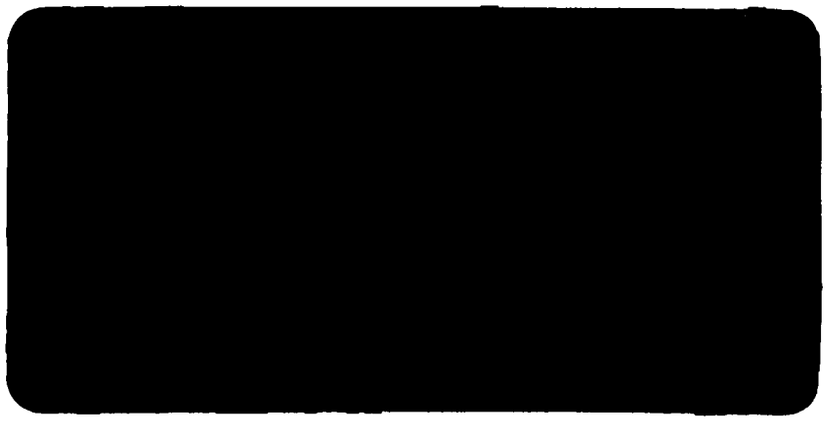
CONSTRAINED EXAMPLE GENERATION

Edwina L. Rissland

COINS Technical Report 81-24

Computer and Information Science

University of Massachusetts at Amherst



CONSTRAINED EXAMPLE GENERATION

Edwina L. Rissland

COINS Technical Report 81-24

Department of Computer and Information Science
University of Massachusetts, Amherst
Amherst, Massachusetts 01003

ABSTRACT

This paper addresses the problem of generating examples that have specified properties used to direct and constrain the generation process, which we call CONSTRAINED EXAMPLE GENERATION. We present and analyze several examples of CEG taken from protocols. Based upon such examples, we present a model of the CEG process. We then briefly describe a computer implementation of the CEG model.

This material is based upon work supported by the National Science Foundation under Grant No. IST-8017343.

CONSTRAINED EXAMPLE GENERATION

Edwina L. Rissland

Department of Computer and Information Science
University of Massachusetts
Amherst, MA 01003

Abstract

This paper addresses the problem of generating examples that have specified properties used to direct and constrain the generation process, which we call CONSTRAINED EXAMPLE GENERATION. We present and analyze several examples of CEG taken from protocols. Based upon such examples, we present a model of the CEG process. We then briefly describe a computer implementation of the CEG model.

This material is based upon work supported by the National Science Foundation under Grant No. IST-8017343.

1. INTRODUCTION

The ability to generate examples having specified properties is important in disciplines like mathematics, linguistics, philosophy, law and computer science [Baker 1978; Collins 1979; Levi 1949; Rissland 1978a, 1980a]. Examples lie at the heart of efforts that involve reasoning and learning, whether carried out by a person or a machine. They are useful for inductive reasoning, sharpening of conjectures, planning, programming and concept formation [Anderson 1982, Hawkins 1980; Hayes-Roth 1980; Lenat 1977; Polya 1968, 1973; Soloway 1978; Winston 1975]. Having a rich stock of examples is intimately related to understanding [Rissland 1978b].

In reasoning processes like proposing, proving and refuting conjectures [Lakatos 1976], there is counterpoint between developing and trying to prove conjectures and trying to refute or refine them. As Alpha of Lakatos' Proofs and Refutations says,

> "Discovery does not go up or down, but follows a zig-zag path: prodded by counterexamples, it moves from the naive conjecture to the premises and then turns back again to delete the naive conjecture and replace it by the theorem."

Examples are often the critical turning points in what Polya calls the "alternating procedure" [Polya 1965]:

> "...should we try to prove the assertion A or should we try to disprove it? We have here a choice between two different directions. To prove A we should look for some propositions from which, or for some strategy by which, we could derive A. To disprove A we should look for a counterexample. A good scheme is to work alternately, now in one direction, then in the other. When the hope to attain the end in one direction fades,

> or we get tired of working in that direction, we turn to the other direction, prepared to come back if need be, and so, by learning from our work in both directions, we may eventually succeed."

Examples play a central role in the evolution of theories; the examples, especially the paradigms, shared by members of a scientific discipline, show well what they believe their theory is [Kuhn 1970, 1974]. In disciplines like mathematics, the ability to generate examples can be considered as important as the ability to prove theorems; both processes are necessary links in the cycle of conjecturing-refining-proving/refuting by which such a subject grows. Examples thus can be thought of as having the same power and status as proofs, in the sense that one counter-example can disprove a conjecture just as a proof can establish it as a theorem.

