In *Proceedings of the Seventh International Conference*
on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems (IEA/AIE-94)
Langhorne, PA: Gordon and Breech Science Publishers, 1994.

THE FIGURE UNDERSTANDER: A SYSTEM FOR INTEGRATING TEXT AND DIAGRAM INPUT TO A KNOWLEDGE BASE

Raman Rajagopalan and Benjamin Kuipers
Department of Computer Sciences, University of Texas at Austin,
Austin, Texas, USA, 78712.
Email: raman@cs.utexas.edu, kuipers@cs.utexas.edu

ABSTRACT

Figures, consisting of a diagram and an accompanying textual description, are heavily used in human communication. Diagrams carry static spatial information about the scene while text is used to describe semantic and dynamic information. The saying 'a picture is worth a thousand words' exemplifies the great value of diagrams in describing a scenario. Since diagrams and text describe the same object in different ways, the primary task in automating the ability to use both text and diagram input in a modeling system is to find methods for coordinating the differing descriptions. This paper describes an implemented research tool, the Figure Understander, for accomplishing this task.

INTRODUCTION

Figures, consisting of a diagram and an accompanying text description, are heavily used in both formal and informal modes of human communication. Diagrams illustrate objects and their spatial attributes and relationships, and text often describes the semantics and dynamics of those objects. The unique role of each form of input, diagram and text, has already been analyzed in great detail by such authors as Larkin and Simon [LS87].

As shown by the recent symposium on diagrammatic representations [SS92], there has been much interest in utilizing both text and diagram input in AI applications. Novak and Bulko [NB90, NB93], for example, have demonstrated a methodology using a blackboard architecture for utilizing both text and diagrams to solve textbook physics problems. In their work, diagram input is treated as simulated visual input, consisting of interconnected lines, arc, and points. They demonstrated effective results on several non-trivial examples, but their approach requires addressing ambiguities in the interpretation of the diagram. They suffer from the classic visual scene analysis problem [BB82] of deciding which line segments belong together as parts of the same object, and which are only accidentally intersecting. PICTION [Sri91] is another example of a system for coordinating text and pictoral input given in visual form.

We describe an implemented research tool for integrating figure (diagram + text) input into a knowledge base that avoids many of the ambiguities of visual scene interpretation, and directly confronts the problem of multi-modal reference to the same conceptual object. Our program, the Figure Understander, takes as input a diagram in the form of a PostScript file produced by an object-oriented graphical editor (in the current implementation, `idraw`) and one or more sentences of descriptive text. Its output is a fragment of a knowledge base, which can be equivalently described as either a set of ground logical assertions or a set of frames, slots, and values. In the current implementation, the Algernon knowledge representation language [Cra90, CK92] is used to maintain the knowledge base. We are using the Figure Understander as the input processing module in a magnetic fields problem solving system [RK94].

Our approach is motivated by the observation that a figure is a communication act involving at least two agents [Gri75].

- The author of a figure has the goal of communicating a description of a situation to a human reader.
- The reader of the figure has the goal of extracting the intended knowledge from the diagram and the text description.

The author tries to make the reader's task as easy and unambiguous as possible. The reader knows this, and can make use of it, since the intended interpretation should have no close competing interpretations.

The Figure Understander has the same goal as the reader, and can take advantage of the same assumption of a cooperative author, but it has much less world knowledge than a human reader. Accordingly, we provide the following additional knowledge and assumptions:

- The graphical objects in the diagram are in one-to-one correspondence with the conceptual objects to be created in the knowledge base. Graphical features and spatial relations in the diagram, and information in the text, provide additional attributes of and relations among the conceptual objects, but no new objects.
- There is an order-of-magnitude difference between intended and unintended spatial relationships appearing in the diagram. Therefore, small deviations in the position and orientation of objects with respect to the coordinate system, or between objects, can be rounded to zero.
- The syntax of the text will be straight-forward to parse, and the meanings in the lexicon will have a straight-forward correspondence with the object types, attributes, and relations in the knowledge base.
- Referring phrases, including pronouns, will refer unambiguously to objects in the knowledge base at the time each phrase is processed, assuming that the diagram is processed before the text.

The way each of these is used by the Figure Understander will be discussed in detail in the sections below. We will now briefly describe the Algernon knowledge representation system.

ALGERNON OVERVIEW

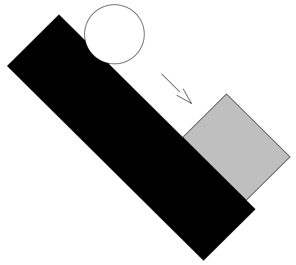
Algernon is a frame-based system that represents knowledge in terms of object classes and relations between them. It supports both forward and backward chaining rules for manipulating the contents of the knowledge base.

