING SAFE)))))))
```

modals, direct discourse, and the like. These puzzles are also solvable by introducing modal operators. However the intensional solution made possible here appears to be much more convenient (as illustrated in Montague,1970). Actually the ^ and v are reminiscent of the ^ and v in Montague (ibid.)

6. VIEWING-AS

Suppose we have the following representation problem. We want to say about a relation (e.g. the subset-relation) that it is a transitive relation. Which amounts to saying that if the subset-relation holds between the superset of a subset-relation and another set, the subset-relation also holds between the subset of the first relation and this other set.
We could of course represent this information explicitly in the subset-relation frame as follows (leaving out descriptions defining the normal constraints on the slots in the subset-relation)

```
(ABSTRACT-CONCEPT SUBSET-RELATION
  (WITH SUPERSET (FILLED-BY THE-SUPER-SET))
  (WITH SUBSET
        (WHEN THE-SUPER-SET
              ((DESCRIBED-AS SUBSET SUBSET-RELATION
                     (WITH SUPERSET (FILLED-BY A-THIRD-SET)))
                (DESCRIBED-AS SUBSET SUBSET-RELATION
                     (WITH SUPERSET (FILLED-BY A-THIRD-SET)))))))
```

But that is not really what we want. We want to introduce a frame for transitive relations and say then that the subset-relation is described-as a transitive relation. What we need in order to do this is a way of specifying mappings from the frame-name of one frame to the frame-name of another one and from the aspects of a frame to the aspects of another frame. We will do this by introducing a *viewed-as* operator denoted by /.

Returning to our example, we first define the transitive relation

```
(ABSTRACT-CONCEPT TRANSITIVE-RELATION
   (WITH BEING)
   (WITH ARG1
         (WHEN THE-ARG-2
               ((DESCRIBED-AS ARG1 TRANSITIVE-RELATION
                      (WITH ARG2 (FILLED-BY A-THIRD-ARG)))
                 (DESCRIBED-AS ARG1 TRANSITIVE-RELATION
                      (WITH ARG2 (FILLED-BY A-THIRD-ARG))))))
   (WITH ARG2 (FILLED-BY THE-ARG-2)))
```

and then we have as subset-relation frame

```
(ABSTRACT-CONCEPT SUBSET-RELATION
  (WITH BEING
        (DESCRIBED-AS BEING TRANSITIVE-RELATION/SUBSET-RELATION
              (WITH ARG1/SUBSET (FILLED-BY THE-SUBSET))
              (WITH ARG2/SUPERSET (FILLED-BY THE-SUPERSET))))
  (WITH SUBSET (FILLED-BY THE-SUBSET))
  (WITH SUPERSET (FILLED-BY THE-SUPERSET)))
```

The description attached to the being slot of the subset-relation frame can be paraphrased as "described-as being a transitive-relation viewed as a subset-relation with arg1 viewed as the subset and with arg2 viewed as the superset.

To illustrate the meaning of /, we now go through an example. Suppose we have three sets S1, S2 and S3 such that

```
(INDIVIDUAL-CONCEPT S1
    (WITH BEING
          (DESCRIBED-AS SUBSET SUBSET-RELATION
                (WITH SUPERSET (DESCRIBED-AS BEING (INDIVIDUAL S2))))))
(INDIVIDUAL-CONCEPT S2
    (WITH BEING
          (DESCRIBED-AS SUBSET SUBSET-RELATION
                (WITH SUPERSET (DESCRIBED-AS BEING (INDIVIDUAL S3))))))
```

then based on our knowledge of the subset relation we should be able to infer that (DESCRIBED-AS BEING S1) implies that

```
(DESCRIBED-AS SUBSET SUBSET-RELATION
      (WITH SUPERSET (DESCRIBED-AS BEING S3))).
```

From (DESCRIBED-AS BEING S1) we obtain immediately

```
(DESCRIBED-AS SUBSET SUBSET-RELATION
      (WITH SUPERSET (DESCRIBED-AS BEING S2))).
```

We now look at the constraint on the being slot of the subset-relation.

First of all we see that (FILLED-BY THE-SUPERSET) and thus also (FILLED-BY THE-ARG2) is co-referentially related to (DESCRIBED-AS BEING S2) and that (FILLED-BY THE-SUBSET) is co-referentially related to (DESCRIBED-AS BEING S1) thus to the ARG1 slot in the TRANSITIVE-RELATION frame. This frame yields then further information about the ARG1-slot as follows

