A Methodology and Interactive Environment for Iconic Language Design

S. K. Chang, G. Polese

Department of Computer Science
University of Pittsburgh
Pittsburgh, PA 15260

S. Orefice, M. Tucci

Dipartimento di Informatica ed Applicazioni
Universita' degli Studi di Salerno
Baronissi (SA), Italy

ABSTRACT

We describe a design methodology for iconic languages based upon the theory of icon algebra to derive the meaning of iconic sentences. The design methodology serves two purposes. First of all, it is a descriptive model for the design process of the iconic languages used in the Minspeak™ systems for augmentative communication.Second, it is also a prescriptive model for the design of other iconic languages for human-machine interface. An interactive design environment based upon this methodology is described. This investigation raises a number of interesting issues regarding iconic languages and iconic communications.

1. Introduction

Iconic languages are visual languages where each visual sentence is a spatial arrangement of icons. An essential feature of the iconic languages is that such languages are based upon a vocabulary of icons where each icon has unique or multiple meanings. These languages have been used successfully in human-computer interface, visual programming, and human-human communication. The iconic language used in human-computer communication usually has a limited vocabulary of icons and a specific application domain: database access, form manipulation, image processing, etc. There are also iconic languages for human-human communication used in *augmentative communication* by people with speech disabilities. Finally, there are "natural" iconic languages such as the Chinese ideographs, the Mayan glyphs and the Egyptian pictograms.

In this paper we present a design methodology for iconic languages. The design methodology serves two purposes. First of all, it is a descriptive model for the design process of the iconic languages used in the Minspeak™ systems for augmentative communication.Second, it is also a prescriptive model for the design of other iconic languages for human-machine interface. We hope the experience learned from the case study of iconic language design for the Minspeak™ systems will lead to some valuable insightin the design of iconic languages in general. We do not claim that iconic languages are adequate or even appropriate for all purposes. However, if there is a need for designing an iconic language, the design methodology presented in this paper can be useful.

There are a variety of augmentative communication systems for people with physical limitations or speech impediments, ranging from unaided communication such as American Sign Language, to aided communication systems such as computerized voice output systems. The Minspeak™ systems conceived by Bruce Baker use the principle of *semantic compaction* Baker Barry Disabled Individuals Bray Baker Nyberg It involves mapping concepts on to multimeaning icon sentences and using these icon sentences to retrieve messages stored in the memory of a microcomputer. The stored messages can be words or word sequences. A built-in speech synthesizer can then be used to generate the voice output. Over the past ten years, more than 20,000 Minspeak™ units have been distributed all over the world. Swedish, German, Italian and other Minspeak ™ systems are being developed.

The Minspeak™ Iconic keyboard is shown in Figure 1.When the user touches the icons on the keyboard, the system produces the voice output. Thus the Minspeak™ keyboard can serve as an augmentative communication system. For example, when the APPLE icon and the VERB icon are depressed in that order, the system produces the voice output "eat". The sequence "APPLE VERB" is called an *iconic sentence*. A different iconic sentence such as "APPLE NOUN" will produce the voice output "food". The APPLE icon by itself is thus ambiguous. The basic idea of *semantic compaction* is to use ambiguous icons to represent concepts. For example, the APPLE icon can represent "eat" or "food". Ambiguity is resolved, when several icons are combined into an iconic sentence. This principle allows the representation of many concepts (usually around two thousand) using a few icons (usually around fifty). It is important to note that the user can enter a concept with just a few keystrokes, instead of entering every letter of the word. The design of the Minspeak™ systems therefore involves the encodingof concepts into iconic sentences.

Our approach is to formalize the methodology to design iconic languages based upon the theory of *Icon Algebra* to derive the meaning of iconic sentences. The formalization of the design methodology will lead to a deeper understanding of iconic languages and iconic communications, so that certain theoretical issues can be further explored in this framework.

The paper is organized as follows. Section 2 introduces the theory of *Icon Algebra* Chang Icon Semantics - A Formal Approach Chang Principles of Pictorial Information Systems as a formal approach for deriving the meaning of iconic sentences. Section 3 outlines the design methodology for iconic languages including in particular the Minspeak™ systems. Based upon this design methodology,an interactive design environment, which has been implemented on IBM PC, is presented in Section 4. Since knowledge representation is critical to the success of the design methodology, we explain in Section 5 how the frame-based representation can be augmented by the theory of *Conceptual Dependency* to serve as the semantic model of iconic sentences. The algorithm for making inferences using the icon algebra and the augmented frame representation is presented in Section 6. In the design process, a critical issue is the construction of the conceptual similarity function, which is presented in Section 7. Section 8 describes the design methodology in detail and also gives a detailed example based upon the Minspeak™systems. In Section 9, we discuss a number of interesting issues for further exploration.

2. The Icon Algebra for Deriving Icon Semantics

As a formal model to describe the semantic combination of icons we have exploited the theory of *Icon Algebra* Chang Icon Semantics - A Formal Approach Chang Principles of Pictorial Information Systems Icon algebra provides a powerful way to derive new meanings of an icon, or an iconic sentence, by applying some formal operators on it. An icon X is seen as a pair (X_m,X_i) where X_m represents the meaning of the icon, or the logical part, and X_i represents the image, or the physical part. An essential characteristic of an icon is that the logical part and the physical part are mutually dependent. That is, if the image is changed, its meaning will also be changed and vice versa. When more than one icon is employed to derive the new meaning, all the involved icons together will constitute the iconic sentence to encode the newly derived concept. Furthermore, an icon image contains both global and local features. A global feature of an icon represents the primary concept expressed by the image, whereas a local feature represents a secondary concept. Therefore, the meaning part X_m of an icon X is in general a conceptual structure.

As mentioned above, the icon algebra applies some formal operators to icons to derive new icons. The icon operators are defined below, where X and Y are the operand icons and Z is the resultant composite icon. In what follows, X.[u]A means X has an attribute u whose value is A. If A has again an attribute v whose value is B, we write X.[u]A.[v]B. The primary meaning P of the icon X is usually denoted by X.[is]P, although any X.[u]P could be made to be the primary meaning of X. We often use the primary meaning P to refer to the conceptual structure X_m, and write $X = (P, X_i)$.The six icon operators - combination COM, marking MAR, contextual interpretation CON, enhancement ENH, inversion INV and indexing IDX - will now be explained.

1. Combination COM: COM (X,Y)

(Figure of COM operator in minsfig/com.ps)

The COM operator performs the *conceptual merge* of the meanings associated with the individual icons X and Y. In the example above, the combination of "jump" and "volt" yields "hurry" because they both have the secondary meaning "hurry" among their possible meanings, although this is not the primary meaning in either of them. Therefore only their combination leads to the concept "hurry". Formally, X.[is]jump.[recall]fast combined with Y.[is]volt.[quality]fast results in Z.[is]hurry. (In the above, "jump" leads one to recall the concept "fast", "volt" has the quality "fast", and the resultant composite icon has the primary meaning "hurry".) The conceptual merge is possible only when "jump" and "volt" or their derived attributes are in the conceptual structure of "hurry". In other words, the two concepts are somehow related within the conceptual structure of the resultant concept.

2. Marking MAR: MAR (X,Y)

The marking operator marks the image of the icon Y with the image of the icon X to emphasize a local feature. Here the first icon plays the role of "marker image". For example:

(Figure of MAR operator in minsfig/mar.ps)

Inside the "chest" there is "treasure", and the "color" of "treasure" is "gold". Since "rainbow" means "color", the marking of "chest" by "color" results in "gold". Formally, Y.[is]chest.[quality]treasure.[color]gold marked by X.[is]rainbow.[recall]color results in Z.[is]gold. Using the Chinese character as another example, Y.[is]tree.[part]root.[location]low marked by X.[is]low_marker.[recall]low results in Z.[is]root. Thus marking is a *conceptual restriction* to extract an important local feature.