Object classes are described through a taxonomic hierarchy. The relation 'isa' is used to denote class membership. For example, the hierarchy given below indicates that the set 'things' contains two subclasses: people and pets. John is a member of the set 'people', and Fido is a member of the set 'pets'.

```
(:taxonomy (things (people john) (pets fido)))
```

Relations between classes are defined by the :slot form. For example we can define the relation 'pet-of' between the set of people and the set of pets as shown below:

- **Diagram:**



- **Text:** The loop is rolling down the ramp.

FIGURE 1: The diagram and accompanying text description provided as input for a magnetic fields problem.

```
(:slot pet-of (people pets))
```

We may then assert the predicate `(pet-of john fido)` to include the knowledge that Fido is John's pet. This predicate forms an `access path` into the knowledge base. An access path should lead to a unique object, or address, where a new piece of knowledge is to be added, or where information being queried may be found. The assertion `(pet-of john fido)` allows Algernon to first locate a frame for John, after which John's pet-of slot can be modified to contain Fido.

Our system provides a taxonomic hierarchy and a collection of relation definitions for capturing the spatial properties of the diagram. We expect the user to provide a domain-specific taxonomy and a collection of relations for capturing the semantic knowledge from the text and the diagram.

FIGURE UNDERSTANDER SYSTEM OVERVIEW

Figure 1 shows an input diagram and an associated text sentence. Both pieces of information are necessary to fully describe the scenario. The diagram shows the spatial relationships between the objects in the scene, and the text describes the dynamic changes that are taking place. The problem of interpreting the object references in the diagram and text is complicated by fact that the text refers to the objects as 'loop' and 'ramp' while the diagram contains only two polygons, a directed line, and a circular object.

We stated previously that Novak and Bulko [NB90, NB93] assume that pictoral input is provided visually and therefore have to face the problems of visual scene interpretation [BB82]. In our work, we address diagrams for describing scenarios where the exact visual features are unimportant, and abstractions may be used to approximate the exact spatial extents of objects. Example of such scenarios include textbook diagrams, maps of college campuses, and flow charts. We draw all closed objects (e.g., rectangles, general polygons, circles) as a single entity with the use of a graphical editor. Thus, each unique object is directly identifiable from the output of the graphical editor, and the scene interpretation problem can be avoided. The key issue then is to integrate information in the diagram and text.

The first step in integrating diagram and text input is to associate a semantics with the objects in the diagram. We accomplish this by associating the use of shading and patterns in the diagram with a semantic interpretation for the object. In Figure 1, the white shading is used to designate a conducting loop of wire without any initial current flow, the black shading is used to designate an immobile supporting object, and the dot pattern is used to indicate a magnetic field directed outward. The connection between the shading and the semantics is given through a `picture semantics`, as illustrated in Figure 2. We will provide a more detailed description of the picture semantics further below.

Then, when we process the diagram, we extract both spatial information, such as the connectivity, orientation, and relative position of each object, and can also retrieve semantic knowledge about each object from the shading and patterns used. E.g., after processing Figure 1, we learn both the spatial features of the circular object and also that it is a mobile, rigid, conducting loop of wire without any initial current flow.

However, even with the use of patterns, we may not be able to capture all the semantic information within the diagram itself. Text may also be required. In general, diagrams can be used to describe static states, but text is often used for describing any dynamic changes that are going on and for describing special domain specific properties, such as the names of objects.

For example, consider the directed line in Figure 1. In our application, a magnetic fields problem solver, a directed line is taken to be a vector, such as a force or velocity. However, in this example, the diagram does not specify the object the vector is acting on or whether the vector is a force or a velocity. We obtain this information by processing the text.

From the text input, "the loop is rolling down the ramp.", we make several domain dependent inferences. The word 'loop', in our application, means a mobile, conducting loop of wire. The word 'rolling' indicates that the loop is in motion. The word 'down' partially describes the direction of motion, and the word 'ramp' shows the presense of another object.

Given the specific domain knowledge that there will always be a vector in the diagram which precisely identifies the direction of motion, when interpreting the text that the 'loop' is 'rolling', we could trigger processes to find a vector and a loop in the diagram, and to assert that the vector is a velocity and that the direction of motion of the loop is the direction of the vector.

Summarizing, with the use of a domain specific picture semantics, we can interpret the graphical patterns associated with the objects in Figure 1 to conclude that the diagram contains a conducting loop, a magnetic field, and a supporting ramp. These facts can then be stored in the knowledge base. We also recognize that the directed line is a vector, but cannot determine which object it is acting on or whether the vector is a force or a velocity.