Given the importance of examples an obvious question is "Where does an example come from?". In particular, how does one generate an example with desired properties? In this paper we look at the process of generating examples that satisfy certain constraints: Constrained Example Generation (CEG). Our approach was to: (1) make some initial guesses based on our own experiences, (2) take protocols of CEG, which were used to (3) couch a CEG model in computational terms, (4) implement the CEG model (in LISP), and (5) experiment with it. This paper presents our CEG model and some of the protocols upon which it is based; the CEG implementation is only described briefly; a more detailed discussion will be covered in another report.

2. OVERVIEW OF THE CEG PROCESS

Our model of CEG incorporates three phases: RETRIEVAL, MODIFICATION, and CONSTRUCTION.

When an example is sought, one can search through one's storehouse of examples for one that matches the properties desired. If one is found, the example generation problem has been solved through RETRIEVAL. In retrieval, there are many semantic and contextual factors -- like the last worked problems -- and therefore one is not merely plunging one's hand into an unorganized knowledge base. Thus even though retrieval sounds simple, it is no doubt very complex.

However, when a match is not found, how does one proceed? In many cases, one tries to MODIFY an existing example that is judged to be close to the desired example, or to have the potential for being modified to meet the constraints. Often the order of examples selected for modification is based on judgements of closeness between properties of known examples and the desiderata, that is, how "near" the examples are to what is sought.

If attempts at generation through modification fail, experienced example generators, like teachers or researchers, do not give up; rather they switch to another mode of example generation, which we call CONSTRUCTION. Under construction, we include processes such as combining two simple examples to form a more complex one and instantiation of general model examples or templates to create an

instance. Sometimes one constructs an example from a definition: one reads and picks it apart; then paraphrases and otherwise studies components of the definiton; finds examples for these more elementary constituent elements; and finally, combines them into an example. Sometimes this resembles back-chaining in theorem proving. Construction is usually more difficult than either retrieval or modification.

In examining human protocols, one sees two types of generation: (1) retrieval plus modification; and (2) construction. That is, one does not necessarily try first retrieval, then modification, then construction; sometimes construction is attempted straightaway.

3. EXAMPLES OF HUMAN CEG

We have taken protocols of 19 subjects working CEG problems from elementary function theory and geometry. The subjects were a mix of undergraduates (11), graduate students (5) and professors (3). Unless otherwise noted, any solution we present is typical of them all. We worked individually with each subject in a session in which we asked the subject to work on a sequence of CEG problems. Subjects were encouraged to think out loud as much as possible and to make as many notes and sketches as they wished; they could also spend as much time on a problem as they wanted. We tape recorded each session, which typically lasted 30 to 40 minutes,

and then used the recordings, some of which were transcribed, and the subjects' notes in our analyses.

To give the reader an idea of the richness and complexity of the CEG process, in this section we present synopses of some protocols for CEG tasks taken from the domain of plane geometry, concerning quadrilaterals. A complete transcript of one protocol for the second problem is included as Appendix A.

Since we were interested in spontaneous CEG behavior, we did not force subjects to draw or re-draw their solutions with accurate measures of angles and lengths, say with ruler and protractor. This, no doubt, would precipitate another type of CEG. However, to make it easier for the reader to follow, we include sketches and diagrams, often redrawn from what the subjects drew; for instance, if the subject marked three sides as being equal, we draw the figure that way.

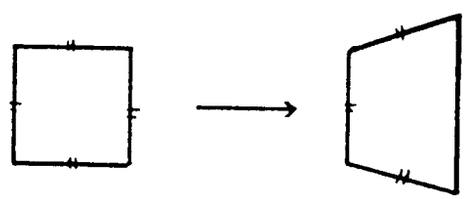
3.1 Three Sample CEG Problems with Solutions and Comments

Problem Geom1:

Draw a quadrilateral (4-sided planar figure) with exactly three equal sides.