3. Contextual Interpretation CON: CON (X,Y)

The meaning of the icon X is considered in the context of Y, and the result usually is a *conceptual refinement* (specialization) of the meaning of X. For example:

(Figure of CON operator in minsfig/con.ps)

Since "apple" recalls the concept of "food", "food" in the "morning" leads to "breakfast". Therefore X.[is]apple.[is_a_concrete]food in the context of Y.[time]morning results in Z.[is]breakfast, and "breakfast" is a subclass of "food" in the hierarchy.

4. Enhancement ENH: ENH (X,Y)

This operator enhances the conceptual richness of the icon X by adding attributes from Y, and the result usually is an enrichment of the meaning of X. For example:

(Figure of ENH operator in minsfig/enh.ps)

Since "low temperature" corresponds to "cold", X.[is]thermometer.[use]temperature enhanced by Y.[is]thumb_down.[recall]low leads to Z.[is]cold. Enhancement is similar to combination, in that the two concepts must be somehow related. However, for the combination operator, the two icons contribute equally and are both indispensable. For the enhancement operator, the first icon plays the more important role, and the second icon is usually a modifier. For example, the second icon can be replaced by Y.[is]low_marker.[recall]low, and the result would be the same.

5. Inversion INV: INV (X)

The meaning of the icon X is inverted. The inversion operator has only one argument. For example:

(Figure of INV operator in minsfig/inv.ps)

God stands for "true", so the negation of "true" is "false". In the Minspeak^{TM} iconic language, the icon"knot" stands for negation because it is homonymous to "not". Thus X.[is]god.[recall]true with the negation operator Y.[is]knot.[rhyme]not leads to Z.[is]false.

6. Indexing IDX: IDX (X)

The index operator extracts the important meaning of an icon that will serve as an index to the original icon. For example:

(Figure of IDX operator in minsfig/idx.ps)

The icon X may have multiple attributes, but one important attribute is "big", which is extracted to become the primary meaning of the new icon Y. Later on we will use a fuzzy semantic parameter to express the importance of an attribute.

In addition to the six iconic operators described above, there are other operators Chang Principles of Pictorial Information Systems Design so that both the meaning and the image of the composite icon can be modified.

3. A Design Methodology for Iconic Languages

The objective of the design methodology can be stated as follows. Let K be aset of words (for example part of the vocabulary of a natural language, as in the case of the Minspeak, or the set of feasible commands and entities in some application domain), the objective is to design an iconic language for K, i.e., a set of icons I such thateach word in K is encoded by an iconic sentence with at most *max* icons from I, where *max* is a predefined parameter.The encoding should be accomplished such that each word in the application domain is associated with a visual sentence that evokes the meaning of the word.

The design methodology consists of three major steps. In what follows, we will give a brief description of the design process, which will be treated in more detail in Section 8.

The first step is to find the basic words in K, i.e., those very important words whichshould be represented directly by icon images. This subset of basic words, K_B,can be obtained by opportunely partitioning K and then picking upthe most relevant words from each set of the partition according to the following criterion of relevancy. The *relevancy* of a concept is a composite measure, including the frequency of usage of the concept, the intrinsic importance of the concept, and how the concept is likely to cover the meaning of other concepts. The formula we use to compute the relevancy r_j of a word k_j is as follows:

$r_j = 1$ if $p >= t$

$r_j = w_1 f + w_2 i + w_3 p$ otherwise
where:
f is the frequency of the word, f in [0,1],
i is the intrinsic importance of the word, i in [0,1],
p indicates how likely will the word cover the meaning
of other words, p in [0,1],
t is a fixed threshold in [0,1], normally close to 1,
w_k are weights indicating the contribution of the corresponding feature to the relevancy in the domain specification, $w_k >= 0$ and $w_1+w_2+w_3=1$.

The second step is to invent an icon image for each basic word in K_B. The image of the icon associated with a basic word should immediatelyevoke the meaning of the word. Moreover, the graphics can be enriched in order to cover other possible meanings. These icons are included in the set of basic icons I. In addition, I can be augmented by a few more icons depending on the application domain. Forexample, in the Minspeak^{TM} iconic language I also contains some icons representing the syntacticcategories C = {noun, adjective, verb, ...}.

Finally, in the third step the encoding of words into iconic sentences is accomplished. For each word k_i to encode,the designer must choose one iconic sentence from a set of candidate sentences with similar meanings to encode the word. If a failure in the encoding process occurs, the designer first attempts to redesign the icons without changing K_B, because maybe the icon images are not appropriate. If anencoding still cannot be found, the designer then modifies K_B.

The above design approach requires a knowledge representation to describe the semantics of words, icons and iconic sentences, and a conceptual similarity function to measure the similarity between words (or concepts), icons and iconic sentences. The knowledge representation will be described in Section 5. The conceptual similarity function serves as a metric to partition the set of words K and to perform the encoding of the words. It willbe described in Section 7.

4. An Interactive Design Environment

From the previous sections, it becomes clear that to design iconic languages, we need to provide the models and tools for representing the semantic knowledge of iconic languages, applying powerful inference mechanisms capable of interpreting the underlying meanings, combining icons in order to infer new meanings, and giving explanations regarding the rationale used during such inferences. The designer can then employ these models and tools to design a customized iconic language.

In this section we describe the architecture of such an interactive design environment. The meanings of individual icons, as we will discuss in more detail in Section 5, can be represented through a frame structure with semantic attributes created according to the metaphors appropriate for the context of this visual language. The iconic sentence with more than just one icon can still be represented by the same structure, but it turns out to be very difficult to determine its attributes. Indeed, the only thing that can be automatically derived from the semantics of the icons in an iconic sentence is a list of values belonging to the frames of these icons. The problem is that individual icons in the iconic sentence provide only a portion of the semantics. The rest of it is a result of the combination of these icons, and cannot be detected without a global semantic analysis of the sentence.

In order to perform such a kind of analysis, the interaction of the designer, at first, seems to be of vital importance. But at the same time, the nature of the problem suggests that a lot of knowledge involved in that process can be formalized, together with a set of iconic operators that will support general semantic inferences, in order to derive all the possible meanings of the iconic sentences and submit them for the approval by the designer. In this way, the designer can perform a more effective semantic analysis by interacting with an environment that would allow the exploration of a number of solutions that might not have come to mind intuitively.

Our approach gives a more complete representation of the knowledge in an individual icon, representing not only static knowledge but also the dynamic aspects of it. The interactive design environment provides a set of formal iconic operators to perform inferences and derive possible meanings of iconic sentences, starting from the knowledge available from the component icons.

The architecture of the interactive design environment, which has been implemented on an IBM PC, is shown in Figure 2. It consists of a knowledge base, an inference engine, the SIL iconic system Chang Tauber and the user interface. In the knowledge base we store the semantics of icons and iconic operators. The inference engine applies the iconic operators to derive new meanings of the iconic sentence which is given as input through the SIL's visual sentence parser. The user interface allows the user to provide feedback to the system in choosing the possible meanings of icons and iconic sentences.

5. Knowledge Representation

When choosing a model for representing knowledge for iconic languages, we should consider the following. We need to provide a unified representation for words, icons and iconic sentences, which are all considered as *objects*. The representation should allow ambiguity because we want to capture all the possible meanings of an icon according to several metaphors. Moreover, each object carries both dynamic and static knowledge. This is particularly true for the icons where each associated action must be represented in some way. Finally, the knowledge representation should have characteristics that will facilitate an inference algorithm (see Section 6) to perform the inferences that are needed.