While interpreting the text for Figure 1, we trigger processes which query the knowledge base to find a conducting loop, a supporting ramp, and a vector in the diagram. We can then add the following facts: the vector is a velocity and the direction of velocity of the loop is the direction of the vector.

In the following sections, we go into further detail on the features of Figure Understander.

PICTURE SEMANTICS

One of the assumptions given in the introductory section for our Figure Understander system was that the graphical objects in the diagram are in one-to-one correspondence with the conceptual objects to be created in the knowledge base. In our system, we use such graphical features as patterns and shading to distinguish semantically distinct objects. We will refer to the process of attaching a semantics to particular graphical patterns as defining a `picture semantics`.

The picture semantics is defined via the DefPict form, as shown in Figure 2. Since spatial properties can change if motion is allowed, we specify whether a property is permanent or changeable by scoping each property to the scenario or initial model respectively. For example, since diameter is a non-changeable property, we can assert `(diameter-of loop 16 S)` to indicate that the diameter of the loop is always 16, where S is the user-specified name for the scenario given under `:scenario-name`.

The keyword, i.e, category, `:patterns-for-2d--objects` allows the user to specify labels for two-dimensional objects drawn using each of the pattern types allowed by IDRAW. We think of each label as defining a specific type of object. In Figure 2, we include the knowledge that a solid-white pattern represents a no-current-loop (a conducting loop with no initial current flow). Similarly, the keyword `:brush-types-for-vector-objects` may be used to specify internal labels for different types of vectors, based on line thickness, whether the lines are broken

```
(defpict pictures-for-mag-fields
    :scenario-name s
    :initial-model-name s-init-model

    :patterns-for-2d-objects
                    ; IDRAW pattern         ; USER Domain Object
                   ((light-rect-mesh        field-out-of-page)
                    (dark-rect-mesh         field-into-page)
                    (horizontal-line        horizontal-field)
                    (vertical-line          vertical-field)
                    (solid-black            other-object)
                    (solid-white            no-current-loop)
                    (backslash              clockwise-current-loop)
                    (slash                  counter-clockwise-current-loop))

    :brush-types-for-vector-objects
                    ; very-light, solid line pointing outward from
                    ; the origin (first vertex) of the vector.
                   ((vl-s-vector-out      mot-dir-vect))

                    ; Object Class  Abb. Name    Class Members
    :object-groups  ((conducting-loops   L   (no-current-loop
                                              clockwise-current-loop
                                              counter-clockwise-current-loop))
                    (magnetic-fields    F   (field-out-of-page field-into-page
                                              horizontal-field vertical-field))
                    (other-objects      O   (other-object))
                    (vectors            V   (mot-dir-vect)))

                            ; Class Name
    :object-group-semantics ((conducting-loops
                              ; Semantics Expressed as ALGERNON Predicates.
                             ((member conducting-loops ?obj)
                              (is-conducting-object ?obj TRUE)
                              (mobility-of ?obj mobile)
                              (is-solid ?obj TRUE)))
                            (vectors
                             ((member vectors ?obj))))

    ; format is (class-name (individual-object-type semantics-for-type)* )
    :individual-object-semantics
                           ((magnetic-fields
                             (field-out-of-page
                              ((field-direction-of ?obj '(0 0 1))))
                            ; Since the frame of reference can change, a
                            ; directional vector given in the input may need
                            ; to be modified to reflect this change.  This is
                            ; done via the function `get-transformed-dir'.
                             (horizontal-field
                               ((:bind ?dir (get-transformed-dir '(1 0 0)))
                                (field-direction-of ?obj ?dir)))))
```

FIGURE 2: Example of a partial picture semantics for our magnetic fields problem solving application.

or solid, and the placement of the head of the vector.

The keyword `:object-groups` is used to specify object classes from the individual types given in `:patterns-for-2d-objects`. In Figure 2, we have defined the class 'conducting-loops' and have established that a no-current-loop is a member of this set. In addition, we also specify the internal names to be used for objects of the class. In Figure 2, we have specified that all conducting loops should be named 'L'. Thus, the first loop identified in the diagram will be named 'L', the second 'L-1', the third 'L-2', and so on.

The object semantics are specified in two ways. Properties common to an object class as a whole are given through `:object-group-semantics`. The properties of individual object types within a class are given through `:individual-object-semantics`. The properties should be specified as a collection of Algernon predicates to be asserted once an object of the specified type has been identified. The variable ?obj may be used in in the predicates and will be bound to the particular object being examined.

