AUTOMATIC PROGRAMMING GROUP
Project MAC, M.I.T., Cambridge, MA

Internal Memo 8

# More MAPL: Specifications and Basic Structures

by

William A. Martin,

Rand B. Krumland,

and

Alexander Sunguroff

In Memo 6 we introduced a notation or language for building world models. Here we modify and expand the specifications for the language. We show how to handle certain basic problems such as characteristics, time, and certain physical spatial relations. In addition we develop other structures expressly for the world of business (WOB) model.

## 1. MAPL Specifications and Changes

Since Memo 6 various changes have occurred in MAPL. We attempt to describe them here and to give further specifications of the language. MAPL is intended as a language in which models of various worlds or contexts can usefully be cast. As such we can think of three levels or types of specifications:

1) MAPL system level: specification of language data types and constructs which are allowable in the language;

2) world level: specification of how MAPL constructs are utilized to store general knowledge and information about a world; and

3) user level: specification of how a user's particular situation can be cast as an instance of the more general world model.

This section of the paper will be devoted to further work at the MAPL system and world system levels. Sections 2 and 3 explore the subjects of assigning characteristics and treating time which are kinds of world information which will be of use in many world models. Section 4 will deal with some of the constructs in the World of Business. Throughout Sections 2, 3 and 4 we will cover many of the facets of how MAPL constructs are utilized to store world information.

### 1.1 Identifiers

The symbol "#" as a data type identifier has been eliminated for the time being. MAPL is primarily based on set theory terms and notions, and the use of # to designate a concept or predicate did not seem to enhance its power to any great extent. The other symbols previously used remain. The symbol "%" designates a set, and "$" designates a variable which can represent any element of that set. We will often refer to a $ variable as a typical element.

## 1.2 Constructs

The basic construct or statement of the language is a relationship which is given as a LISP-like list -- e.g.,

(%foo bar-1 bar-2 ... bar-n).

The first element of a relationship is a set which is a relation on the remaining n elements of the list, that is, it is a set whose elements are ordered n-tuples. An element may be either a set %bar-i or a typical set element $bar-j. To specify the form of a relationship we use a special relation called %a-r-o (for "a relation on"; note that %a-r-o replaces #a-p-o of Memo 6). For example,

(%a-r-o %generator-of $activity $data-item)

establishes or "enables" the relation %generator-of. Once a relation is established or enabled by %a-r-o it can be further instantiated by a relationship in which it is the relation and n other objects are given in place of the n-tuple in the enabling relation.

The most basic relation we have is %a-k-o (for "a kind of"):

(%a-r-o %a-k-o %object-1 %object-2).

this says that the set %object-1 is a subset of %object-2. For example, to declare that Pontiacs are a subset of cars we could use an instantiation such as

(%a-k-o %pontiac %car).

The %a-k-o relation serves as a restraint on how other relations can be instantiated. If the ith element of an n-tuple in an enabling relation is a set, %a, then only sets which are a kind of %a can be used as the ith element in any instantiation. Similarly, if the ith element is a typical element, $b, then only typical elements of the set %b or of any subset of %b can be used in an instantiation. Thus, if we have

```
(%a-r-o %speed-of $speed $car),
  (%a-k-o %fast %speed), and
    (%a-k-o %pontiac %car),
```

we could say

```
(%speed-of $fast $pontiac),
```

but not

```
(%speed-of $fast $chair).
```

Other restraints on instantiation and the meaning of any particular relation can be more completely specified or limited by the use of the mode, mapping-type, scope and/or constraint properties given in Memo 6.

## 1.3 Specifying Names and Values

In Memo 6 we named relations by the use of the #a-n-o predicate; for example, we stated

```
(#a-n-o %r1 (#a-c-o #color $fruit)).
```

However, we also used #a-n-o in a statement such as

```
(#a-n-o $a&t-store-sales $dollar-value).
```

This dual usage of #a-n-o made it ambiguous. In MAPL we actually need three types of "naming" mechanisms: to give names to relations, to give the range and number-type for typical elements of sets which are measures, and to give specific number values for specific measures. Consequently, we use three separate relations which collectively replace

#a-n-o: %h-r-o-f, %specified-by, and %has-value.

A relationship is an object and as such it can be given a name. We accomplish this by stating that there exists a set, each of whose elements is a relationship of a certain form. This is done with the %h-r-o-f (for "has relationship of form") relation; e.g.,

(%h-r-o-f %r1 %color-of $color $fruit).

By our normal conventions $r1 stands for any 3-tuple of the form

(%color-of $color $fruit).

In section four of this memo we will show how properties that convey truth and time can be attached to relationships using this ability to name them.

For measures or concepts which are kinds of measures we need to indicate the class and range of numbers which a typical element may take on or be assigned as well as the %unit-of-measure characteristic of the element. For example, a typical element of the set %a&t-store-sales is a positive rational number with two decimal places to the right of the decimal point and the unit of measure is %dollar. We could assign these properties directly, but it is more in line with our normal conventions to state that %a&t-store-sales is a kind of %sales-measure which in turn is a kind of %dollar-measure. A %dollar-measure is a kind of ratio-measure with the specifications

(%specified-by $dollar-measure $rational-number), and

(%unit-of-measure-of $dollar $dollar-measure).

Thus, %specified-by is seen to assign classes of allowable numbers to typical elements of sets which are kinds of measures. As we mentioned in

Memo 6, one kind of number may have many elements which it specifies, but an element can only be specified by one kind of number.

In order to give a specific element a specific number value we use %has-value. Thus if $a&t-store-31-sales are $3,172.10 we state

value-of
(%has-value $a&t-store-31-sales 3172.10).

The unit of measure, dollars, is implied if $a&t-store-31-sales has earlier been declared to be a kind of %a&t-store-sales. Note that %specified-by and %has-value are complementary relations; a %has-value statement will be questioned or rejected unless its first argument is a kind of a measure which has previously been assigned a number type in a %specified-by relation.

## 1.4 Sequences

A sequence is defined to be a set of ordered subsets of a set. A frequent case is where the subsets contain only one element apiece. The relative order of two of the subsets of a set is indicated by the relation %has-order on the set and its indicated subsets:

(%a-r-o %has-order %object %object-i %object-j)

where it is required that %object-i be different from %object-j and for both of them to be a kind of the first object. Thus if %miniscule and %small are each a kind of %size we can indicate their relative order by

(%has-order %size %miniscule %small).