A more refined model should also be capable of classifying the meanings associated with an object in a hierarchical way, because a concept at a certain abstraction level can be more descriptive for the object than another one. For example, the object "orange" is both an object with a circular shape and an object with the color "orange", but the latter is often more descriptive for the object "orange". Hence, we need to provide a parameter indicating how appropriate is a semantic interpretation in describing a given object. Also, each icon should have a (preferably unique) primary meaning associated with it, which serves as an index for the icon.

In order to achieve the above, we use an approach for knowledge representation, combining the frame-based representation with features of the *theory of conceptual dependency* Conceptual Dependency: A theory of natural language understanding

The frame based representation is very suitable, because describing an icon through a set of attributes is a natural way to represent knowledge carried by the icon. On the other hand, as mentioned above, we need also to represent the dynamics that are depicted in the icon, because they can be very helpful in compacting a large amount of knowledge in the icon. The theory of conceptual dependency provides the means to represent the conceptual meanings of sentences expressed in a textual language, capturing both the static and the dynamic aspects. Since an iconic sentence can be regarded as a visual representation of a sentence in a textual language, both visual and textual languages have underlying conceptual meanings to be captured and opportunely represented. Therefore, in both disciplines the common goal is to have formal structures for representing knowledge in such a way that supports manipulation and inferencing to detect similarities in meanings and conceptual relationships among these structures.

In what follows, we explain in detail how the frame-based representation can be augmented by the Conceptual Dependency theory for representing knowledge for iconic languages.

5.1. The Frame-Based Representation

The knowledge underlying an icon is represented using not only conceptual meanings but also many other kinds of metaphors depending on the particular context of the visual language in question. According to these metaphors, we set one or more attributes in a frame structure which will be used to describe the semantics of an icon. In the case of the Minspeak^{TM} visual languages we have made use of the metaphors from the *Types of Semantic Relationships* diagram Baker, Models and Semantic Relationships and directly translated those metaphors into a frame structure. For instance, the metaphor "quantity" can be expressed through the two attributes "mass_quantity" and "count_quantity". If we want to express the knowledge in the icon "elephant", among the various attributes, we can fill the attribute "mass_quantity" with the value "big". In the following, the frame structure representing part of the semantics of an icon is given. Only the macro-attributes at the first nested level are included to clearly illustrate the correspondence with the metaphors.

```
. SOUND    . ACTIVITY
 .. Alphabetic abbreviation  .. Use
 .. Sound
 .. Rhyme    . EMOTION
 .. Recall    .. What does it  recall

. TIME     . CONVENTION
 .. Time sequencing    .. Linguistic
    .. Cultural
. SHAPE
 .. Type of Shape   . QUANTITY
    .. Mass
. COLOR    .. Count
 .. Color
    . IS_A
. LOCATION     .. Abstract
 .. Where    .. Concrete

. QUALITY
 .. Quality
```

We will use this structure for representing knowledge about different objects: words to be encoded, icons, or iconic sentences. The frame of an object is filled with the values of the attributes describing the object, according to a set of metaphors, together with a *fuzzy parameter* in the range [0,1], denoted by Ω. More precisely, this semantic parameter indicates the importance and appropriateness of the value of the attribute for describing the given object. In the previous example on the object "orange", the Ω parameter for the value "orange" in the attribute "color" should be higher than the Ω parameter for the value "circular" in the attribute "shape", because it is more intuitive and appropriate to think of an orange as an object having orange color than as an object having circular shape.

In the following, we present three example of frames, where the icons THERMOMETER and MOUNTAIN appear in the keyboard shown in Figure 1, and the word COLD is a derived concept.

Example 1: icon THERMOMETER

```
 shape:   bar    0.8
 color:   red    0.8
 location:  wall    0.6
    emergency box   0.7
    weather bureau   0.7
 quality:  mercury    0.6
 use:   takes the temperature  0.9
 what does it recall: flu    0.9
    warm    0.8
    cold    0.8
 cultural conv.:  forecast   0.6
 mass:   normal    0.2
 quantity:  single    0.6
 is_a_concrete:  tool for temperature  0.9
```

Example 2: word COLD

```
 time sequencing: winter    0.9
 color:   white    0.9
 location:  north    0.8
 use:   to keep    0.6
 what does it recall: snow    0.8
    cold    0.9
    Christmas   0.8
```

```
    death    0.7

 is_an_abstract  cold    0.8
```

Example 3: icon MOUNTAIN

```
 alphabetical abbr.: MT    0.7

 sound:   echo    0.6

    silence    0.7

 rhyme:   mounting   0.6

 time sequencing: winter    0.9

    summer    0.7

 shape:     pyramid    0.6

    triangle   0.4

 color:   white    0.7

    green    0.6

 quality:  rock    0.6

    snow    0.6

 use:   to ski    0.7

    to relax   0.7

    to explore   0.5

 what does it recall: clean air   0.8

    cold    0.9

    fatigue    0.8

    sport    0.8

    majestic   0.7

 linguistic conv.: Mohammed   0.5

 mass:   very big   0.7

 quantity:  two    0.6

 is_an_abstract:  stately person   0.7

 is_a_concrete:  peak    0.8
```

5.2. Conceptual Dependency

Conceptual Dependency (CD) is a theory of natural language and of natural language processing Conceptual Dependency: A theory of natural language understanding language processing. It was created by Schank and can be employed for the construction of computer programs capable of understanding sentences of natural language, summarizing them, translating them into another language and answering questions about them. The basic axiom of the theory is:

For any two sentences that are identical in meaning, regardless of language, there should be only one representation.

Thus, any information in the sentence that is implicit must be made explicit in the representation of the meaning for that sentence.

Through this theory, understanding of concepts is performed by mapping linear strings of words into conceptual structures. A conceptual structure is a kind of semantic net. It is defined as a network of concepts, where some classes of concepts may have specific relationships with other classes of concepts.

The meaning of a linguistic proposition is called a conceptualization or CD form. A conceptualization can be active or stative. An active conceptualization consists of the following slots: actor, action, object, and direction. The latter is subdivided into source and destination. A stative conceptualization consists of the following slots: object, state, and value.

From this theory of Conceptual Dependency, we have derived rules and CD forms suitable for our purpose. In our adaptation of this theory, rules for concepts combination use the following conceptual categories Conceptual Dependency: A theory of natural language understanding

PPs: Picture Producers. Only physical objects are PPs. They may serve in the role of objects as well as source, destination and recipient of actions.

ACTs: Actions. Actions can be done by an actor to an object.

LOCs: Locations. They are considered to be coordinates in space and can modify conceptualizations as well as serve as sources and destinations.

Ts: TIMES. The time is considered to be a point or a segment on a time line.

AAs: Action Aiders. AAs are modifications of features of an action.

PAs: Picture Aides, or Attributes of an Object. A PA is an attribute characteristic such as color or size plus a value for that characteristic, a number of them can be used for describing a physical object.

Here are some rules through which classes of concepts can combine Conceptual Dependency: A theory of natural language understanding Identification of Conceptualization Underlying Natural Language

Rule 1. Certain PPs Can ACT. For example, the sentence "Kevin walked" may be represented using the primitive act PTRANS (A list of primitive actions is given later) as

Actor: Kevin
Action: PTRANS
Object: Kevin
Direction: From: Unknown To: Unknown

The kind of ACT and the PP performing it can only be determined in each case by the semantic nature of these two objects.