In Figure 2, under `:object-group-semantics`, we have specified that all conduction loops are members of the Algernon set 'conducting-loops', that they are conducting objects, and, for the purposes of our application, that they are solid and mobile. Under `:individual-object-semantics`, we specify the field direction for two magnetic fields objects.

HUMAN ERRORS IN DIAGRAMS

Another assumption for Figure Understander was that there is an order-of-magnitude difference between intended and unintended spatial relationships appearing in the diagram. The differences between the diagram given as input to Figure Understander and the intended diagram may come about due to human error.

In our system, we input figures via the IDRAW graphical editor and extract the spatial data about the objects in the scene from the PostScript files generated by IDRAW. The spatial data is given in terms of coordinate values, from which such information as the slopes of lines can be extracted.

Since graphical editors store information in a very precise fashion, unintended distinctions may be present. Two edges intended to be parallel could have slightly different slopes. Two objects intented to be in contact could instead by slightly overlapped, or disjoint. We overcome these errors by using epsilon measures when comparing the position and orientation of objects both with respect to the global coordinate system and with respect to other objects.

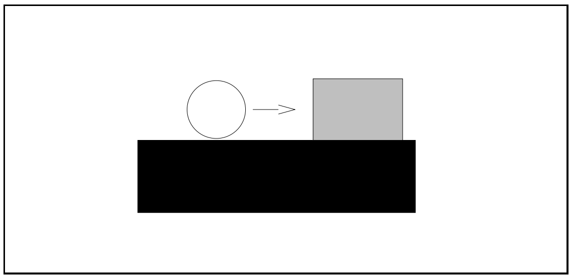


FIGURE 3: Diagram in Figure 1 redisplayed within a frame of reference chosen to maximize the number of edges aligned with the coordinate axes.

FRAMES OF REFERENCE

In our magnetic fields problem solving application, cross product operations on vectors are common occurrences, and we found that it is often useful to examine and reorient the frame of reference of the input diagram to simplify the vectors involved. The criterion used is to maximize the number of edges and vectors that are aligned with the coordinate axes. In the event of ties, priority is given to edges of immobile objects, since their orientation will not change. If a new frame of reference is selected, all coordinate values associated with the diagram are numerically manipulated to conform to the new frame of reference. In addition, a new PostScript file is generated for displaying the rotated diagram in IDRAW. The rotated diagram for our example is given in Figure 3.

This feature may be invoked at the user's option to rotate the initial diagram in any application. Note that if this feature is used, care must be taken to ensure that all spatial relations in any accompanying text are resolved properly. For example, for the diagram in Figure 1, assume that the accompanying text description of the scene contains the statement 'the loop is above and to the left of the field'. While this statement accurately describes the situation in Figure 1, the scenario in Figure 3 would no longer fit the statement.

SPATIAL PROPERTIES

Once the frame of reference has been selected, each individual item in the diagram is examined with respect to the semantic data given via the DefPict form, and an internal object is created in the knowledge base. We currently allow convex polygons, circles, and vectors in the diagram. Each internal object is named according to the user-specified naming abbreviations given in the picture semantics.

The user-provided semantic data is then attached to the internal object description. In addition, we determine such things as the connectivity of the diagram object, the orientation of the object with respect to the global coordinate system, and the relative position of the object with respect to other objects in the diagram [Raj93, RK94].

The connectivity of a polygon is stored by creating knowledge base structures for all the edges and vectors, and storing the connections between the edges and vertices. The orientation of an object is described in terms of its extremal points: the rightmost, leftmost, topmost, and bottommost points of the object. Then, for applications involving only static worlds or for those where only translational motion is possible, we approximate the spatial extents of objects by drawing a rectangular bounding box around the extremal points of the object.

This allows us to model the relative positions between diagram objects, such as whether the loop is to the *left* of the field, through inequality relations between their extremal points. In Figure 3, since the X-coordinate value of rightmost point of the loop is less than the leftmost point of the field, we may conclude that the loop is to the left of the field.

The inequality relations are stored in two lists, initial-ordinal-relations and permanent-ordinal-relations. The former describes properties that could change during the simulation while the latter describes permanent properties, such as the fact that the X coordinate value of the rightmost point of an object is always greater than that of the leftmost point of the same object.

TEXT PROCESSING

Even with the ability to specify semantic information about objects in the diagram through the use of patterns and shading, it may not be possible to capture all intended semantic information within the diagram itself. Text may also be required. In general, diagrams can be used to describe static states, but text may be used for describing any dynamic changes that are going on and for establishing certain connections between objects, such as associating names with objects. The present example (Figures 1 and 3) illustrates the use of text input to identify the dynamic processes acting on the loop (motion) and to identify the function of the arrow in the diagram.