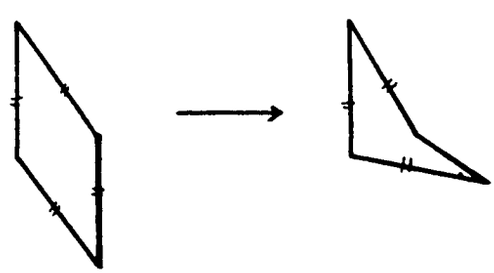
Solution 1:

Take a square, which has 4 equal sides, and modify it so that it has only 3 equal sides:



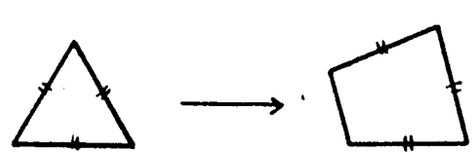
Solution 2:

Deform a rhombus, which has four equal sides, into a non-convex quadrilateral:



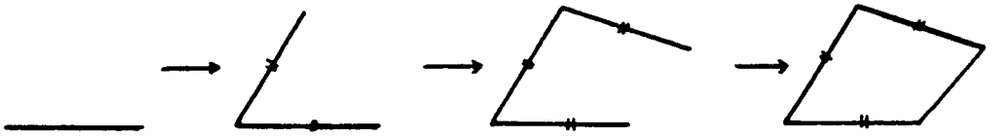
Solution 3:

Take an equilateral triangle, which has exactly three equal sides, and "open it up":



Solution 4:

Draw one side. Then draw two more sides in succession, each having the same length as the first. Then merely draw a line between the unattached endpoints of sides one and three, thus completing the figure:



Comments

Solutions 1, 2, and 3 follow a retrieval-then-modify progression. In Solution 1, the example retrieved, a square, is an important "reference" example, one of everybody's favorite quadrilaterals and one about which a great deal is known. {NOTE 1.} The rhombus of Solution 2, while also an example of a quadrilateral, is not as "standard" as the square. The equilateral triangle in Solution 3 is another well-known and important reference example in plane geometry, but not of a quadrilateral.

The square and rhombus are in some sense from the "right" class of objects, quadrilaterals, and meet the "having 4 sides" constraint. In contrast, the triangle is from the more general class of plane figures; it meets the other constraint, "exactly three equal sides".

The statement of problems and items of knowledge like results, concepts, and examples, often includes a pre-condition or hypothesis that in its treatment is different from the other conditions and hypotheses. For instance, the Bolzano-Weierstrass

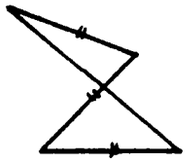
Theorem in real analysis states "In R**n (Euclidean n-space), every bounded, infinite sequence has a convergent subsequence". "In R**n" is a condition of the theorem that is treated differently from the rest of the theorem: it declares th context in which the theorem is to be set for proof, use or study [Rissland 1978a]. Similarly, in the geometric problems in this section, the requirement that the example be a quadrilateral sets the context: we are in the domain of quadrilaterals. Such a labelling of context highlights the difference between parsing the problem in these two ways:

1. "Find an X in QUADRILATERALS such that 1)..."
2. "Find an X such that 1) quadrilateral, 2)..."

In a strictly mathematical sense, there is no difference, although the emphasis is different. In an information retrieval sense, the first way specifies that we are to search through a knowledge base of quadrilaterals; the second says that quadrilateralness is simply one of the desired attributes. If the KB is organized into sub-KB's like one for quadrilaterals, the first way eliminates the need to test quadrilateralness once that KB is "brought in" and thus can be more efficient.

Solution 4 is constructed in a progressive "cascading" fashion in which another constraint is satisfied at each step. No problems were encountered in this construction -- for example, no intermediate results were nullified in later steps, as will be the case in Problem Geom3 below -- and there was no need for back-tracking, which would have made the problem-solving process much more complicated.

Solution 4 also involved fortuitous drawing of sides and angles so that the sides did not intersect; another choice could have lead to the following:

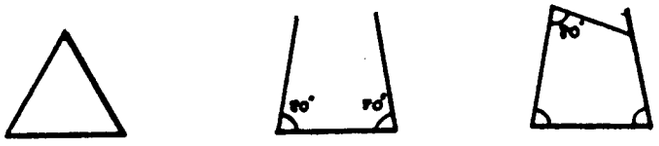


Problem Geom2:

Draw a quadrilateral with exactly three equal angles.