Note that %size may have other subsets besides the two indicated and other %has-order relationships will have to be given to establish the order of the subsets with respect to each other and/or to those already ordered. Many of the characteristics and time sets which we discuss later will have order relations on their subsets. Also, subsets of some sets may have a circular or ring-like ordering in that while there is a distinct ordering, the order essentially repeats itself and there are no definite breakpoints; an example is the set %day-of-week. To describe such a case, the ordering relations are given and %cycle is noted as a characteristic of the ordered set.

## 1.5 A Note of Caution

There are a number of important points that must be remembered when using MAPL, especially at the world modeling level. MAPL is a formalism for expressing meaning, and in particular, the type of meaning that humans possess about the real world. In other words, meaning concerns an intelligent entity's "internal model" of some "external world" (where the use of "world", "model" and even "world model" can lead to confusion by applying these ideas recursively; see Minsky (1965)). MAPL is a method for expressing these internal models; the meaning of any MAPL construct is therefore relative to the person possessing the internal model, or more precisely, to the internal model which is being expressed.

At present time there are no programs that consider that they know the meaning of MAPL constructs. Thus at the present time establishing what something means in MAPL whether at the System level or the World Modeling level entails establishing a correspondence between the internal models of some group of humans and the constructs of MAPL.

The following are some of the ways in which people, or even one person at different times, who are attempting to arrive at a consensus, run into difficulties.

Most problems appear to arise from the fact that a person's internal model appears to him to be constructed in terms of natural language. In attempting to establish the correspondence between our models and a MAPL description we are faced with problems introduced by this dependency on natural language.

Words in English have either multiple meanings or variations in meaning that are dependent on context. In assigning names to sets that are supposed to be unambiguous, one often runs out of single word names. Worse, once a convention is established for naming sets a situation often develops in which two sets seem to compete for some good name. For example, it is easy to confuse a set of characteristics with a set containing objects that possess those characteristics.

This leads us to a second problem. Once a set is established its meaning must be kept clear; that is, the elements it was meant to contain must be defined and separate from other possible system elements. In the previous example a reference to an ambigous set might yield an object or an object's characteristic. Let us take a physical example from Winston. If the set %arch contains arches, then a subset of one element would contain an arch. However one could create a set called %arch-prime which contained atoms and which had arches as subsets. In the set %arch-prime the structural constituents of arches--i.e. blocks--could be referred to by using the subset relation while in the set %arch a specialized relation such as %part-of might serve an equivalent purpose. Trouble can easily result from such dual (or over) specification.

Another difficulty is in the use of the concepts of a structural description and primitivity. Their common interpretation is that information is stored by constructing a structure from primitives. If the structural descriptions exist in some body of data, then the primitives have their definition in some program (or some other data base being read interpretively). For example, the LISP interpreter knows about subrs and Winston's routines know about such things as left and right. However, in the case of an interpreter it is clear what the direction of simplicity is, machine language, and there is only one program, the interpreter, which defines the meaning of the primitives, the subrs. Thus, this notion of primitivity is based on the idea that there is one conceptual module which contains the meaning of an entity and which does not share this information with everyone else. MAPL, though, is meant to be a language with which different modules of a system can communicate. There is no one module that contains information that can be said to define a MAPL construct. Instead, any of the modules that speak MAPL may have some additional information that they do not share in the common base. It is debatable whether it is of use to stretch the notion of primitivity to be able to handle this more complex case. In this context, primitivity becomes relative to the modules which contain unshared information.

As an example let us use a zero order model of the top part of

Protosystem 1. The following modules will all speak MAPL:

1) a "learner",

2) a "deductive system",

3) an English language system,

4) various "activity expert modules".

In addition, a person who circumvents the English languge system becomes in effect, yet another module. Each of these has information binding MAPL constructs to other forms. The activity expert modules have definitions of MAPL contructs in terms of lower level simulation languages. The English language system has defintions expressed in English. A human has all his thoughts as to what some particular construct means. Only the learner and the deductive system can at all compare to the power of the authority of the LISP interpreter. They are the ones who are convinced of the meanings of things stated at the MAPL System level.

## 2. Specification of Characteristics

In Memo 6 we described how objects were assigned as characteristics to other objects by the use of the #a-p-o predicate or the #a-c-o predicate. The #a-p-o predicate has been eliminated and newer versions of input macros are being developed. The set %characteristics are descriptors which are usually regarded as characteristics, attributes and/or properties of physical or meta-physical objects in ordinary language. We handle them by making them a subset of %object's and assigning them to objects via relations. In order to specify characteristics and relations between objects according to characteristics more completely, we employ a classification scheme from measurement theory (see Siegel, pp. 21-30; this illustrates that to some extent we are taking our concepts from practitioners rather than linguists and philosophers). That is, any concept which is a characteristic of another object will also be considered to be a kind of a %characteristic. Each characteristic will be located in a hierarchy of measurement scales which indicate how values of the characteristic can be established. The four measurement schemes or scales are:

1) nominal or classificatory scales: these specify classes or partitions of the objects, no order or relative rank implied;

2) ordinal or ranking scales: these specify some kind of order or rank relation between different equivalent classes, no measure of amount or magnitude of differences implied;

3) interval scales: these specify rankings in which differences or distances between any two numbers on ranking scales are of known

size, no true zero point exists on the scales; and

4) ratio scales: these are the same as interval scales, but in addition each scale has a true zero point and hence ratio of any two values on the scale is independent of the unit of measurement.

The scheme for specifying characteristics is shown as a net in Figure 1. (In that diagram unmarked lines represent the %a-k-o relation.) Also note that ordinal, interval and ratio scales are a kind of %scalar characteristic, and that both interval and ratio scales are a kind of %measure's. Note that the measurement schemes are arranged in increasing order of specificity; that is, an ordinal scale is also a nominal scale, a ratio scale is an interval scale, etc. We enable the specification of nominal and ordinal characteristics by forming a relation of the characteristic name and the suffix "-of"; e.g.,

(%a-r-o %color-of $color $object).

A characteristic can then be specified by

(%color-of $red $block-1).