```
          (WHEN THE-ARG-2
                ((DESCRIBED-AS ARG1 TRANSITIVE-RELATION
                      (WITH ARG2 (FILLED-BY A-THIRD-ARG)))
                   (DESCRIBED-AS ARG1 TRANSITIVE-RELATION
                         (WITH ARG2 (FILLED-BY A-THIRD-ARG)))))
```

In other words we know that (DESCRIBED-AS BEING S1) implies

```
 (DESCRIBED-AS ARG1 TRANSITIVE-RELATION
       (WITH ARG2 (FILLED-BY A-THIRD-ARG)))
```

when (DESCRIBED-AS BEING S2) implies

```
(DESCRIBED-AS ARG1 TRANSITIVE-RELATION
       (WITH ARG2 (FILLED-BY A-THIRD-ARG)))
```

But we did not include the effect of the viewed-as operator which would transform both descriptions into

```
(DESCRIBED-AS ARG1/SUBSET TRANSITIVE-RELATION/SUBSET-RELATION
       (WITH ARG2/SUPERSET (FILLED-BY A-THIRD-ARG)))
```

which becomes now

```
(DESCRIBED-AS SUBSET SUBSET-RELATION
       (WITH SUPERSET (FILLED-BY A-THIRD-ARG)))
```

Because it follows immediately from (DESCRIBED-AS BEING S2) that

```
(DESCRIBED-AS SUBSET SUBSET-RELATION
      (WITH SUPERSET (DESCRIBED-AS BEING S3)))
```

we now know that (FILLED-BY A-THIRD-ARG) is co-referential with (DESCRIBED-AS BEING S3). Thus we have shown that (DESCRIBED-AS BEING S1) implies

```
(DESCRIBED-AS SUBSET SUBSET-RELATION
     (WITH SUPERSET (DESCRIBED-AS BEING S3))).
```

This simple example shows already some of the steps the reasoning system working with this language will have to go through.

The TRANSITIVE-RELATION frame clearly has a general application. For example in the frame for equality we can specify that it is a transitive relation:

```
(ABSTRACT-CONCEPT SET-EQUALITY
   (WITH BEING
         (DESCRIBED-AS BEING TRANSITIVE-RELATION/SET-EQUALITY
               (WITH ARG1/SET1 (FILLED-BY THE-SET-1))
               (WITH ARG2/SET2 (FILLED-BY THE-SET-2))))
   (WITH SET-1
         (FILLED-BY THE-SET-1))
   (WITH SET-2
         (FILLED-BY THE-SET-2)))
```

Note that we can also use the viewed-as operator in other directions. For example as in

```
(ABSTRACT-CONCEPT TRANSITIVE-RELATION
  (WITH BEING
        (SPECIALIZABLE-AS BEING SUBSET-RELATION/TRANSITIVE-RELATION
                (WITH SUBSET/ARG1 (FILLED-BY THE-ARG1))
                (WITH SUPERSET/ARG2 (FILLED-BY THE-ARG2))))
  (WITH ARG1 (FILLED-BY THE-ARG1))
  (WITH ARG2 (FILLED-BY THE-ARG2)))
```

Finally note that the viewed-as operator is virtually identical to the /-operator in MERLIN (Moore and Newell,1973) and that it gives an elegant solution to second order inheritance.

7. SUMMARY

As a summary we will now give a formal syntax of the language in terms of a context-free grammar in BNF notation. We use square brackets for optional constituents because round brackets are part of the language itself.

```
<frame> := <concept-carrier> | <description>

<concept-carrier> := (<type> <frame-name> <slot-list>)

<slot-list> := (WITH <aspect-name> [<indirect-referential>]
                         [<description>])*

<description> :=  (<connective> <description>^n) |   ;n >= 2
                  (<negation-indicator> <description>) |
                  (<description-indicator> <aspect-name> <frame-name>
                              [<slot-list>] ) |
                  <co-referential> |
                  (<conditional-indicator> <referring-name>
                         (<description> <description>)+
                        [(ELSE <description>)]) |
                  <evaluation-indicator> <description>

<description-indicator> := DESCRIBED-AS | DEFINED-AS

<co-referential> := (FILLED-BY <referring-name>)