The task of attaching semantic information given in an accompanying text sentence to diagram objects requires one to first locate the appropriate diagram objects to which the information is to be attached. We require that text must provide sufficient information to *uniquely* locate the diagram objects to which the information is to be attached.

Since the picture semantics can be used to provide the identity of the conceptual class for each diagram object, if we can obtain the conceptual class of each text object, we would have a means to match text objects against diagram objects. We use domain-dependent semantic interpretation functions, as illustrated in Figure 4, to provide the conceptual class for any nouns or pronouns that may be used in the text as reference objects. In Figure 4, the function 'make-variable' is used to return a variable name, such as ?V1 or ?V2, to be substituted into the `:the` from.

Algernon's definite description facility (`:the`) is used to connect object references in text to objects in the knowledge base (KB). The form `(:the ?v1 (member conducting-loops ?v1))` asks Algernon to find a *unique* object, to be bound to the variable ?v1, that is a member of the set conducting-loops. If it is the case that there is more than one conducting loop in the knowledge base, additional conditions may be added to the query to narrow the possibilities. Figure Understander provides several primitives to allow the use of spatial relations in the text to distinguish among otherwise identical diagram objects. The list of binary spatial relations recognized by Figure Understander is given in Figure 5.

The text processing is accomplished by first parsing the sentence through an ATN grammar, and then interpreting the parse output through a word interpreter, which, for each sentence type, looks for particular words at the position of the subject, verb, direct object, etc., and generates an appropriate Algernon form for those words. Since word interpretation is heavily domain dependent, we suggest that our methods should be used only as a guide by users in building word interpreters for their own applications.

As we process text, we generate an Algernon query-and-assert form to find the appropriate set of internal objects in the knowledge base and to attach any new information for those objects. Figure 6 shows the Algernon forms generated after processing the text in Figure 1. The first three lines are used to identify the loop, ramp, and vector respectively. These Algernon forms were derived partially by executing the semantic interpretation functions shown in Figure 4. The additional condition on the conducting loop, that it must be mobile, was added after recognizing that the verb in the sentence, rolling, is describing the motion of an object. The word 'rolling' also triggered the need to look for a vector and to retrieve its direction, and resulted in the check of the spatial relation that the loop and the ramp must be *touching* each other.

Once we have located the appropriate set of objects in the KB, we can attach new information obtained from the text to the object descriptions. In Figure 6, the final two lines assert that the vector is a velocity and that the direction of motion of the loop is that of the vector. Incidentally, note that the text in-

```
(defparameter *semantic-functions*
             '((loop (lambda ()
                       (let ((?v1 (make-variable)))
                            `(:the ,?v1 (member conducting-loops ,?v1)))))

              (ramp (lambda ()
                       (let ((?v1 (make-variable)))
                            `(:the ,?v1  (member support-objects ,?v1)))))    ))
```

FIGURE 4: Examples of semantic interpretation functions used to obtain the conceptual class for textual objects.

Right-of	Strictly-right-of	Intersecting-right	Touching-right	Adjacent-right
Left-of	Strictly-left-of	Intersecting-left	Touching-left	Adjacent-left
Above	Strictly-above	Intersecting-top	Touching-above	Adjacent-above
Below	Strictly-below	Intersecting-bottom	Touching-below	Adjacent-below
Ahead-of	Strictly-ahead-of	Intersecting	Touching	Adjacent
Behind	Strictly-behind	Overlapped	Enclosed-in	Encloses
Equals	Inside	Next-to	Connects	Along

FIGURE 5: Collection of binary spatial relations covered by Figure Understander.

```
Interpreting text sentence:
 (THE LOOP IS ROLLING DOWN THE RAMP)

 The following Algernon path shall be asserted:

((:THE ?V1 (MEMBER CONDUCTING-LOOPS ?V1)
           (MOBILITY-OF ?V1 MOBILE))
 (:THE ?V2 (MEMBER SUPPORT-OBJECTS ?V2)
           (TOUCHING ?V1 ?V2 S-INIT-MODEL))
 (:THE ?V3 (MEMBER VECTORS ?V3)
           (VECTOR-DIRECTION-OF ?V3 S ?VEL-VECT))

 (ISA ?V3 VELOCITIES)
 (VELOCITY-VECTOR ?V1 S-INIT-MODEL ?VEL-VECT))
```

FIGURE 6: The Algernon query-and-assert form generated for the text statement in Figure 1.

terpreter also uses the scenario name and the initial model name (s and s-init-model), as specified in the picture semantics given in Figure 2, to distinguish between 'permanent' properties and 'initial' properties.