Rule 2. PPs and Some Conceptualizations Can Be Described By an Attribute . For example, the sentence "Nancy is heavy" may be represented using the following stative conceptualization:

Object: Nancy
State: WEIGHT
Value: Above the Average

Rule 3. ACTs Have Objects. For example, the sentence "Perry Kicked the cat" may be represented using the primitive action PROPEL (Physical force applied, see below) as

Actor: Perry
Action: PROPEL
Object: cat
Direction: From: Unknown To: Unknown

Rule 4. ACTs Have Direction. For example, the sentence "Bill fell from the ladder" may be represented using the primitive action PTRANS (see below) as

Actor: Bill
Action: PTRANS
Object: Bill
Direction: From: ladder To: ground

Rule 5. ACTs Have Recipients. For example, the sentence "John donated blood to the Red Cross" may be represented using the primitive action ATRANS (see below) as

Actor: John
Action: ATRANS
Object: blood
Direction: From: John To: Red Cross

Rule 6. ACTs Can Have Instrumental ACTs. For example, "John hit Bill with his hand" will be represented as

Actor: John
Action: PROPEL
Object: Bill
Instrument: Actor: John
Action: MOVE
Object: hand
Direction: From: John To: Bill

The use of a set of primitive actions reduces the complexity of inferences to be made as the inference rules need only be written once for any ACT rather than many times for each verb that references that ACT. Often an action can be expressed through more than just one verb, but for our purposes we want to get rid of this redundancy and have only a general representation for sentences with the same conceptual meaning. Furthermore, this representation makes more evident the similarities in meaning among sentences than the textual representation does.

In the following a list of the most important primitive ACTs is given Conceptual Dependency: A theory of natural language understanding Identification of Conceptualization Underlaying Natural Language

ATRANS is the transfer of an abstract relationship, such as possession, ownership or control.

PTRANS is the transfer of the physical location of an object.

PROPEL is he application of a physical force to an object.

MOVE is the movement of a body part of an animal by that animal.

GRASP is the grasping of an object by an actor.

INGEST is the taking of an object by an animal to the inside of that animal.

EXPEL is the expulsion of an object from the body of an animal into the physical world.

MTRANS is the transfer of mental information between animals or within an animal.

MBUILD is the construction by an animal of new information from old information.

SPEAK is the action of producing sounds. ATTEND is the action of attending or focusing a sense organ toward a stimulus. For example the verb "listen" means ATTEND ear, and the verb "see" means ATTEND eye.

As far as stative CD forms are concerned, they can be used for representing statements requiring stative conceptualization. Attached to each attribute-value pair we may have scales to show the range of the value for that attribute.

5.3. The Augmented Frame-Based Representation

As discussed above, compacting semantic information in an icon is accomplished by adding some meaningful dynamic aspects to the icon image. For example, if an icon image of the sun was designed to represent words such as "yellow" and "bright", we may enrich the icon by designing a rising sun to encompass also some other words such as "dawn", "morning" and "start". The newly added meanings can be represented in the knowledge base in the same way as CD theory would represent the sentence "The sun is rising" by using the primitive action PTRANS

Conceptual Dependency: A theory of natural language understanding as follows:

Actor: Sun
Action: PTRANS
Object: Sun
Direction: From: Sea to: Sky

Such a structure can be integrated into the frame structure described above by adding some more slots. These new attributes are very useful to infer additional meanings about the icon, such as "The sun is rising to the sky" which will yield the new value "sky" for the attribute "where", thus allowing the designer to encode additional concepts and words belonging to the given context. For the example of the rising_sun icon, other than filling the dynamic attributes as discussed above, the semantic knowledge is represented by the attributes of the frame structure for the rising_sun which now also includes the new value "sky" inferred by the presence of the CD attributes:

Color: Yellow
Time : Morning
Shape: Circular
Mass : Big
Use : Energy
Where: Sky

We can complete the semantic knowledge by adding the semantic parameter values.

The construction of the frame (including the CD form slots) for each single word to be encoded by the visual language, is done by the designer. The frame for each single icon is obtained by inheriting the frame of the word directly associated with the icon Deriving the Meaning of Iconic Sentences with some more information added due to the additional meanings conveyed by the icon image. As far as the frame for an iconic sentence is concerned, we need to perform inferences on the augmented frames (basic frames plus CD slots with newly inferred attribute values) of the icons composing the sentences. In Section 6 we will discuss the formal criteria to perform such inferences. However, having the augmented frames of the single icons available, we can already derive a skeleton of the frame for the iconic sentence using the following algorithm. The input are the frames of two icons v_i and v_j,and the output is the frame of the iconic sentence v_iv_j.

```
Procedure Construct_Frame(frame(v_i),frame(v_j),frame(v_iv_j))

{

for each matching value do

 if  the matching value appears in the frames

    input the common attribute A with Ω

    parameters equal to Ω_i and Ω_j, respectively,

 then fill the attribute A of the resulting frame with

  the matching value and set the Ω parameter to
```

$$\frac{\Omega_i * [1+\frac{|\Omega_i-\Omega_j|}{2}]+\Omega_j * [1-\frac{|\Omega_i-\Omega_j|}{2}]}{2}$$

else if the matching value appears in the frames

input two different attributes A and B with Ω parameters equal to Ω_i and Ω_j, respectively

then fill the attributes A and B of the resulting

frame with the matching value and set the Ω parameter to

$$\frac{\Omega_i * [1+\frac{|\Omega_i-\Omega_j|}{2}]+\Omega_j * [1-\frac{|\Omega_i-\Omega_j|}{2}]}{2}$$

let Ω_1 be the minimal Ω parameter computed at this step,
for each value in the frame of v_i and not in the frame of v_j with Ω parameter greater than a threshold D_1 **do** fill each attribute of the resulting frame in which such values appear with the value itself and the Ω parameter multiplied by Ω_1;
let Ω_2 be the minimal Ω parameter computed at this step,
for each value in the frame of v_j and not in the frame of v_i with Ω parameter greater than a threshold D_2 **do** fill each attribute of the resulting frame in which such values appear with the value itself and the Ω parameter multiplied by Ω_2;
}

The above algorithm, applied to the object icons THERMOMETER and MOUNTAIN in both ordering sequences, yields the following frames:

iconic sentence THERMOMETER/MOUNTAIN
time sequencing: winter 0.686475
use: takes the temperature 0.76275
what does it recall: flu 0.76275
cold 0.8475
is_a_concrete: tool for temperature 0.76275
D_1=0.81; D_2=0.85

iconic sentence MOUNTAIN/THERMOMETER
time sequencing: winter 0.76725
use: takes the temperature 0.690525
what does it recall: flu 0.690525
cold 0.8525
is_a_concrete: tool for temperature 0.690525
D_1=0.81; D_2=0.85

In order to obtain a complete representation of the semantics of the iconic sentences that these frames represent, we need to enrich the frames according to the semantics conveyed by the icons composing the sentence as they were viewed as a whole. This will be discussed in the following section.

6. An Inference Algorithm

Our goal here is to build an inference algorithm to derive the meanings associated with iconic sentences from the meanings associated with the individual icons forming the sentences. In order to achieve this goal, we will make use of the Icon Algebra operators and some principles described in the Theory of Conceptual Dependency Conceptual Dependency: A theory of natural language understanding

In our approach, we have formalized the semantics of a set of Icon Algebra operators which, combined with the primary meanings of the component icons, give the possible meanings of the iconic sentence. The derived meanings are submitted to the designer who will decide whether to accept or discard the new meanings. He may also perform some actions such as assigning the Ω parameter or the appropriate slotto fill the new information in.