<type> := INDIVIDUAL-CONCEPT | ABSTRACT-CONCEPT

<frame-name> := ... arbitrary frame-name

<individual-name> := ... arbitrary individual-name

<aspect-name> := ... arbitrary aspect-name

<referring-name> := ... arbitrary referring-name

<conditional-indicator> := WHEN | UNLESS

<evaluation-indicator> :=  ^ | v

<connective> := AND | OR | XOR | MULTIPLE-AND

<negation-indicator> := NOT | NOT-NECESSARILY
```

8. DISCUSSION

In this paper I proposed a number of syntactic constructions to realize the framework for representing knowledge introduced in Minsky (1974). Of course what we discussed in this paper is only one part of the story. Supporting the language is a theory of procedural embedding which parallels the society of mind theory (Minsky and Paper, forthcoming) in that the basic computational entity is an expert (where an expert is a special kind of actor (Hewitt and Baker,1977)) which implements reasoning behavior by communicating with other experts. Some of the experts are responsible for frames, others embody principles of activation, augmentation and acquisition. More about all this in an upcoming paper.

When we say that a language is frame-based, we mean that at least all of the following principles of knowledge representation are reflected in the language:
1. Knowledge is organized in units called frames which contain all there is known about a certain subject.
2. A frame contains slots for 'conceptual aspects' of the concepts.
3. There is organizational control, i.e. the location of a piece of knowledge determines its accessibility during reasoning. E.g. in the present system selection restrictions cannot be extracted without knowing what frame is to be recognized or confirmed.
4. Frames are hierarchically related based on the instantiation relation, thus forming frame-systems (cf. the DESCRIBED-AS, DEFINED-AS constructs)
5. There are ways to express relations between frames in the sense of viewing one frame as another one. (cf. the VIEWED-AS operator)
6. There are ways to specify default properties of slot-fillers in a certain frame. (cf. the UNLESS-construct)

Other languages have been designed that introduce linguistic constructs realizing a frame-based representation methodology, such as FRL (Roberts and Goldstein,1977), KRL (Bobrow and Winograd, 1977), OWL (Szolovits, et.al.), a.o.. The semantics of these languages (which used to be somewhat unclear) have been clarified by relating them to predicate calculus (cf. Hayes,1977). Although there are certain similarities, there are (apart from the introduction of a number of new powerful language constructs such as the declarative quote and unquote) also substantial differences between those languages and the present system.
Let me mention two of them.

1. In contrast to other frame-based languages a frame does NOT represent an object of the domain. For example a frame with frame-name TABLE does not represent a table. Only slots have interpretations in terms of the domain (because they contain slot-fillers). This step leads to a more elegant language and as a consequence to more uniform and simpler activation mechanisms (as will be demonstrated in another paper). It also resolves the epistemological problems (known as the nominalism-realism debate) created by the need to postulate co-referential links between individuals and abstract concepts.

2. The inheritance mechanism discussed here differs substantially from

existing proposals. On the one hand first order inheritance, i.e. the straightforward extraction of descriptions from slots, can be performed in a uniform manner via every aspect of a frame instead of having to concentrate all inheritance on one single special aspect (often called AKO, or SELF). Moreover thanks to the viewed-as operator we now have a clear mechanism to represent second order inheritance, i.e. the mapping of one frame onto another by mapping the frame-name and the aspects. The merit of our solution is that it avoids dangerous practice such as making aspect-names global or changing aspect-names during the process of inheriting.

3. No attempt is made to introduce ways of attaching procedures at points in a frame. In early frame-based approaches all activation was performed by special purpose procedures which were attached to specific points in a frame (see e.g. Kahn (1975), Kuipers (1977), Goldstein and Roberts (1977b), a.o.). There are however two problems with this approach. First of all it appears that many procedures such as those dealing with the connectives, fetching descriptions and other mechanisms of reasoning are needed in all frames. We would like to capture these generalizations. Second the procedural attachment restricts the use of a frame to the application for which procedures have been defined. It seems that a more powerful approach consists in making smart, more or less general purpose activators which are task independent. Third it is clear (at least in principle) how declarative knowledge is acquired (see e.g. Winston (1970) or Piaget (1975)) but it is not at all clear how all the procedural knowledge gets into specific frames.