Solution 1: Retrieval+Modification

Draw an equilateral triangle and open up the two base angles, say to 80 degrees. Then draw a new side at 80 degrees to one of the opened sides and continue it over to the other opened side to complete the quadrilateral:

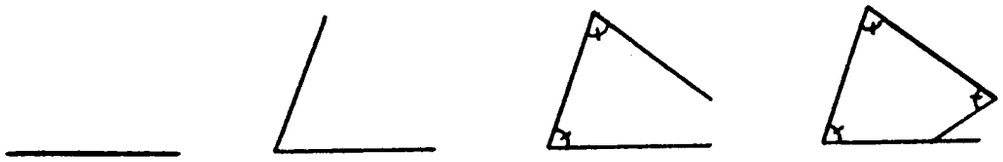


Solution 2: Cascading Construction

Draw a side. Draw another side meeting the first. Draw two more sides, attached to the endpoints of the first two sides, and making the same angles as between the first two sides. Let them intersect where they will:

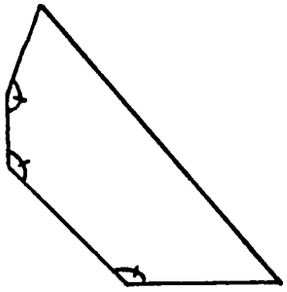
Solution 3: Another Cascading Construction

Draw a side. Draw another side meeting the first. Draw a third side, attached to the second and making the same angle as the first and second. Draw a fourth, attached to the third with the same angle. Continue it until it intersects the first side:



Buggy Solution 3:

The following 5-sided figure was produced with the cascading construction of Solution 3. In a sense, its "bug" is the result of a bad choice for the angle parameter in the construction.



Comments

One subject described the cascading construction of Solution 3 as "walking along a line and then making a turn" (See Appendix A.) much like the "TO-POLY" routine in LOGO Turtle Geometry [Abelson and DiSessa 1981]. It is interesting to wonder whether forcing subjects to use environments like LOGO, SmallTalk [Byte 1981] or even ruler-and-protractor would help.

There are often implicit criteria or standards of judgement in the generation and acceptance of solutions in these protocols. For instance, to be correct, Solution 1 would need a "trimming off" of the "unused" portion of the right side of the triangle. Both subject and interviewer "knew" to infer this. Thus, one can see a

tradeoff between judgement and modification -- simplicity in one often leads to compensating complication in the other.

Judgements of examples with respect to the constraints are an important part of CEG; without them incorrect examples can be offered as solutions, as with Buggy Solution 3 which subjects sometimes fail to note produces a 5-sided polygon. Thus, retrieval+modification is really a shorthand description for retrieval, judge, modify, judge.

Modifications, like "opening up" Solution 1, are procedures, which means they can be misapplied as one subject did when he used it to modify a quadrilateral into a five-sided figure, thus producing a buggy version of Solution 1. Constructions, also clearly procedural, can also go wrong, for instance through "bad" parameter choices as in Buggy Solution 3.

In applying procedures like the "opening up" modification of Solution 1 or the cascading constructions of Solutions 2 and 3, there is a problem of choosing the parameters with which to apply them; this is, in fact, another example generation problem: instantiation. Some subjects tried different angles and lengths in generating their solutions. However, no one chooses 90 degrees. Not choosing involves considerable knowledge. (For instance, the example of the opening up procedure applied with the argument of 90 degrees, its result (a rectangle), the theorem stating that the sum of the angles must be 360 degrees.) In procedural domains like computer science, an expert knows that certain parameter choices, like 0 in numerical and NIL in list

processing situations, can cause trouble; that is, that they are counter-examples. Thus, choosing involves knowing what not to choose.

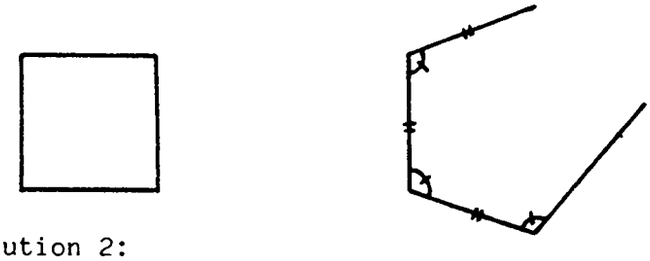
The following problem has no solution. However, attempted solutions involved CEG in interesting ways.

Problem Geom3

> Draw a quadrilateral with exactly three equal sides and three equal angles.