AN ANNOTATED EXAMPLE

The previous sections have described the rationale and theory behind the implemented code. The following shows an annotated run of the example described by Figures 1 and 3. Recall that the text description for the diagram in Figure 1 is: "The loop is rolling down the ramp".

Figure Understander is invoked through the function 'parse-figure'. The arguments to **parse-figure** include the name of an **idraw** generated PostScript file containing the input diagram, a list of input sentences, and the name of the DefPict picture semantics form to be used. In addition, the user may instruct Figure Understander to examine, and if necessary, modify the frame of reference for the diagram by setting the keyword argument **:optimize-FR** to T (True). If the latter option is exercised, the user should provide the name of an output file for the final diagram through the keyword argument **:pict-out-file**. For the purposes of this example, we have not provided a text sentence as part of the input so that we may distinguish the output of the diagram interpreter from that of the text interpreter.

After the **parse-figure** form, we show the output of Figure Understander. Note that the first line in the output (after the banner) shows an 'automatic' correction of human error. Our system does not handle general ellipses; only circular objects. The loop in Figure 1 is not perfectly circular. Figure Understander has detected this human error, and forced the loop to be circular. The new diameter is computed via an averaging and rounding procedure.

```
> (parse-figure
    "ramp.ps"      ; Input diagram - PostScript.
    nil            ; Input text (nil for now).
    'pictures-for-mag-fields     ; DefPict form.
    :optimize-FR T
    :pict-out-file "figout.ps")
```

```
************************************************
***            Parsing The Figure           ***
************************************************

Coercing ellipse with length 17 and height 16
to a circle with diameter 16.

Choosing the new frame of reference.

The old coordinate system is being rotated by
45 degrees.

Determining new vertex coordinates.

Extracting spatial information from the diagram.

Creating a postscript file for the diagram after
the change in the frame of reference.
```

At this point, the processing of the spatial and semantic features of the diagram is complete, and information about the connectivity and extremal points of individual objects, and the inequality relations between the extremal points of different objects, has been extracted. Let us now examine some object descriptions. We begin with the description of the conducting loop. The internal Algernon name chosen for this object is 'L', as described in the picture semantics of Figure 2. For brevity, we have deleted the irrelevant information from the description of 'L'. Further, the output of Algernon's 'vf' command has been modified to enhance readability. The actual output encloses each value for a relation in a list with nil. For example, the value of the relation 'name' was actually given as ((l) nil).

```
> (algy)
Algernon 1.2.4 of 16 December 1992.

algy> vf L    ; view-frame
L:
  isa:        things objects
              convex-2-d-objects solid-objects
              mobile-objects circular-objects
  name:       (l)
  is-rectangular: false

  ; from picture semantics
  is-conducting-object: true
  mobility-of: mobile
  is-solid:   true

  ; diameter of the loop
  bsph-diameter: 16.0

  ; coordinates of extremal points
              ;X  Ymin Ymax
  rightmost:  (302 430 430 s-init-model)
  leftmost:   (286 430 430 s-init-model)
              ;Xmin Xmax Y
  topmost:    (294 294 438 s-init-model)
  bottommost: (294 294 422 s-init-model)
```

In the description above, one should note that the diameter of the loop 'L' and its extremal points, such as the topmost point, have all been identified. The semantic description given in Figure 2 for conducting loops, such as the observation that the loop is conducting, mobile, and rigid, has also been added. The knowledge that is missing is that the loop is in motion.

Now let us examine the description of the ramp, 'O'. Since the ramp is a polygon, information about its connectivity (vertices and edges) has been extracted. Note that the identities of the vertices and edges which form the extremal points are stored along with the actual coordinate values for the extremal points.

Here, the Y-coordinate value of the topmost point of the ramp, 'O', is 413. Since the Y-coordinate values of the bottommost point of the loop, 'L', is 422, strictly speaking, the two objects are not in contact. However, we treat all coordinate values within 10 pixels of each other as identical to counter human errors in drawing the diagram and rounding errors in processing the diagram. Although, with the correction for the human/processing errors, we can prove that the loop and the ramp are in contact, this fact has not been asserted directly into the description of the loop, L. We only infer spatial relations between objects on a demand-driven basis.

```
algy> vf O
O:
  isa:        things convex-2-d-objects
              immobile-objects solid-objects
              convex-polygons
  name:       (o)
  is-rectangular: true
  mobility-of: immobile
  is-conducting-object: false
  is-solid:   true

  edge-of:    o-e-1 o-e-2 o-e-3 o-e-4
  vertex-of:  o-v-1 o-v-2 o-v-3 o-v-4
  perimeter-of: (o-v-1 o-e-1 o-v-2 o-e-2
                 o-v-3 o-e-3 o-v-4 o-e-4)
  topmost-edge: (o-e-2 s-init-model)
  topmost-vertex: (o-v-2 s-init-model)
                  (o-v-3 s-init-model)
  topmost:    (251 403 413 s-init-model)
  bottommost: (251 403 373 s-init-model)
  leftmost:   (251 373 413 s-init-model)
  rightmost:  (403 373 413 s-init-model)
```