In general such declarations will be accepted if a) the nominal characteristic has earlier been declared to be a characteristic of the elements of a set of which the object indicated is a member, and b) the value has been declared to be %a-k-o the characteristic. Thus, for the above example to be accepted the system must have previously been told that %red is a kind of %color and that %color is a characteristic of the elements of a set, say %block, of which %block-1 is a subset.

One way to think of nominal characteristics is that they arise through the operation of mechanisms which people use to order and

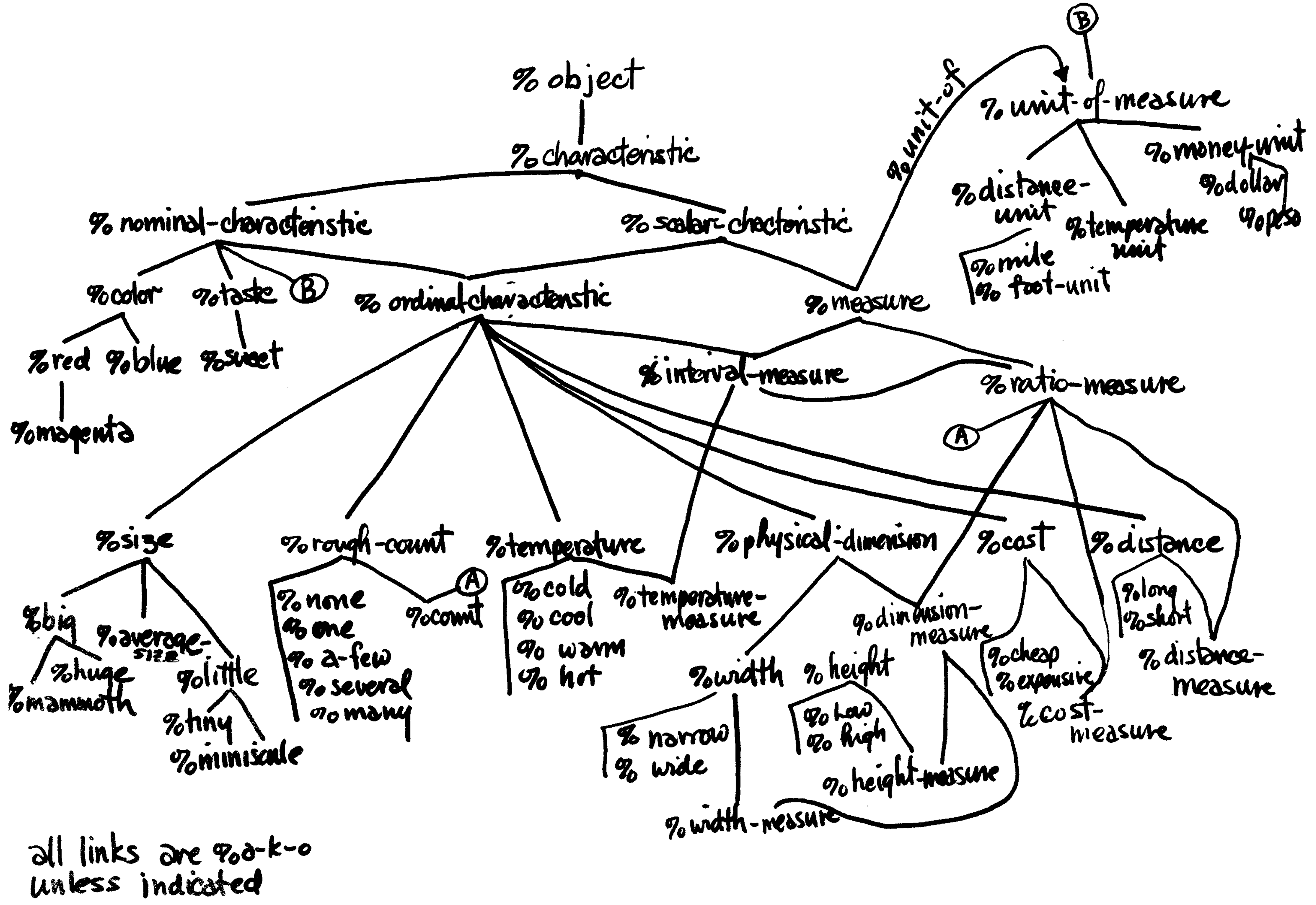

Figure 1. Partial concept net for characteristics.

classify their environment; such mechanisms involve fairly course transformations to sensory input data. For example, for a physical characteristic such as color, a person distinguishes objects based on differences which his eyes are able to sense in the radiation reflected by the objects. Rather than assign a separate category to every discernible hue and shade, certain economies of information handling and storage are affected, and objects are clustered around a few points such as white, red, yellow, blue and black which more or less cover the space of all colors and around which other objects can easily be grouped. Thus, objects are clustered to maximize similarity with other objects according to the color property.

Given these clusters a "typical object", or in the case of color a "typical shade" can be abstracted from each group. New objects can be compared against this typical one in order to place them in the correct cluster. To make such comparisons measures of similarity are necessary; thus, an object might be closer to a typical red object than a typical blue object in color and would therefore be assigned the color red, and within the red group it might be more similar to the typical red object than some other object and would therefore be considered more red than the other. In MAPL we make such comparison statements by using the ordinal characteristic %similarity which relates two objects according to some other characteristic: the enabling relation is

(%a-r-o %similarity-of $similarity $characteristic $object $object),

and an example might be

(%similarity-of $high $color $sky $ocean).

In addition, a statemment such as "object-1 is redder than object-2" is easily handled if we allow ordinal scales on such things as "redness" within the cluster of red objects; i.e., we can handle it with the mechanisms used for ordinal characteristics described below.

Scalar characteristics allow objects to be compared according to that characteristic. Ordinal characteristics have monotonic scales associated with them and objects are ranked by the specification of their position on those scales. The scales have regions which function as nominal classes except that the regions are ordered. The ordering is usually implied by the English words used to name regions. Thus, on the temperature scale %warm is "less than" %hot. For each ordinal characteristic all relevant regions are a kind of that characteristic -- %light-weight is a kind of %weight -- and the order of the regions on the scales is given by %has-order relations --

(%has-order %weight %light-weight %heavy).