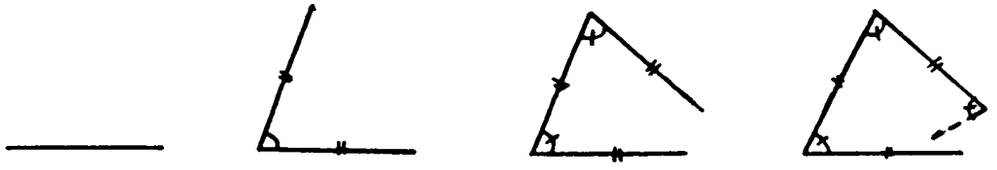
Attempted Solution 1:

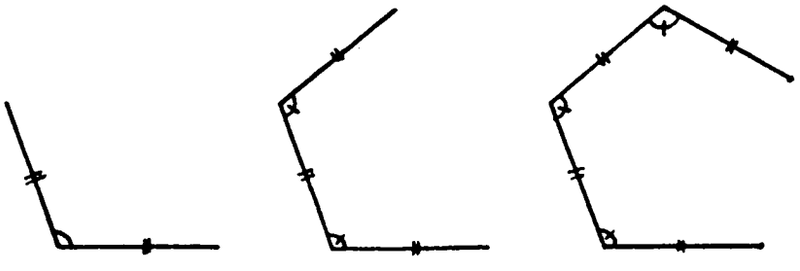
Take a square and unbend it so as to destroy the equality of one of the angles and one of the sides:



Attempted Solution 2:

Repeat the cascading Solution 3 to Problem 2 but "be careful" to keep the constructed sides equal:





Comments

Intuitively, most subjects felt the impossibility of this problem and made comments like: "there isn't enough room to move around"; "if you get three equal sides, trying to get only three equal angles messes up the inequality of the fourth side". They are commenting on a serious constraint interaction problem. Only one subject, a mathematician, sketched a proof to show the impossibility of the desired quadrilateral. {NOTE 2}

In some cases, the subject actually went through his "whole" data base of examples: square, rectangle, equilateral triangle, rhombus; in turn, he judged each a failure. Upon completing a retrieval phase, he then tried, unsuccessfully, to use modifications.

There is also something to be said here about the parsing and ordering of constraints and the agenda control that they engender. One can treat "three equal angles" as a single constraint or as a cluster of constraints, for instance as (1) Angle2 same as Angle1; (2) Angle3 same as Angle2. The same can be said for the "three equal sides" constraint. Attempted Solution 1 treats the constraints in the first way: that is, first handle the "three equal sides" constraint and then the "three equal angles" constraint; this could be called "block" control. Attempted Solution 2 treats the constraints in the second way; that is, alternate between meeting a side constraint and an angle constraint: do Side1, then Angle1, then Side2, then Angle2, etc.; this could be called "shuffle" control.

In this problem, one sees a conflict between "global", overall progress and "local" improvements. Considering the constraints parsed as (1) "three equal angles; (2) "three equal sides" (with the setting "quadrilaterals" as another constraint), one might quickly satisfy the "three equal angles" constraint. One could then attend to the "three equal sides" constraint and deform the figure to meet it; this looks like further progress since another constraint has been met, but satisfaction of the previous constraint has been undone. In other words, local, constraint-specific progress has been made at each step, and yet the overall satisfaction of the example still stands at only two of the three constraints; in other words there has been no change globally. Illusory local progress was also seen in Buggy Solution 3 to Problem Geom2 where subjects generated a figure with three equal angles but failed to notice that it was five- and not four-sided. {Note 3}

Conflict between local and global progress is indicative of constraint interaction problems. Such problems can be very difficult [Sussman 1975] [Stefik 1980]. However, recognition that one is encountering constraint interaction is useful in itself since it can alert one to switch problem-solving modes and perhaps invoke more sensitive judgement and control mechanisms and, at the very least, to give up before expending too many resources.

3.2 General Comments on CEG

In summary, some general points about CEG which these protocols illustrate are:

1. there are different generation strategies; for example, "retrieval+modification" and "cascading construction";

2. in addition to a collection of retrievable examples, there are also many known modification techniques (e.g., "opening up");

3. modifications are often applied to well-known "reference" examples; a modification episode can be described as a base example plus a sequence of modifications;

4. well-understood examples (e.g., models, references, counter-examples) focus the search for a new solution; examples capture the essence of past experience;

5. there are variations in what subjects consider the appropriate ground-level class of objects in which to search or attempt modifications; the specification of the class of the desired object (e.g., quadrilateral) could be considered another, but exalted, constraint much like the "setting" of theorems;

6. the importance of both local and global judgements of candidate examples' satisfaction of the constraints; laxity in judgement leads to accepting examples that are not solutions;

7. "good" constructions are not always good; they can go wrong through "bugs" or "bad" parameter choices;

8. there are several parsing and control strategies for the constraints themselves (e.g., the order in which they are dealt with);

9. modification often raises the problem of how to set the procedure parameters (e.g., what angle or length to use).