However, before going into the process of inferring the semantic knowledge of the visual sentence, the inference algorithm may need to perform some inferences on the single icons forming the sentence. In fact, especially for the frame slots for the dynamic knowledge (the CD slots) that may contain some implicit meanings, we need to make that explicit in the frame. In order to do that, we need to attach to each primitive ACT certain inference-generating procedures to compute the consequences of that primitive ACT. For example, some of the consequences for the primitive ACT PTRANS may be computed by the following lisp function Conceptual Dependency: A theory of natural language understanding

```
(Defun PTRANS-CONSEQ()

  (NOTICE $(ACTOR) "CD")

  (ADD-CONSEQ (IS-AT (OBJECT)(TO)))
  (COND ((FROM)(NOTICE(ACTOR)

    (ADD-CONSEQ

     (NEGATE (IS-AT (OBJECT)(FROM))))))))
```

The operator '$' catches the value of the slot to which it is applied. The function NOTICE notifies an object (first argument) that something has happened (second argument). Thus, in its first use, NOTICE notifies the ACTOR that a PTRANS has taken place. The details of this action are described in the CD form given as second argument. The function IS-AT notifies the change of location for the object which has moved. Finally, the COND instruction checks whether the object left from a source (in this case the slot FROM would have a positive filler). If that was the case, the function NEGATE cancels the belonging of the object from the old position. This kind of inference procedures will enrich the frames originally built for the icons. In what follows, we will see how the inference engine will act on these enriched frames to derive the meanings for iconic sentences.

The design of this part of the inference engine is essentially a parser that parses iconic sentences into single frames. The iconic sentences to be parsed are to be viewed as having one of the icon algebra operators in them. The algorithm will parse them, in turn, with all of the operators described in Section 2. For each iconic operator, we can define a set of primitive words whose meaning can be viewed as a textual representation of the meaning of the operator. Put it in another way, these words represent values of attributes describing the operators such that, when combined with attributes describing the icons in the sentences, produce candidate attribute values for the frames being built, e.g. the frames representing the iconic sentences.

An outline of the parsing algorithm is presented in the following.

```
Procedure Parse


{


1. Select a set of meaningful attributes from the frames of the icons
in the sentence.  These will be the ones whose
Ω parameters have values greater than a prefixed threshold. Each time,only one attribute per object icon is selected.


2. for pair of meaningful attributes, one per each frame, and


 for each iconic operator,

    apply the iconic operator to generate a candidate

    attribute value for the frame being constructed;

    submit  the new value to the designer;

}
```

The parser starts by examining index attributes, which are obtained by applying the IDX operator, whose implementation simply provides the attribute value with the maximum **Ω** parameter. At the end of thisstep, every attribute whose value is above the threshold, will have been combined with the corresponding attributes of the other object icons, according to all the icon operators available. For iconic sentences of length greater than 2, the parsing will be recursive, and more than one icon operator may be applied to the sentence. In other words, the parsing is performed by applying the first iconic operator to the first two icons in the sentence, then a new iconic operator is applied to the resulting frame and the third icon in the sentence, and so on.

The semantic definitions of the iconic operators are given below.

1. The operator CON of Contextual Interpretation refines the meaning of the first icon in the context of the second icon. Therefore we use the attribute word IN_THE_CONTEXT_OF or briefly IN_THE to describe its semantics. For example, we can derive "Food_IN_THE_CONTEXT_OF_Morning" as a result of applying CON to X.[is]apple.[is_a_concrete]food and Y.[is]rising_sun.[time]morning. At this point, the designer should decide which slot in the frame for the iconic sentence should contain the new value. This new attribute value should enhance the opportunities to encode new words in the language. In fact, a similarity with the frame for the word "breakfast" can be detected using the similarity function introduced in the next section. However, the user may decide to change the new value directly to "breakfast" and put this value in the right slot.

2. The operator COM of Combination may have the word attributes MERGED_WITH, AND, or WITH, because it combines the meanings of two object icons. For example, combining the icons "Jump" and "Volt" by applying COM to X.[is]jump.[recall]fast and Y.[is]volt.[quality]fast, we obtain the slot value "fast_MERGED_WITH_fast" which implicitly means "hurry". Again, the designer can make the meaning explicit by directly assigning it to the object, or by performing the similarity inference. In the later case, the similarity with the frame of the word 'hurry' should be detected.

3. For MAR, the marking operator, we may associate the word attribute AS_MARKER_OF or simply OF, because it uses the first icon as a characteristic of the second icon. Thus for example, the application of MAR to X.[is]rainbow.[recall]color and Y.[is]chest.[quality]treasure will lead to the attribute value "color_OF_treasure". After the marking operation, the similarity with the word "gold" becomes more evident.

4. The operator ENH "enhancement" emphasizes the meaning of the first icon by partially incorporating the meaning of the second icon. So we may associate the word attribute ENHANCED_BY, because it enhances the meaning of the first icon with that of the second icon. Thus enhancing X.[is]thermometer.[use]temperature by Y.[is]thumbs_down.[recall]low will lead to the attribute value "Temperature_ENHANCED_BY_low" which means cold.

5. Finally, the operator INV "Inversion" performs semantic inversion on the icon in the sentence (it is noted that NOT is not binary). We have associated it with the word attributes INVERTED, OPPOSITE_OF, NOT, or WITHOUT, so that by applying INV to X.[is]god.[recall]true which is the index attribute value for "God", we get "OPPOSITE_OF_true", "NOT_true", or "INVERTED_true", which implicitly means false.

7. The Conceptual Similarity Function

As discussed above, one critical issue is to find an adequate similarity function. The techniques for measuring similarity between two objects have been studied in statistics, econometrics, psychology and sociology. These techniques include factor analysis, multi-variate analysis, least squares, classification, component analysis, and clustering Bartholomew Exploring hyperspace Miller Weiss Hunter Bendix Romesburg Classification has been widely studied in information retrieval Chauncey Szalay Sparck Jones

All these techniques are based on the principle of constructing a data matrix in which the individuals to be compared are measured along certain attributes. Each individual can correspond to a row and each column reports the value of an attribute for each individual. These attributes are called the anchors on which the similarity is based.

In this section we will describe a similarity function based on comparisons of attributes. We define a function **g** which takes as input two objects O_i and O_j and returns a value in [0,1] indicating how conceptually similar the objects are. As the meaning of an object is expressed by its frame, the function **g** works on the frames F_i and F_j representing the objects O_i and O_j as described in the previous sections, taking into account the matching values and their **Ω** parameters. As said above, the similarity between two objects is usually based upon the number of the matching values. In our approach, instead, what is important is the significance of the matching values expressed by the **Ω** parameter. Thus, it can be enough even having just one matching value between two objects, but very significant for both, in order to establish a high degree of similarity.

The function **g** will take into account the following three types of matching:

1) Real match. We will say that there is a real match when two objects have the same value in a same attribute of their frame. For example the frames of the words BLOOD and FIRE both have the value "red" in the attribute "color". This is the most important type of matching since both objects can be thought in terms of the same attribute.

2) Diagonal match. We will say that there is a diagonal match when two object have the same value in different attributes of their frame. For example the frame of the word MOUNTAIN has the value "rain" in the attribute "rhyme" while the frame of the word CLOUD has the value "rain" in the attribute "what does it recall". This type of matching is less significant than the real match, but it is anyway important in the comparison of two objects because both objects recall the same meaning.

3) Mismatch. We will say that there is a mismatch when two objects have the same attribute empty. For example the fact that the frames of abstract objects have attributes as "shape" empty could be considered a small index of similarity. Anyway, the mismatch will contribute a very small value to the similarity and basically it serves to distinguish this particular case from the ones in which there is no-match, i.e. different values in the same attribute or just one empty.