Many ordinal scales have a simple two region dichotomy as does %weight (as long as we neglect such potential categories as %burdensome and %crushing). For such scales two modifiers -- very and extremely -- could be used to add more regions; e.g., %very-heavy. The ordering of such additional regions with respect to the original regions is straightforward. However, for certain other characteristics such as %size or %temperature, there are seemingly many possible regions which have more than one name. For example, several of the possible regions on the %size scale are given in Figure 2. Note that if the overlapping regions are arranged as in Figure 2, they give us an option of the degree of precision to be employed.

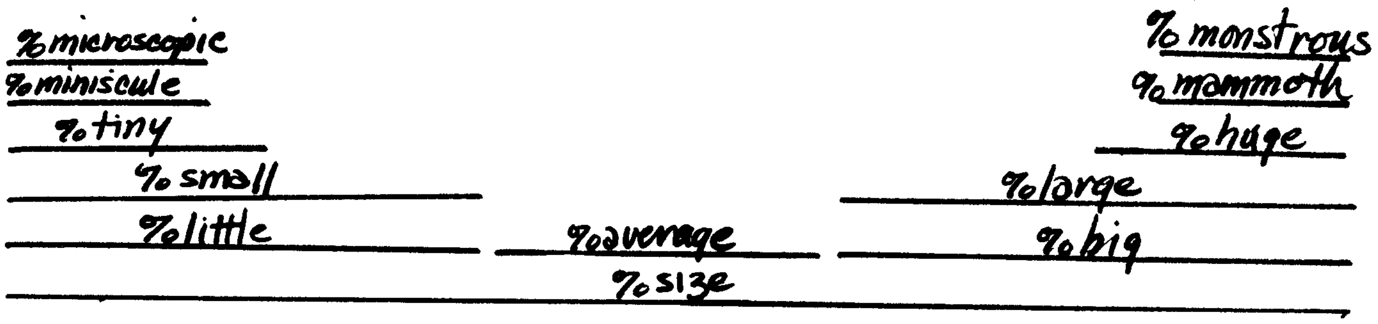

Figure 2. %size Scale.

For names which we consider to be synonyms we eliminate all but one of them set by making an "alias check" in the input routines. The resulting unique, but perhaps overlapping, regions are then ordered, and in this case ordered with respect to their relation to the middle of the scale:

(%has-order %size %miniscule %tiny),

(%has-order %size %tiny %small), etc.

Such an ordering will allow the recording and comparison of characteristics such as %size, but some care must be exercised in dealing with overlapping regions.

There are several difficulties with this scheme as it stands. The complexity introduced by allowing several regions, some of which overlap each other, on a single scale has not been explored. Also, in ordinary discourse when a reference is made to an ordinal scale in describing a characteristic, an implicit relation is usually made or an implicit reference frame is usually in mind. When a person says that a truck is big, he means that it is big when compared to other trucks, or to some average truck, or perhaps to all vehicles which move on the highways.

Similarly, a "warm" oven is certainly of a higher temperature than the "hottest" of days. Whether or not this will be a problem will depend on the demands to be placed on the knowledge structure. Our objective here is to provide a language for guiding heuristics as well as for making precise, unambiguous statements depending on what the situation calls for. We should also note at this time, that such relativity is a major reason for wanting to be able to speak about characteristics, independent of the objects to which they are applied. Any element of the set, %miniscule-object, which contains all objects that have the characteristic $miniscule, is not smaller than any element of %tiny-object. The ordering applies only to the characteristics $miniscule and $tiny. Whether the ordering or any other characteristic of a characteristic carries through to the objects that possess that characteristic, is arbitrary. If some characteristic does carry through then this has to be so stated.

In order to express a relation between two objects with respect to some ordinal characteristic two avenues are open. We can assign them both to appropriate regions on the scale of the characteristic and let the relation between the objects be implied by the relation between the assigned regions. Or, we can place the relation directly into the data base by means of a relation called %magnitude-relation. The general form of such a statement is

```
    (%a-r-o %magnitude-relation $characteristic $object $object),
```

and a specific example might be

```
                (%greater-than $cost $apple-1 $apple-2).
```

Allowable kinds of magnitude relations are:

1) %much-greater-than,
2) %greater-than,
3) %about-equal-to,
4) %equal-to,
5) %less-than, and
6) %much-less-than.

The second kind of scalar characteristics are %measure's. Some properties of %measure's were given in Memo 6, i.e., they each evaluate to a real number or to a number from a subset of the reals, and they each have an associated %unit-of-measure as a characteristic. Several relations are used in describing a measure and its current value for a given object: if

```
(%a-r-o %cost-measure-of %cost-measure $apple), and
    (%specified-by $cost-measure $real-number),
```

then we can say

```
              (%a-k-o %apple-1 %apple),
         (%a-k-o %apple-1-cost %cost-measure),
    (%cost-measure-of $apple-1-cost $apple-1),
         (%has-value $apple-1-cost .20), and
   (%unit-of-measure-of $dollar $apple-1-cost).
```

Shorthand will be developed for inputting such statements. Also note that various input conventions will allow more efficient specification of characteristics for individual objects than is indicated by this example. For example, if the costs of all items under consideration in a modelled system are specified to have a $dollar unit of measure, then that fact need not be specified for individual items as they are made known to the concept net.

Nearly all %measure characteristics also have %ordinal counterparts; for instance, %weight has an ordinal scale that runs from %light-weight to %heavy, and a %weight-measure may have a %pound or a %gram scale associated with it. Such a structure is necessary to allow the kinds of specifications that people normally give and to capture the knowledge or information that is conveyed in such specifications. However, certain problems may arise. For instance, if we attempt to compare a characteristic specified on an ordinal scale for one object with the same characteristic specified on a %measure scale for the same kind of object, we may have trouble relating the characteristics; e.g., if

```
                (%cost-of $cheap $apple-1),
      (%cost-measure-of $apple-2-cost $apple-2), and
             (%has-value $apple-2-cost .20),
```

how is the cost of $apple-1 related to that of $apple-2? This is another part of the frame of reference problem we noted earlier. If we know the %physical-dimension's of a %truck as well as its %carrying-capacity, %weight and %horsepower, what do we know about its %size? (Almost any person who knew the values of all those measure characteristics would be able to say something about the truck's size.) These type problems may be overcome in some cases by the use of heuristics to relate ordinal and measure scales. However, the important thing to realize is that our knowledge and data structure is not intended to provide definitive answers to such problems, but is rather intended as a flexible means for describing appropriate states of the world and for allowing interaction with that description. For an attack on related problems the reader is referred to the paper by McDermott.