An alternative (which was also implied by Minsky's original proposals) consist in having general purpose reasoning mechanisms which however play a smaller role than in earlier problem-solving systems. Frames would be introduced which contain almost complete solutions to possible problems. The function of the activation mechanisms consists in (i) finding the appropriate plan based on a description of the problem and (ii) adjusting the plan to the actual situation, e.g. by filling in the details.
This wholistic approach to problem solving is exemplified in Sussman's theory of "Planning by debugging almost right plans" and in McDermott's recent work (e.g. McDermott (1978a,b)). The method is reminiscent of the application of the frame hypothesis to the domain of text understanding (cf. the review in McDonald,1978).

Finally a word on the relation to another family of declarative languages descending from 1st-order predicate calculus. It is quite clear that the constructs making up the language introduced in this paper all have an equivalent in some logical calculus. The difference between logical calculi and frame-based languages does therefore *not* consist in the nature of the language, nor the possibility of being subjected to a Tarskian style formal semantics (whatever the value of that may be for understanding intelligence or communication, cf. Mondadori,1978). What is different however is the way knowledge is organized and this influences both what one can express and how this has an impact on the inference mechanisms. (All this in response to Hayes,1977).

9. REFERENCES

Bobrow, D. and T. Winograd (1977)
An overview of KRL: A Knowledge Representation Language.
in: Cognitive Science I, 1.

Carnap, R. (1947)
Meaning and Necessity. Chicago: Chicago University Press.

Church, A. (1951)
The Calculi of Lambda-Conversion. Princeton: Princeton University Press.

Doyle, J. (1978)
Truth Maintenance Systems for Problem Solving. MIT-AI TR-419.

Fahlman, S. (1977)
A system for representing and using Real World Knowledge. MIT-AI TR-450.

Goldstein, I. and Roberts, F. (1977)
NUDGE. A Knowledge-based Scheduling Program. MIT-AI memo. 405

Hewitt, C. and H. Baker (1977)
Actors and Continuous Functionals. MIT-AI memo 436.

Hughes,G. and M. Creswell (1973)
Introduction to Modal Logic. London: Methuen and Co. Ltd.

Hayes, P. (1977)
In defense of Logic. IJCAI-1977. p.559-565

Hayes, P. (1977)
The Logic of Frames. mimeo.

Hewitt, C. (1976)
How to use what you know. MIT-AI WP.93

Kahn, K. (1975)
Mechanization of Temporal knowledge. MIT-MAC TR-155

Kuipers, B. (1977)
The representation of knowledge about large scale space.
MIT-AI TR-402.

Martin, W.A.M. (1978)
Descriptions and the Specialization of Concepts. MIT-LCS. TM-101

McCarthy, J. (1977)
Epistemological Problems in Artificial Intelligence. IJCAI-1977. p.1038-1044, Cambridge, Ma.

McDermott, D (1978a)
Flexibility and Efficiency in a Computer Program for Designing Circuits. MIT-AI TR-402

McDermott, D. (1978b)
Planning and Action. Cognitive Science, 2,2.

McDermott, D. and G. Sussman (1972)
Why conniving is better than planning. MIT-AI memo 255

McDonald, D. (1978)
Story Understanding: The beginnings of a consensus. MIT-AI WP 168

Minsky, M. (1974)
A framework for Representing Knowledge. in: P. Winston (ed.) The Psychology of Computer Vision. New York: Mc Graw Hill, 1975.

Minsky, M. and S. Papert (forthcoming)
The Society of Mind.

Moore, R. and A. Newell (1973)
How does MERLIN understand? in Gregg, (ed.) Explorations in Cognition. Potamac, Md: Lawrence and Erlbaum Ass.

Mondadori, F. (1978)
Interpreting Model Semantics. p.13-40.
in: Guenther, F. and C. Rohrer. Studies in Formal Semantics. Amsterdam: North-Holland Publ. Co.

Montague, R. (1970)
The Proper Treatment of Quantification in ordinary English.
in: R. Thomason (ed.) Formal Philosophy: Selected Papers of Richard Montague. New Haven: Yale University Press, 1974.

Piaget, J. (1975)
L'equilibration des structures cognitives. Paris: P.U.F.

Popper, K. (1974)
The logic of scientific discovery. London: Penguin books.

Roberts, F. and I. Goldstein (1977)
The FRL manual. MIT-AI memo.409

Russell, B. (1940)
An Inquiry into Meaning and Truth. London: Allen and Unwin.

Smith, B. (1978)
Levels, Layers and Plans, The Framework of a System of Knowledge Representation Semantics. MIT, unpublished Ms. Thesis.

Strawson, P.F. (1970)
Individuals. London: Methuen and Co. Ltd.

Szolovits, P., B. Hawkinson and W. Martin (1977)
An overview of OWL, A language for knowledge representation
MIT/LCS/TM-86.

Winograd, T. (1978)
Semantic primitives and prototypes. in TINLAP-2. Illinois.

Winston, P. (1970)
Learning Structural Descriptions from Examples. in Winston, P. (ed)
The Psychology of Computer Vision. New York: McGraw Hill,1975.

Acknowledgement

Comments of Roger Duffey improved the content and the style of this paper.