4. CEG: THE MODEL

In this section we present our model of the CEG process. Its high-level architecture is domain-independent and describes our experiences with CEG in a variety of domains including mathematics, computer science, law, lingistics and planning.

4.1 General Skeleton of the CEG Model

CEG has five basic subprocesses:

Retrieval, Modification, Construction, Judgement, Control

Presented with a task of generating an example that meets specified constraints, one:

1. SEARCHES for and (possibly) RETRIEVES examples JUDGED to satisfy the constraints from an EXAMPLES KNOWLEDGE BASE (EKB); or
2. MODIFIES existing examples JUDGED to be close to, or having the potential for, fulfilling the constraints with domain-specific MODIFICATION OPERATORS; or
3. CONSTRUCTS an example from domain-specific knowledge, such as definitions, general model examples, principles and more elementary examples from a DOMAIN KNOWLEDGE BASE (DKB).

Retrieval, Modification and Construction

are usually attempted in that order.

Clearly, this model needs many other features to describe the CEG process in its entirety: for instance, there are support processes for parsing of problem statements into constraints, displaying and representing information (e.g., images). In addition, understanding of a domain is an important adjunct process to CEG.

4.2 Details of CEG Model

There are many details, many of them domain-specific, needed to flesh out our model. In particular, some of the knowledge involved in the various subprocesses and knowledge bases is as follows:

1. EXAMPLES KNOWLEDGE BASE (EKB) - For each domain, there is an EKB. The EKB consists of examples and an organization of them. The examples have facets and attributes such as taxonomic classification (e.g., reference, model, counter), recency of creation, rating of importance. One important facet of an example is its pictorial representation, which can include a single or a sequence of images. Our subjects tended to share much of the EKB, and seemed to use similar description sets.

2. DOMAIN KNOWLEDGE BASE (DKB) - The domain knowledge can be very complex in the sense of the richness of its content and high degree of interconnectedness. The examples known (the EKB) are but one "sub-space" of the DKB; others include the formal results, definitions, heuristic principles, etc. [RISSLAND 1978]. For CEG, the most obviously relevant part of the DKB is the EKB. Content of the other parts (e.g., procedural knowledge about concepts) is distributed in other parts of the CEG model.

3. RETRIEVAL - Retrieval involves knowledge of ways to access and search information in the EKB. Some of this is domain-independent like knowing how to search a knowledge base or do associative retrieval on a taxonomic classification (e.g., "favorite reference example"), or that a good heuristic is to retrieve standard reference examples, known counter-examples, and recent successes (i.e., examples that have solved CEG problems) before other types of examples.

Some is domain-specific like retrieving on domain-specific attributes like "quadrilateralness".

4. MODIFICATION - The primary domain-specific aspect of modification is knowledge of domain-specific modification operators. This includes knowledge of which operator to use, and when, and how, in order to achieve (o improve) an example's satisfaction of a given attribute. Modification also involves domain-specific strategies like "modify angles before sides" and domain-independent stategies like "keep going if success is being had" and "switch modification techniques for variety or if things are not going well".

5. CONSTRUCTION - Domain-specific aspects of construction include combining and instantiation techniques and knowledge of how to build "from scratch", as in a cascading construction, or how to combine two examples, as with mathematical "curly" brackets used to define a new function from others. A less domain-dependent aspect is the ability to call CEG recursively, for instance for "sub-examples" to combine; however, domain-dependent knowledge would be needed to set up such calls.

6. JUDGEMENT - Judgement involves knowing how to evaluate a domain-specific attribute of an example, e.g., 4-sided-ness. Judgement is not just a binary yes/no evaluation of an attribute, but includes judgements like "with respect to 4-sided-ness, this example has 3 sides and thus lacks one" or "with respect to angles-between-sides, this example has small angles". Judgement also includes the ability to do comparative judgement such as "this example is better than the last one" or "example-1 has wider angles than example-2".

7. CONTROL ELEMENTS - The Control in the CEG model is distributed throughout the model. For the sake of discussion, the control issues (e.g., what to do next) are culled together here. Typical control questions and possible answers are:

    1. What retrieval policy should be used: e.g., most recent success, most important, on the basis of taxonomic class (references before...);

    2. What modification policy should be used: e.g., if some improvement has been made by the last modification, try again; try 4 times with a given modifier before giving up; pick the modifier with the most successes to its credit as the first modifier to try; vary which modifier to use;

    3. How should retrieval and modification interact: Should retrieval be completed on the whole EKB before modification is begun; or should a single example be retrieved and immediately be subject to modification (if it does not meet the constraints).