## 3. Time and Truth

In order to construct almost any kind of useful model for a world, we need various notions regarding time. For instance, if a relationship is only true for some part of the overall time period considered by the model, we need mechanisms to represent that in the modelling language. The basic mechanism we use is the %t-a-t-of (%time-and-truth-of) relation which establishes a correspondence between a structural relation, a time period to relate to that relation, and a relative truth value; e.g.,

(%a-r-o %t-a-t-of $truth-value $relationship-name $time-interval).

In this section we explore the concepts and structures related to "time-intervals" and "truth".

### 3.1 Time Intervals and Time Points

The basic time data items are elements called $time-intervals's. In English an interval is commonly regarded as the "space" between two objects or points. Consequently, a $time-interval will be distinguished by its beginning or $start, its $end and its length or $time or $duration. The set %time is a kind of %ordinal-characteristic with subsets %short-time, %average-time, %long-time, etc. The set %duration is a kind of %time, but it also is a kind of %ratio-measure with possible $units-of-measure such as $hour, $day, $year, etc. The sets %start and %end are kinds of %time-point's which are more difficult to handle, but which can be considered as %interval-measure's. A time point has several defining characteristics as shown in Figure 3. The characteristics

$time-point, $time and $duratation are restricted to apply only to $time-interval's. Note that a $time-interval can be defined which is not strictly related to any relationship by appearing in a %t-a-t-of relation.

Therefore if $r1 is a name of a relationship which defines a selling activity, i.e.,

(%h-r-o-f %r1 %a-k-o $a&t-store-selling $selling)

then the time when an $r1 occurs (the time for which the relationship is true) could be given time interval $ti1 in relations such as

(%start-of $start-of-ti1 $ti1).

To specify that A&T stores open at eight o'clock in the morning we say

(%hour-of $start-of-ti1-hour $start-of-ti1)

and

(%has-value $start-of-ti1-hour 8:00).

It is evident that relations proliferate in making extensive or precise time specifications. However, the structure allows a natural specification of time in the same way that people make and use time references. The result is that the time specifications and the entire time structure of a model in MAPL may be ambiguous and incomplete to some extent. Even so the model may contain enough information to allow substantial deduction, question-answering, simulation, specification of information systems, etc. In some instances, parts of the system will require more complete time specifications before or as the time data is used. The strategy, however, is not to require such specification until it is necessary.

Time points may be specified by giving pertinent characteristics

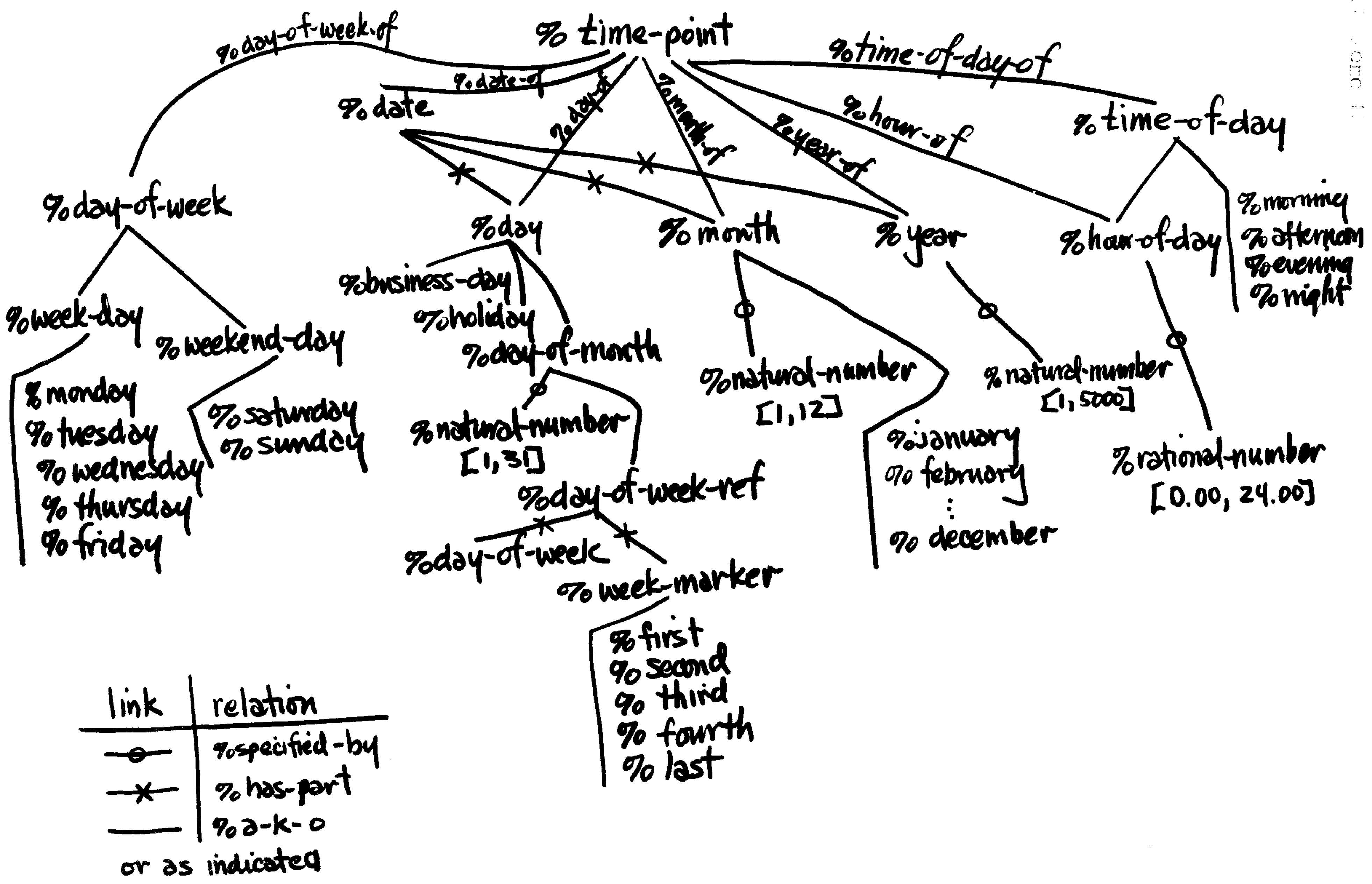

Figure [illegible]. Concept net for %time-point's.