Now we are ready to show the formulation of the function **g**. First of all we construct the extended frames F_i^* and F_j^* (on which the function **g** will work) by a simple procedure which, for each value in the frame F_i checks if that value appears in the frame F_j within adifferent attribute introducing some new fictitious attributes in order to allow **g** to work comparing the same attributes. As an example, with reference to the following situation:

	Attribute$_1$	**Attribute$_2$**
object1	val$_1$**Ω**$_1$	val$_2$**Ω**$_2$
object2	val$_1$**Ω**$_3$	val$_1$**Ω**$_4$

The extension will produce:

	Attribute1	Attribute2	Attribute*
object1	val$_1$**Ω**$_1$	val$_2$**Ω**$_2$	val$_1$**Ω**$_1$
object2	val$_1$**Ω**$_3$	val$_1$**Ω**$_4$	val$_1$**Ω**$_4$

The computation of **g** is presented in the algorithm for similarity computation. The heart of the algorithm is the function **d** that basically computes the similarity between two objects with only one matching attribute (single-valued). Actually, if Ω parameters of both objects are high we expect the function to return a high value, whereas if both are low we expect a low value to be returned, and finally if at least one of the two parameters is close to high values we expect a high-medium value to be returned as that value can guarantee a high degree of similarity with other objects sharing that value. Let Ω_{max} and Ω_{min} be the maximum and the minimum of the Ω-parameters associated with the matching values in the two frames, respectively. The function **d** can be the following:

$$\mathbf{d}(\Omega_1,\Omega_2)=\frac{\sqrt{\Omega_{max}*\Omega_{min}}}{[1-\frac{4}{5}(\sqrt{\Omega_{max}}-\sqrt{\Omega_{min}})]*\sqrt[8]{\Omega_{max}}}$$

This function **d** is constructed to achieve the above mentioned results. The function is conceived on a probabilistic base, the product between Ω_{max} and Ω_{min}, the difference between the two values, and the necessity to give a greater weight to Ω_{max}. The following table shows some significant values of the function:

Ω	0.10	0.20	0.30	0.40	0.50	0.60	0.70	0.80	0.90	1.00
.1	.13	.19	.25	.3	.35	.41	.47	.54	.62	.7
.2	.19	.24	.31	.37	.44	.5	.57	.64	.72	.8
.3	.25	.31	.35	.42	.48	.55	.62	.7	.78	.86
.4	.3	.37	.42	.45	.52	.59	.66	.74	.81	.9
.5	.35	.44	.48	.52	.55	.62	.69	.76	.84	.92
.6	.41	.5	.55	.59	.62	.64	.71	.79	.87	.95
.7	.47	.57	.62	.66	.69	.71	.73	.81	.88	.96
.8	.54	.64	.7	.74	.76	.79	.81	.82	.9	.98
.9	.62	.72	.78	.81	.84	.87	.88	.9	.91	.99
1.0	.7	.8	.86	.9	.92	.95	.96	.98	.99	1.0

March 15, 1997

The function $\mathbf{d}^*$ extends the function **d** to the case of one matching attribute, but multi-value and with more internal matching values (for example, the values "red" and "green" in the attribute "color").
The spirit is the same as the global function **g** on all the matching attributes (single and multi-value). That is, we have split the contributions of the matching attributes (resp. the matching values for $\mathbf{d}^*$) into two parts. The first one affects the result more and thus should be multiplied by a larger weight w_2 (resp. w_4) and occurs in corres
weight w_1 (resp. w_3) and adds the averaged contributions of the other matching attributes (matching values for $\mathbf{d}^*$).

Procedure Compute_Similarity
{

1. if $O_i=O_j$ **g** returns 1.

2. Apply the procedure Construct_Extended_Frames to the frames of the two objects in comparison.

3. Compute p = number of matching (real and diagonal) and non-matching attributes of the two extended frames not both empty.

4. Compute the function **g** as follows:

$$g(F_i^*,F_j^*)=w_1*\frac{\sum_{A_r} d^*(A_r(F_i^*),A_r(F_j^*))}{p}+w_2*d^*(A_{max}(F_i^*),A_{max}(F_j^*))$$

(where $A_r(F_i^*)$ indicates the attribute A_r of the frame F_i^*, A_{max} is the attribute which gives the maximum value to $\mathbf{d}^*$ and w_1 and w_2 are weights with sum 1).

Note that, the parameter p in the denominator serves to low the average value in case of unmatch respect with the cases of mismatch and those of diagonal match respect with cases of real match.

5. Compute the function $\mathbf{d}^*(A_k(F_p^*),A_k(F_q^*))$ as follows:

if $|A_k(F_p^*)|=|A_k(F_q^*)|=1$

(i.e., both objects have only one value (v_k)

for the k-th attribute)

if $v_k(F_p^*)=v_k(F_q^*)$ (i.e., there is match)

then

$\mathbf{d}^*(A_k(F_p^*),A_k(F_q^*))=\mathbf{d}(\Omega_{\{vk(F_p^*)\}},\Omega_{\{vk(F_q^*)\}})$

else $\mathbf{d}^*(A_k(F_p^*),A_k(F_q^*))=0$

else {

compute n=|VAL($A_k(F_i)$)cupVAL($A_k(F_j)$)|,

where VAL($A_k(F_w)$)) is the set of the

values of the attribute A_k in the frame F_w;

compute

$$d_1=\frac{\sum_{v\,member\,VAL(A_k),v(F_p^*)=v(F_q^*)} d(\Omega_{v(F_p^*)},\Omega_{v(F_q^*)})}{n}$$

$d_2 = d(\Omega_{v_{max}(F_p^*)}, \Omega_{v_{max}(F_q^*)})$

$\mathbf{d}^*(A_k(F_p^*), A_k(F_q^*)) = w_3 * d_1 + w_4 * d_2$

where v_{max} is the matching value which gives the maximum value to $\mathbf{d}$ and w_3 and w_4 are weights with sum 1.

}

}

8. Refinement of the Design Approach

In this section we describe in detail the design methodology for iconic languages introduced in Section 3. The major steps of the design process are explained to facilitate the understanding of the procedures in Section 8.3.

8.1. Selecting The Basic Set of Words

To begin with, the design problem is to encode a set K of words using a limited numberof icons. The set of icons I must be small enough to enable a user to understandwith relative eas Therefore, we would like to find the minimum number of icons necessary to encode the basic words in the vocabulary K. Our approach consists of finding apartitioning of K which conveniently covers the wholeset of words. Then, the most relevant word is picked up fro partition, forming K_B.

Let us see how it is possible to find such a partitioning. As we associate an icon with each basic word, our aim is to find a partitioning of K that leads to a minimum number of basic words.The number of basic words in K_B is initially set to thesmallest integer n such that:

$$\sum_{i=1}^{n} \binom{n}{i} * i \geq m$$

where m is the total number of words in K. Indeed, it is easy to verify thatchoosing a smaller n we cannot construct enough iconic sentences to encode all the words in K.

Furthermore among the partitions of size n we search for the one that maximizes the sum of the similarities between all the possible pairs of words belonging to the same subset. This strategy is to obtain a partition which conveniently covers the whole domain of words in order for the encoding process to terminate with success. Note that more than one partition can give this maximum value or a value reasonably close to it. Then, among these partitions we select the one that minimizes the variance of the number of elements in each single subset. The last strategy allows us to obtain a partition which is balanced with respect to the number of elements in the single subsets.