Now let us examine the description of the vector given in the diagram. Checking under the slot 'isa', we note that there is no evidence at this time that the vector is a velocity.

```
algy> vf V
V:
  isa:         things objects vectors
  name:        (v)
  is-rectangular: false
  vertex-of:  v-v-1 v-v-2
  transformed-figure-slope: 0
  vector-direction-of: (s-init-model (1 0 0))
```

Now we will enter the additional text input, and examine the descriptions of the two objects again.

```
algy> (interpret
       '(the loop is rolling down the ramp))

The ATN parse of the sentence:
 `` (THE LOOP IS ROLLING DOWN THE RAMP) '' is:

(S (NP (DET THE) (NOUN LOOP))
 (VP (AUX IS) (VERB ROLLING)
  (CLAUSE (AP (ADV DOWN)
            (NP (DET THE) (NOUN RAMP))))))

The text description accompanying the figure
has been processed.

 The following Algernon path shall be asserted:

 Algy path =

((:THE ?V1 (MEMBER CONDUCTING-LOOPS ?V1)
           (MOBILITY-OF ?V1 MOBILE))
 (:THE ?V2 (MEMBER SUPPORT-OBJECTS ?V2)
           (TOUCHING ?V1 ?V2 S-INIT-MODEL))
 (:THE ?V3 (MEMBER VECTORS ?V3)
           (VECTOR-DIRECTION-OF ?V3 S ?VEL-VECT))

 (ISA ?V3 VELOCITIES)
 (VELOCITY-VECTOR ?V1 S-INIT-MODEL ?VEL-VECT))
```

The output given above shows the ATN parse of the text input and the Algernon query-and-assert path generated to find the appropriate diagram objects and to assert any additional facts. In the Algernon path, note the predicates beginning with `:the`. These clauses attempt to identify a unique object description in the Algernon knowledge base that satisfies the specified properties. In our example, these clauses are used to identify the objects 'L', 'V', and 'O'. The query also requires proving that the loop is mobile and that the loop and the ramp are in contact (touching).

If the appropriate loop, ramp, and vector objects are uniquely identified, the additional facts given at the end of the path will be asserted. These facts add the knowledge that the loop is moving by attaching a velocity vector to the description of the loop, and conclude that the vector in the figure is a velocity.

Let us examine the description of the loop and the vector again. The new additions are markered by a '*'. For 'L', we find that the 'touching' relation with object 'O' has been proven, and in addition, the relation 'touching-above' has also been proven. This indicates that the loop is above the ramp. The rule for proving the relation 'touching' between two objects succeeds if there is surface contact at any boundary (top, bottom, left, right) of the enclosing bounding boxes. We also find that the direction of motion of the loop has also been attached to the description of the loop. Under the modified description of the vector, V, we find that it is now a 'velocities', and under the description of the ramp, O, we find that there is surface contact with the loop, L, and that the ramp is below the loop.

```
algy> vf L
L:
  isa:        things objects
              convex-2-d-objects solid-objects
              mobile-objects circular-objects
  name:       (l)
  is-rectangular: false
  is-conducting-object: true
  mobility-of: mobile
  is-solid:   true
  bsph-diameter: 16.0
  rightmost:  (302 430 430 s-init-model)
  leftmost:   (286 430 430 s-init-model)
  topmost:    (294 294 438 s-init-model)
  bottommost: (294 294 422 s-init-model)

* touching:   (o s-init-model)
* touching-above: (o s-init-model)
* velocity-vector: (s-init-model (1 0 0))

algy> vf O
O:
* touching-below: (l s-init-model)

algy> vf V
V:
*  isa: things objects vectors velocities
  name: (v)
  is-rectangular: false
  vertex-of:  v-v-1 v-v-2
  transformed-figure-slope: 0
  vector-direction-of: (s-init-model (1 0 0))
```

CONCLUSIONS

We have described a system, Figure Understander, for capturing knowledge from diagrams and any accompanying textual descriptions into a single knowledge base. The difficulty in performing this task arises from the fact that text and diagrams describe the same objects in different ways. Both forms of input may contain information not found in the other. Diagrams typically emphasize spatial properties while text deals primarily with semantic and dynamic properties.