as above, or by instantiating certain kinds of relations between time points and/or time intervals, i.e., by creating temporal relations. To enable such references the characteristics %time-referent and %time-interval-referent are used:

```
    (%a-r-o %time-referent-of $time-referent $time-spec $time-spec),
and
    (%a-r-o %time-interval-referent-of $time-interval-referent
                                            $time-spec $time-interval).
```

A %time-interval-referent is a kind of %time-referent, and their elements are shown in Figure 4. Note that we qualify the %at to distinguish a temporal "at" relation from a physical or spatial one.

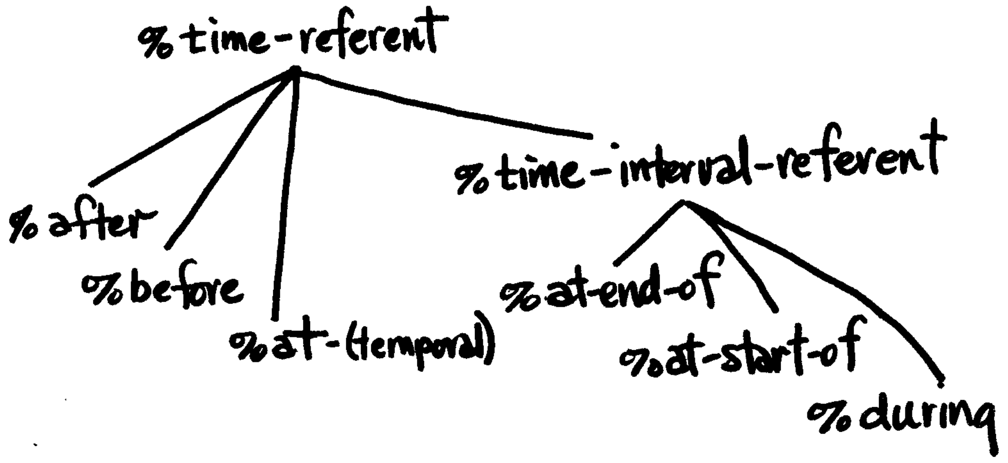

Figure 4. %time-referent's.

The set %time-spec is a union of the sets %time-point and %time-interval; that is, either a $time-point or a $time-interval could be used in an instantiation. The use of these relations is straight forward and directly follows our previous conventions. For example, if

```
        (%a-k-o %ti-r1 %time-interval),
        (%a-k-o %ti-r2 %time-interval),
      (%a-k-o %start-ti-r1 %time-point),
       (%a-k-o %end-ti-r1 %time-point),
```

then we could say

```
     (%time-referent-of $before $start-ti-r1 $end-ti-r2), or
    (%time-interval-referent-of $during $start-ti-r1 $ti-r2)
```

which have the obvious meanings.

Note that there is a potential ambiguity in statements such as

```
          (%time-referent-of $after $ti-r1 $ti-r2)
```

since it is not clear how much of $ti-r1 is "after" how much of $ti-r2. Therefore the convention is that in statements with $after or $before instantiated with $time-intervals, the $time-interval's will not overlap at all; e.g., in the example above all of $ti-r1 will be assumed to occur (sometime) after all of $ti-r2. A time-referent-of relation instantiated with $time-interval's and the time referent $at will have the meaning "at the same time as". The $time-interval-referent's can be used for less precise statements.

## 3.2 Truth

Returning to the original %t-a-t-of relation we also specified a set of %truth-value's. A $truth-value is relatively simple; it can have a value anywhere in the range 0 to 1. Therefore we have the statement

(%specified-by $truth-value $real-number-1)

where %real-number-1's are kinds of $real-numbers restricted to the (0, 1) interval. A relationship which is true (false) for the specified $time-interval has a value 1 (0). Numbers between 0 and 1 indicate the relative probability that the relationship is true for the $time-interval.

## 3.3 Multiple Occurrences

For convenience we call a relationship which participates in a %t-a-t-of relationship an event. The mechanisms described to this point allows the statement of many aspects of time characteristics and their assignment to relationships, that is a fairly powerful description of an event is allowed. However, a description is limited to simple events which occur only once, that is, events which have some truth value only for a single interval. Additional mechanisms are needed for events with multiple occurrences or which have some truth value for more than one interval. The strategy to accomplish this is to specify a subset of time-intervals which contain elements describing the multiple occurrences. Various properties on such a set augment the description in order to handle such things as repitition.

The primary additional characteristic needed to specify multiple occurrences is a %recurrence-pattern. A $recurrence-pattern specifies the time between successive starts of the time intervals that are elements of the subset of time intervals to which it is assigned. The time interval elements are presumably ordered within the subset. The following example should help to clarify the usage. Suppose that we wish to describe the time aspects of a relation $r1 that occurs on Monday's, Wednesday's and Friday's. We could state the following:

```
            (%a-k-o %ti-r1 %time-interval),
          (%t-a-t-of $r1-truth $ti-r1 $r1),
              (%has-value $r1-truth 1),
              (%a-k-o %ti1-r1 %ti-r1),
             (%day-of $monday $ti1-r1),
        (%has-order $ti-r1 $ti1-r1 $ti2-r1),
        (%has-order $ti-r1 $ti2-r1 $ti3-r1),
        (%a-k-o %rp-r1 %recurrence-pattern),
      (%recurrence-pattern-of $rp-r1 $ti-r1),
             (%has-value $rp-r1 1), and
              (%units-of $week $rp-r1)
```

where $ti2-r1 and $ti3-r1 are described similarly to $ti1-r1. Thus $ti-r1 has an order and a recurrence pattern. The conventions associated with such statements imply that the elements of $ti-r1 are repeated (recur) at intervals specified by $rp-r1 and they occur in the order specified by the various %has-order relations on $ti-r1.