8.2. Icon Design

The design of icon images is a crucial step in the process in designing an iconic language. As described in Section 3 we have to draw images for basic concepts and cope with the uncertainty that these basic concepts may have not been selected properly. In other words, as it often happens when applying a a top-down methodology, each step is performed by taking as input the output of the previous step that in turn may need some adjustments according to the feedback coming from the successive steps. Only by repeated iterations over the steps of the methodology can we reach the final solution. Hence, when designing icons for a set of basic words, we have also to detect whether the K_B has been properly designed, that is, verify that covering the remaining words in K is only a matter of adjusting icon images.Often this does not happen in real cases (based upo

We start by trying to design an icon image for each basic word. When doing that, the following requirements have to be taken into account:

(a) The icon should clearly describe the basic word.

(b) The icon should be conceptually rich. It should be created in such a way that when joined to other icons, some other words are covered.

(c) The icon should be related to the way of using it. As an example, in the Minspeak™application, the icon should be related to the alphanumeric character printed on the keyboard.

Furthermore, we need to provide a formal way to accomplish the task. In the following we describe how Icon Algebra can be used for the purpose of designing icons for basic words. We have a set of candidate basic words as input. Then we try to sketch an icon for each of the words in K_B and try to apply the operators ofthe icon algebra to enrich icons and derive new concepts. In doing that, a designer should consider the following issues:

(a) Can this icon be joined with another one to cover a new concept?

(b) Does this image contain a detail that can be emphasized to cover a new concept?

(c) Does the inversion of this image derive a new concept?

Depending on whether the image matches the requirements above, the designer may discard or modify the image.

Finally, we notice that the operators in icon algebra could be applied also to words and not necessarily to icons, so we may think to perform a sort of pre-test on the set K_B to verify if the words in it can guarantee the coverage of the whole domain K.

8.3. Coverage and Encoding

Once the set of icons I for the basic words in K_B have been designed, and the frames for these icons and for the set of feasible iconic sentences I^{max} of length no greater than *max* hav

During the covering process, for each word k_i in $K-K_B$ a set $f(k_i)$ of iconic sentences conceptually similar to it is computed, i.e., those iconic sentences α such that $\mathbf{g}(k_i, \alpha)$ is greater than a predefined threshold. The design of an iconic language is largely determined by the conceptual mapping $f(k_i)$, and it is mostly in this step that the similarity function **g** assists the design, so that the design methodology does not rely only upon the experience of the designer. Moreover, since K could be made up of words belonging to a natural language (as in the case of the Minspeak), we have to consider a further small set of special icons which represent syntactic categories and will be useful for disambiguating in the successive encoding process. For instance, the iconic sentence v_1v_2 may mean both "afraid" and"fear". Thus the extended iconic sentence v_1v_2ADJ means "afraid"(adjective), and the extended iconic sentence v_1v_2NOUN means"fear" (noun). In this case, shortest one. If for a word k_i all the iconic sentences in $f(k_i)$ have already been used to encode previous words, the procedure tries to change one of the codes already assigned in or

A failure in covering process occurs when either at least one set $f(k_i)$is empty or for some k_i, whose $f(k_i)$ is not emptybut all its iconic sentences have already been used, the attemp assigned fails. When a failure occurs we first try to redesign the icon images without changing K_B, because perhaps the icon images are not appropriate. If the number of uncovered words is greater than a predefined threshold, we have to modify K_B by a new partitioning with n+1 elements. In the first case, the processof redesigning the icon images goes on each step if and only if the number of uncovered words decreases. The same process is carried out if the successive encoding attempt fails. Of course, if during this process the number n of basic words, i.e., the number of icons in I,becomes too large, we can revise the thresholds and try to repeat the encoding with a smaller n.

Finally, when the processes of covering and encoding of the words terminate successfully, the iconic language for K is determined.

Let us observe that the design of a new set of icons is a task that an expert designer has to accomplish manually. We have given some general suggestion to assist him/her in the drawing. Conversely, the first and third step can be carried out automatically using the tools provided by the interactive design environment.

The design methodology is outlined below. The procedure Design_Iconic_Language takes as input the words of K and their frames, and returns the encoding function that assigns an iconic sentence in I^{max} to each word in K.A visual grammar G = (I, N, S, P) is also c S is the head symbol, and P is the set of production rules.

Procedure Design_Iconic_Language(K, frames)

{ choose the smallest integer n such that

$$\sum_{i=1}^{max} \left\lfloor \frac{n}{i} \right\rfloor * i \geq m$$

repeat

repeat

Determine_K_B;

1. repeat

Design_Icons(K_B);

Encode(K_B);

Construct_Frames(*I*);

Construct_Frames(I^{max});

Cover($K-K_B$);

if the number of empty $f(k_i)$ sets is less than $Threshold_1$

then set $Threshold_1$ to the number of empty $f(k_i)$ sets;

until there are no empty $f(k_i)$ sets or their number is

greater than $Threshold_1$;

n=n+1;

until the number of empty $f(k_i)$ sets is equal to 0;

Encode($K-K_B$);

if the number of uncoded words is less than $Threshold_2$

and greater than 0

then {

set Threshold$_2$ to the number of uncoded words;

GO TO 1;

}

until number of uncoded words is 0;

for each iconic sentence α in code

add S->α to production rule set P;

return(code);

}

Procedure Determine_K$_B$

{ construct the set PART containing all the partitions of K with
size n which determine a value of the target function

$$\sum_{(x,y)\,\text{member}\,P_i} g(x,y)$$

P_k={(x,y)/oppEi:(x,y)memberP_i}

belonging to a predefined range with respect to the maximum

value computed; select from PART the partition that minimizes

the following variance:

$$var = \sum_{i=1}^{n} |P_i|^2 - \left(\sum_{i=1}^{n} \frac{|P_i|}{n} \right)^2$$

where $|P_i|$ is the number of words belonging to the i-th subset
of the partition;

}

Procedure Design_Icons(K$_B$)

{ **for** each basic word k_i in K_B

design an icon image v_i as suggested above;

}

Procedure Encode(K$_B$)

{ **for** each basic word k_i in K_B

set code(k_i) = v_i;}

Procedure Construct_Frames(*I*)

{ construct the frames for the icons in *I*,

or for the iconic sentences in I^{max},
as described in Sections 5 and 6.

The procedure Construct_Frame described in Section 5 constructs a

skeleton of the frame of an iconic sentence. This skeleton is

then augmented using the inference technique described in Section 6.}

Procedure Cover(K-K$_B$)

{

for each word k_i in K-K_B

{ f(k_i) = 0;

for each iconic sentence α in I^{max}

if **g**(k_i, α) is greater than a predefined threshold

then insert α in f(k_i);

if k_i belongs to a syntactic category I_{cat} from the set
{verb, noun, ...}

then

for each iconic sentence α in f(k_i)

insert α I_{cat} in f(k_i);

}

}

Procedure Encode(K-K_B)

{ Let FREE be a vector whose elements are associated to iconic
sentences in I^{max} such that FREE[j]=0 means that the iconic sentence associated to the j-th position is not marked;
FREE[h] = 0 for h=1,..., $|I^{max}|$;

i=1; /* i is the index of the i-th word in K-K_B */

$f(T_i) = f(k_i)$;

repeat

2. if $f(T_i)$ is not empty **then**

{ choose the iconic sentence α in $f(T_i)$ with maximum

g(k_i, α)

if α is associated to FREE[s] = 0 **then**

{ code(k_i) = α;

FREE[s] = 1; }

else

{ $f(T_i) = f(T_i)-\{\alpha\}$;

GO TO 2; }

}

else

repeat

{ signal =0;

j=i-1;

if code(k_j) = α is in $f(k_i)$ and at least a non marked iconic

sentence is in $f(k_j)$ **then**

{ code(k_i) = α;

choose the iconic sentence α in $f(T_j)$ with maximum

g(k_j, α)

code(k_j) = α;

if α is associated to FREE[s] **then**

{ FREE[s] = 1;

signal = 1; }

}

else

{ j=j-1;

if j=0 **then**

{ insert k_i in a set of uncoded words;

signal=1; }

}

} **until** (signal);

i=i+1;

} **until** i=|K-K_B|+1; }

8.4.
An Example of the Design Approach

The methodology just presented serves two purposes.
First of all, it is a descriptive model of the design process for the
iconic languages of the Minspeak™ systems.Second, it is also a prescriptive model for the design of
other iconic languages for human-machine interface.