Figure Understander solves the problem by providing a means for adding semantics to a diagram (picture semantics) so that both spatial and semantic data, including the identities of the conceptual classes to which the diagram objects belong, can be extracted and stored in a knowledge base. Then, by determining the conceptual classes of any reference objects in the text, we can match text objects against diagram objects. Once an appropriate set of knowledge base objects is identified, new facts from the text can be attached to the object descriptions.

In cases where there is only one diagram object of a conceptual type, as in the example presented in this paper, the use of the picture semantics description introduced in this paper makes the task of matching diagram and text objects straight-forward. If there are multiple objects of the same domain class in the diagram, the text must provide additional information to narrow the choices to the diagram objects of interest. Figure Understander includes a substantial spatial reasoning component and allows the use of spatial relations in the text to assist in isolating the relevant diagram objects.

Although the problem of correlating text and diagram input has been addressed at length in previous work, such as Novak and Bulko's [NB90, NB93] physics problem solving system, and Srihari's [Sri91] PICTION system, a different set of assumptions were utilized. They assumed that the pictoral input is provided visually, and had to address many of the problems associated with visual scene interpretation [BB82], a difficult problem in its own right. We have reduced the scope of the problem, and therefore have produced a simpler solution, by assuming that the diagrams are input via a graphical editor, where each unique object is drawn as a distinguishable entity.

Our methods may be used for the task of integrating diagrams and accompanying text input for any domain where the visual features of objects are unimportant, and spatial abstractions, such as rectangular bounding boxes or polygons, may be used to approximate the spatial extents of objects. We have applied the Figure Understander as the input processing module in a magnetic fields problem solving application and for the task of labeling the streets and buildings in a map understanding problem.

ACKNOWLEDGEMENTS

We thank James Lester, Richard Mallory, and the anonymous referees for providing helpful comments. This research was supported in part by NSF grants IRI-8904454, IRI-9017047, and IRI-9216584, and by NASA contracts NCC 2-760 and NAG 9-665.

REFERENCES

[AL90] Angehrn, A. and H. Luthi. Intelligent Decision Support Systems: A Visual Interactive Approach. *Interfaces* **20:**6, 17-28.

[BB82] Ballard, D.H. and C.M. Brown. *Computer Vision.* Prentice Hall, 1982.

[Cra90] Crawford, J. Access-Limited Logic - A language for knowledge representation. Ph.D. dissertation, TR AI-90-141, A.I. Lab, Dept. of Computer Sciences, Univ. of Texas at Austin, 1990.

[CK91] Crawford, J. and B. Kuipers. Algernon - A Tractable System for Knowledge-Representation. In working notes: AAAI Spring Symposium on Implemented Knowledge Representation and Reasoning Systems, Palo Alto, CA 1991.

[Gri75] Grice, H.P. Logic and Conversation. In P. Cole & J. L. Morgan (Eds.), *Syntax and Semantics, v. 3: Speech Acts.* New York: Academic Press, 1975.

[LS87] Larkin, J. and H. Simon. Why a Diagram is (Sometimes) Worth 10,000 Words. *Cognitive Science* **11**, 65-99.

[NB90] Novak, G. S., and W. Bulko. Understanding Natural Language with Diagrams. In: Proceedings of the Eighth National Conference on Artificial Intelligence (AAAI-90), 1990, pp. 465-470.

[NB93] Novak, G. S., and W. Bulko. Diagrams and Text as Computer Input. *Journal of Visual Languages and Computing* (1993) **4**, 161-175.

[Raj93] Rajagopalan, R. A Model of Spatial Position Based on Extremal Points. In Proceedings, ACM Workshop on Advances in Geographic Information Systems, Arlington, VA, November 1993.

[RK94] Rajagopalan R., and B. Kuipers. Qualitative Spatial Reasoning about Objects in Motion: Application to Physics Problem Solving. In: Proceedings of the Tenth IEEE Conference on Artificial Intelligence for Applications (CAIA-94), San Antonio, Texas, March 1994.

[Sri91] Srihari, R. PICTION: A System that Uses Captions to Label Human Faces in Newspaper Photographs. In: Proceedings of the Ninth National Conference on Artificial Intelligence (AAAI-91), 1991.

[SS92] Working Notes: AAAI Spring Symposium Series. Symposium: Reasoning with Diagrammatic Representations, Stanford University, 1992.