MAPL can allow further statements or qualifications such as that $r1 occurs as above but only during the months of November and December. In making such statements note that we are describing another time-interval for which our previously defined time-interval set, %ti-r1, is true. Therefore we describe such an interval and assign it to %ti-r1 in a second $t-a-t-of relation; e.g.,

```
           (%a-k-o %tii-r1 %time-interval),
      (%t-a-t-of $ti-r1-truth $tii-r1 $ti-r1),
           (%start-of $tii-start $tii-r1),
      (%month-of $tii-start-month $tii-start),
          (%has-value $tii-start-month 11),
```

and similarly

```
            (%has-value $tii-end-month 12).
```

The notation for representing such time information obviously becomes somewhat extensive; however, the time information itself is fairly complex.

## 4. Physical Models

As an example of how MAPL can be used to build up a knowledge structure for a world we examine a class of spatial or positional location relations which could be used in a physical world model. Spatial relations or relative physical locations can be thought of as characteristics of two physical objects,

(%a-r-o %location-of $location $phys-obj $phys-obj).

We could define the following kinds of spatial relations:

1) %at-(spatial),
2) %near,
3) %inside,
4) %outside,
5) %far-from,
6) %on,
7) %under,
8) %above,
9) %below,
10) %in-front-of,
11) %in-back-of,
12) %right-of, and
13) %left-of.

Thus in a world of toy blocks we could say

(%a-k-o %block1 block),
(%a-k-o %block2 %block), and
(%location-of $right-of $block1 $block2)

which would mean that block1's location is to the right of block2. As with time we qualify the spatial concept %at to differentiate it from temporal %at.

## 5. World of Business Constructs

There are many different kinds of situations that it would be useful to allow a user to model in such a broad world as the world of business. For each different situation which the business world modeller provides or allows a certain amount of knowledge or expertise concerning that general situation must be encoded in MAPL constructs. However, certain knowledge is basic to nearly all business world situations and therefore to nearly all models of the business world. Both the $characteristic and $time-and-truth structures are part of such knowledge. For some situations a version or subset of the physical location relations may also be useful as general WOB modelling constructs. In this section we will briefly cover structures concerning two other classes of objects which will be of particular importance in most business world models: $activity's and $data-item's. An activity is a rigorous specification of an intuitive notion such as "selling". Thus, it will not have all of the properties of a meaning of the English word "selling". The use of words such as selling in their more general sense will be discussed in a forthcoming Memo on the translation of English to MAPL.

### 5.1 Activities

In many WOB situations certain kinds of events which represent changes of state in the world are of particular importance. Examples of such changes are production processes, inventory control actions, various kinds of file manipulations and document productions, etc. We refer to such events as %activity's:

```
                    (%a-k-o %selling %activity)
```

All $activity's are recurrent events. Most properties of $activity's are not specific to a given occurrence, but a given occurrence can be given specific properties. $activity's are closely intertwined with %responsibility-center's which we consider to be the basic structural-organizational unit in the WOB. We will loosely follow a scheme used by Schank, et al (1969), and classify the properties of $activity's as mental, social and physical.

The mental properties of an activity break down into two classes: a) specification of an $activity's context, and b) specification of which %activity's plan a given $activity. As described in Memo 6 the context of an $activity is roughly the information needed to specify it unambigously. For instance, an instantiation of an enabling context relation such as

```
    (%a-r-o %has-context $selling $responsibility-center
                  $responsibility-center-set $commodity-set)
```

might be

```
    (%has-context $a&t-store-selling $a&t-store
                            $a&t-customer $a&t-store-item)
```

which would establish $a&t-store-selling as the $activity whereby $a&t-store's sell $a&t-store-item's to $a&t-customer's. The planning property is a relation between two $activity's; e.g., if $a1 and $a2 are $activity's,

```
                        (%planner-of $a1 $a2)
```

states that $a2 is planned (at some as yet unspecified level) by $a1.

The social properties of an $activity are concerned with responsibility and control. The basic enabling relationships here are

(%a-r-o %authorizer-of $responsibility-center $activity),
(%a-r-o %supervisor-of $responsibility-center $activity),
(%a-r-o %authorizing-document-of $document $activity), and
(%a-r-o %control-document-of $document $activity).

Thus one responsibility center may authorize an activity and is therefore responsible for its effects, while another may be responsible for how it occurs. The relationships involving $documents relate the causes of occurrences of relations to the more physical mechanisms given below.

Social properties may also enter at another level. The legal profession uses the language of contracts as a meta-language to interpret daily affairs; thus, in business affairs, "ordering" both generates an order and offers a contract concerning that order. We capture these notions with relations such as the following:

(%a-r-o %offered-by $contract $activity),
(%a-r-o %accepted-by $contract $activity), and
(%a-r-o %fulfilled-by $contract $activity).

Physical properties of activities include such things as time and location which were described earlier. However, descriptions of the inputs needed for activities to take place as well as outputs produced by activities are also needed. Inputs and outputs are described at two different levels: a more abstract level which gives various inputs and outputs regardless of whatever physical form, grouping or format they take, and a more concrete level which gives such things as physical aggregations of inputs and outputs, document formats, etc. At the abstract level we have such relationships as

(%a-r-o %input-data-item-of $data-item $activity),

```
(%a-r-o %input-ingredient-of $ingredient $activity),
(%a-r-o %input-component-of $component $activity),
(%a-r-o %generated-by $data-item $activity),
(%a-r-o %output-data-item-of $data-item $activity),
(%a-r-o %output-commodity-of $commodity $activity),
(%a-r-o %equipment-used-by $equipment $activity),
(%a-r-o %supply-used-by $supply $activity),
(%a-r-o %performer-of $job-title $activity), etc.
```

At the level closer to physical reality we have

```
(%a-r-o %output-lot-of $lot $activity),
(%a-r-o %output-batch-of $batch $activity),
(%a-r-o %input-information-flow-of $information-flow $activity),
(a-r-o %output-information-flow-of $information-flow $activity),
(%a-r-o %labor-requirement-of $man-hours $activity),
(%a-r-o %equipment-requirement-of $machine-hours $activity),
(%a-r-o %supply-requirement-of $supply-measure $activity), etc.
```

Other similar relationships will be defined as the %a-k-o relation becomes developed for various sections and situation of the WOB.