4. When should construction be begun: If a general model is retrieved, instantiate the model and then pass control and the instantiated example back to the calling process; if modification fails or is not going well try to construct an example from more elementary examples.

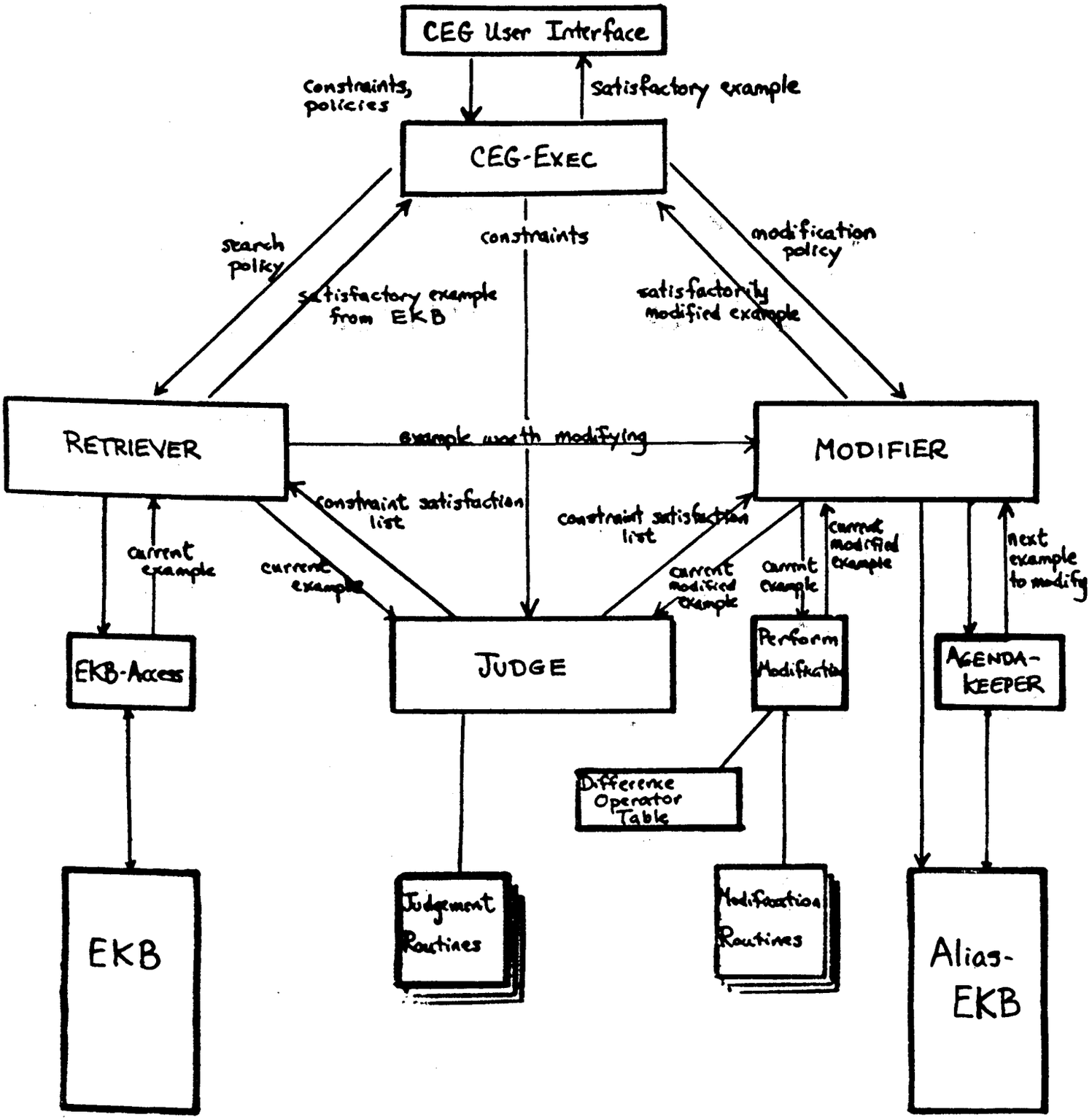
5. How should progress be measured: through devices that measure an example's satisfaction of individual constraints.

5. CEG: THE MODEL IMPLEMENTED

To give the reader a better idea of how the CEG model works, in this section we describe some of the details needed to implement the model of the previous section. This discussion of our implementation is intended to be an elaboration of the ideas in the previous sections and not a detailed discussion of the implementation itself.