In the Minspeak case study, the vocabulary K is formed by the most frequently usedEnglish concepts. The designer will have to provide the semantic frames for these words and the icon Determine_K_B finds the most relevant words. Sincethe Minspeak system is intended for people with speech disabilities, the concepts of K must be encoded by short iconic sequences (usua the efforts of the users. Then, the procedures Cover(K-K_B) and Encode(K-K_B) will be invoked with *max* = 3 to accomplish the encoding.

As an example of the design approach, let us consider the concept "great".
Once the procedure Cover(K-K_B) is called, the set f(great) is filled with all the iconic sentences α such that $\mathbf{g}(great, \alpha)$ exceeds a predefined threshold. So, the set f(great) will "elephants", "apple", "God", "love" and other iconic sentences obtained
by combining these icons and the syntactic category of "great", i.e., "ADJ". At this point the main program will invoke
the procedure Encoding(K-K_B) which tries to find the best code for "great". Table 1
shows the encoding for "great" and for some other concepts involving the icon "elephants"
which is associated with the primary concept "big".

	Key1	Key2	Key3	concept
1	elephants			big
2	elephants	**conj**		and
3	elephants	elephants	**adj**	large
4	elephants	god	**adj**	great
5	elephants	map	**adj**	important
6	elephants	**noun**		couple
7	elephants	**number**		couple
8	elephants	thumbs down	**adj**	weak
9	elephants	thumbs down	**verb**	drop
10	elephants	thumbs up	**adj**	strong
11	elephants	thumbs up	**verb**	lift
12	elephants	**verb**		meet
13	rainbow	elephants	**adj**	gray
14	skull	elephants	**noun**	nose
15	time	elephants	name	September

Table 1. Icon dictionary of MinspeakWordsStrategy™ System

8.5. The Experimental System ILDE

The interactive Iconic Language Development Environment, ILDE,
has been implemented on the IBM PC using Visual C++ Version 1.0 under MS Windows.
It allows the designer to customize the knowledge base KB illustrated in Figure 2,
by defining the frames, the CD forms and the iconic operators.
The designer first enters an iconic sentence. The ILDE will display
the visual sentence using the iconic system SIL, and then use the
inference engine to generate the potential new meanings of the iconic sentence, as illustrated
by Figure 3. In Figure 3(a), the iconic sentence is "skull volt".
ILDE will first generate the basic frame as shown in the upper left window.
The potential new meanings are displayed in the (partially occluded) upper right window.
Selecting the new meaning "death IN THE sky" as the clue, the designer can now assign a new attribute value "crash" to the slot "RECALL",
and optionally assign a new Ω value 0.7 to this attribute.The result is illustrated in Figure 3(b), where the new attribute, its value,
the Ω value and the associated iconic operator are displayed in the lower left window.By clicking on the "Save Attributes" button, this revised frame is saved in the
frame dictionary as illustrated in Figure 3(c).
By applying the CD forms, 43 new meanings can be generated. Moreoever, by applying
the iconic operators exhaustively, 200 additional meanings can be generated.
These meanings are ordered by their Ω values, so that the designercan browse through them to select the appropriate ones as clues.

9. Discussions

In this paper, we presented the design process for iconic languages,
making particular reference to the visual languages of Minspeak™ systems. Our design methodology is based upon the theory of icon
algebra to combine icons into iconic sentences using iconic operators.
To represent the meaning of icons, we used a frame-based representation
based upon the semantic relationship diagrams

to represent the dynamics of icon semantics.
This augmented frame-based representation not only provides a natural
means for representing icon semantics, but also allows us to
develop a powerful conceptual similarity measure to facilitate
the association of a concept (or word) in the application domain with
an iconic sentence (or a set of iconic sentences with similar meanings).
Based upon this knowledge representation, an inference algorithm was developed to make
semantic inferences, where the semantics of
the iconic operators in the icon algebra were precisely defined.

The iconic languages for the Minspeak™ systems have been shown in practice to beeasy to learn by the users. The design process
described in this paper will further allow the designer to study the
ease or difficulty of learning other iconic languages by different groups of potential users in a systematic way.
The designer can first construct a basic iconic language and then add a new iconic sentence, and test
how easy or how difficult it is for the user to learn the new iconic sentence.
In other words, the design process supports incremental design as well as
incremental learning.

An interesting issue for further investigation, is to associate a syntactic structure
to the visual language generated.
In Section 8.3, we mentioned that a visual grammar G can be constructed as a by-product of the design process. In

a syntactic and semantic inference technique was
introduced which enables the users to generate their own visual languages.
Applying the program-by-example philosophy, the user is asked to
draw sample visual sentences for the intended language.
The inference algorithm then generates a visual language defined by a visual
grammar which includes and extends the given samples. The meanings of the newly
inferred visual sentences is derived from tables describing the entities and
the actions available within the specific application environment.

It will be interesting to investigate the refinement of G using such
grammatical inference techniques.
The tools implementing the these techniques
are already available on PC. We can apply grammatical inferences on
visual languages generated through the methodology described in this paper.
Having a grammar G to express the resulting iconic language will be useful
for several purposes. First of all, it can be

employed to predict the legality of a partial iconic sentence.
Given a visual language L, let G be the grammar derived byapplying grammatical inference on L. Usually L(G) contains L,with L(G)-L possibly empty. A partial iconic sentence u is s
coverage of additional concepts. A visual sentence in L(G)-L can be associated with additional concepts, using the semantic rules to obtain new
meanings.

Moreover, we may want to check whether two iconic languages are
equivalent. It will be interesting to find transformations between two
languages at the syntactical level rather than at the semantic level. The model
for knowledge representation proposed in this paper is one approach to attack this problem at
the semantic level.
The similarity function can play a major role in trying to capture similar
meanings from the object representations. But we still need to specify a
formal way to detect similarities in the dynamic aspects of the icons represented
through CD forms.
A methodology for constructing equivalent visual languages will
lead to better understanding of the advantages and limitations of
visual languages.

For general two-dimensional iconic languages,
we also need to study the semantics conveyed by the spatial arrangement
of icons, i.e., we must consider how to specify the semantics of the spatial relations,
and how to make inference (spatial reasoning) based upon these spatial
relations.

In conclusion, from the actual design experience of the Minspeak^{TM}systems, we have learned a lot about the design of iconic languages,
so that other iconic languages can be formulated and
evaluated in a systematic way.

Acknowledgement:
This research was supported in part by National Science Foundation under grant IRI-9002180, *Visual Reasoning for Information Retrieval*, and by a grant from the Semantic Compaction, I

Figures:

Figure 1. The Minspeak Iconic Keyboard click here.

Figure 2. An Interactive Design Environment.

Figure 3(a). Creating a new attribute based upon the clue "death IN THE sky" click here.

Figure 3(b). The new attribute is added to the frame for "skull volt" click here.

Figure 3(c). The updated frame dictionary click here.

Figure of COM operator click here.

Figure of MAR operator click here.

Figure of CON operator click here.

Figure of ENH operator click here.

Figure of INV operator click here.

Figure of IDX operator click here.