## 5.2 Data Items

The set %data-item is a subset of %object in the WOB which includes all kinds of named quantities that are used in business. For example, %total-sales, %depreciation-expense, %depreciation-period, %cash-on-hand, %a&t-store-daily-sales, %loan-amount, %inventory-reorder-point, etc. are all subsets of %data-item. In terms of our earlier classifications nearly all %data-item's are kinds of %ratio-measure's and may have $dollar as a $unit-of-measure. A $data-item may be produced by some $activity -- i.e., it may participate in an $output-data-item-of or $generated-by relation; e.g.,

```
(%generated-by $sales $selling).
```

In addition, a $data-item may be defined in terms of other $data-item's by algebraic operators or functions; thus,

(%a-r-o %plus-of $data-item-i $data-item-j $data-item-j)

enables such statements as

(%plus-of $total-cost $direct-cost $overhead),

where $data-item-i is restricted to have the same units and specifications of the $data-item-j's. We similarly define

1) %minus-of,
2) %times-of,
3) %quotient-of,
4) %greater-then,
5) %less-than, and
6) %equal-to.

$data-item's have properties and characteristics. If a $data-item is generated or outputted by an $activity then it inherits several properties and characteristics from that $activity. Especially important here are time characteristics; $data-item's generated by recurrent events can be considered to be indexed on the occurrences of those events. A set of $data-item's indexed on a set of $time-interval's of $time-point's forms a $time-series and as such can have properties normally given to a time series such as $level, $correlation, etc. For example,

(%level-of $highest $store-sales).

Without qualification such a statement would mean that $store-sales has the $highest value on the characteristic %level-of at all times that the system knows about. Therefore further specification is necessary to convey such information as when $store-sales are $highest; thus, to say that $store-sales are $highest during weekend specials we state

```
            (%h-r-o-f %rl (%level-of $highest $store-sales),
                    (%t-a-t-of $truth-rl $ti-rl $rl),
     (%time-interval-referrent-of $at-same-time-as $ti-rl $ti-wer),
```

where

```
        (%h-r-o-f %wer %a-k-o $weekend-special $store-selling),
                  (%t-a-t-of $truth-wer $ti-wer $wer)
```

and further relationships specify $ti-wer. Note that we could have given specifications for $ti-rl directly. Other relations which are used in this way are %variability-of, %trend-of, %rate-of-increase-of, %rate-of-decrease-of, %average-of, and %accumulated-total-of.

A third class of functions on $data-item's correspond to some of the DSSL subsetting functions described in Memo 3:

1) %subsetplus-of,
2) %subsetmax-of,
3) %subsetmin-of,
4) %subsetcount-of, and
5) %subsetaverage-of.

They are defined as

```
     (%a-r-o %subsetplus-of $data-item-i %data-item-j %index-set)
```

in which the meaning is that $data-item-i is derived from the set of %data-item-j's by adding up each $data-item-j for each element of the %index-set. As an example

```
    (%subsetplus-of $eastern-region-sales %store-sales
                                          %eastern-region-stores)
```

indicates that to calculate $eastern-region-sales values of $store-sales are added for each store in the eastern region. Presumably %eastern-region-stores are %a-k-o %stores and $store-sales are indexed on $stores. The %index-set could contain time values in which case the $data-item-i is an accumulated total.

Other functions such as %correlation-of, %skewness-of, %seasnality-of, etc., will be defined as needed as the deduction modules regarding data items are developed and refined.

## References

Findler, Nicolas V.. and David Chen, "On the Problems of Time, Retrieval of Temporal Relations, Causality and Co-existance", in Proceedings of the Second International Joint Conference on Artificial Intelligence, Imperial College, London: September, 1971.

Ginzberg, Michael J., "Status of the Simulator in Protosystem 1", Internal Memo No. 3, Automatic Programming Group, Project MAC, M.I.T.: July 24, 1972.

Martin, William A., "Interactive Design in Protosystem 1", Internal Memo No. 4, Automatic Programming Group, Project MAC, M.I.T.: August 21, 1972.

Martin, Wiliam A., and Rand B. Krumland, "MAPL: A Language for Describing Models of the World", Internal Memo No. 6, Automatic Programming Group, project MAC, M.I.T.: October 17, 1972.

McCarthy, J., and P. J. Hayes, "Some Philosophic Problems from the Standpoint of Artificial Intelligence", in Machine Intelligence 4, B. Meltzer and D. Michie (eds.), American Elsevier Publishing Co., Inc., New York: 1969.

McDermott, Drew, "How to Change the World", unpublished paper, Artificial Intelligence Laboratory, M.I.T.: November, 1972.

Minsky, Marvin L., "Matter, Mind, and Models", in Semantic Information Processing, Marvin Minsky (ed.), M.I.T. Press, Cambridge, MA: 1968.

Quillian, M. Ross, "A Teachable Language Comprehender: A Simulation Program and Theory of Language", Communications of the ACM, Vol. 12, No. 8: August, 1969.

Sandewall, Erik J., "Representing Natural Language Information in Predicate Calculus", Memo AIM-128, Stanford Artificial Intelligence Project, Stanford University, Stanford, CA: July, 1970.

Schank, Roger C., and Larry Tesler, "A Conceptual Parser for Natural Language", in Proceedings of the International Joint Conference on Artificial Intelligence, Donald E. Walker and Lewis M. Norton (eds.), Washington, D. C.: May, 1969.

Schank, Roger C., Larry Tesler and Sylvia Weber, "Spinoza II: Conceptual Case-Based Natural Language Analysis", Memo AIM-109, Stanford Artificial Intelligence Project, Stanford University, Stanford,CA: January, 1970.

Siegel, Sidney, Nonparametric Statistics for the Behavioral Sciences, McGraw-Hill Book Co., Inc., New York: 1956.

Winograd, Terry, "Procedures as a Representation for Data in a Computer Program for Understanding Natural Language", TR-84, Project MAC, MIT, Cambridge, MA: February, 1971.

Winston, Patrick H., "Learning Structural Descriptions from Examples", AI TR-231 (Ph.D. Thesis), Artificial Intelligence Laboratory, MIT, Cambridge, MA: September, 1970.