5.1 Domain-Independent Shell

We have implemented our model of CEG in a computer program, written in LISP; it runs on a VAX 11/780. In our implementation we have concentrated on retrieval+modification. The modules and knowledge sources for the system -- without CONSTRUCTOR -- are shown in Figure 5.1.


CEG User Interface
constraints, policies
satisfactory example
CEG-EXEC
search policy
constraints
modification policy
satisfactory example from EKB
satisfactorily modified example
RETRIEVER
example worth modifying
MODIFIER
constraint satisfaction list
constraint satisfaction list
current example
current example
current modified example
current example
current modified example
next example to modify
EKB-Access
JUDGE
Perform Modification
AGENDA-KEEPER
Difference Operator Table
EKB
Judgement Routines
Modification Routines
Alias-EKB

Briefly, when the CEG program is presented with a list of constraints, in the system's syntax, (1) the EXECUTIVE initializes the system and then coordinates the other modules; (2) the RETRIEVER searches for and retrieves examples from an Examples Knowledge Base (EKB); (3) the MODIFIER applies modification techniques to an example; (4) the CONSTRUCTOR instantiates general model examples; (5) the JUDGE determines how well an example satisfies the problem constraints; (6) the AGENDA-KEEPER maintains an agenda of examples to be modified.

In our work, we represent an example by a frame-like data structure [Minsky 1975] in which each slot contains information about a different facet of an example, like its name, taxonomic class, rating of its importance, a caption describing what it shows, a picture illustrating it, procedures to construct it. The collection of example frames is organized by the "constructed from" relation in which Example1--->Example2 means that Example2 is constructed from Example1. The set of examples organized in this way form what we call an "Examples-space" [Rissland 1978a,b].

In describing the knowledge that one has about examples and uses in CEG, one needs a many-faceted frame rich with declarative, procedural, pictorial and relational information. However, in our implementation we restrict the representation frame to the few facets specifically needed to perform the retrieval, modification, judgement and construction operations that we are investigating.

In our implementations of the CEG model, we have concentrated on retrieval+modification and construction by instantiation. Thus, we have not explored construction by combining examples and the recursive calls to CEG that that would entail. A truly full implementation of CEG would need a sophisticated construction mechanism, including perhaps a theorem proving capability.

5.2 Domain-Dependent Modules

The CEG system shell is domain-independent with a clean and consistent interface to domain-specific knowledge. Only the MODIFIER, CONSTRUCTOR and JUDGE are domain-dependent; the MODIFIER and JUDGE are really the workhorses of our system.

For the CEG program to work in a given domain, it must be supplied with the following domain-specific knowledge:

1. The EKB
2. modification routines used by the MODIFIER
3. judgement routines used by the JUDGE
4. instantiation routines used by the CONSTRUCTOR
5. policies for retrieval and modification

Some details about the modules are:

1. EXAMPLES KNOWLEDGE BASE: The components use a common knowledge base consisting of two parts: a "permanent" EKB, containing known examples represented as frames and organized with "constructed from" links to form an Examples-space, and a temporary copy of the permanent EKB, an "alias" EKB, containing information gathered in the solution of a specific CEG problem.

   We refer to the temporary knowledge base as an "alias" EKB since it is a copy of the permanent EKB embellished with evaluation information generated during the working of an individual problem, such as the "constraint satisfaction list"

generated by the JUDGE to show an example's satisfaction of problem constraints, and "grown" to include new examples created during the working of a problem, for instance created by modifications of known examples (from the EKB). Upon successful completion of a problem, the newly generated solution example can be added to the permanent EKB -- its frame plus its constructed from links (i.e., modified or instantiated from links) -- before the other temporary information in the alias EKB is deleted in the clean up process, which is a function of the EXECUTIVE.

2. The EXECUTIVE module is responsible for overall control. The EXECUTIVE is responsible for initializing at the outset and cleaning up at the conclusion of a CEG problem. The EXECUTIVE is responsible for passing retrieval and modification policies to the RETRIEVER and MODIFIER.

3. The RETRIEVER is domain-independent. It retrieves examples according to a "retrieval policy" passed to it by the EXECUTIVE. The policies use the structure of the EKB and attributes such as epistemological class, worth-rating, recency of success to define a prioritization of examples to be retrieved and evaluated by the JUDGE. Typical retrieval policies would be "search the EKB in a breadth-first manner and retrieve model examples before reference examples before other types" or "retrieve most recently successful examples first". The RETRIEVER can retrieve on domain-specific attributes (e.g., quadrilateralness) if the examples in the EKB have the corresponding tags.

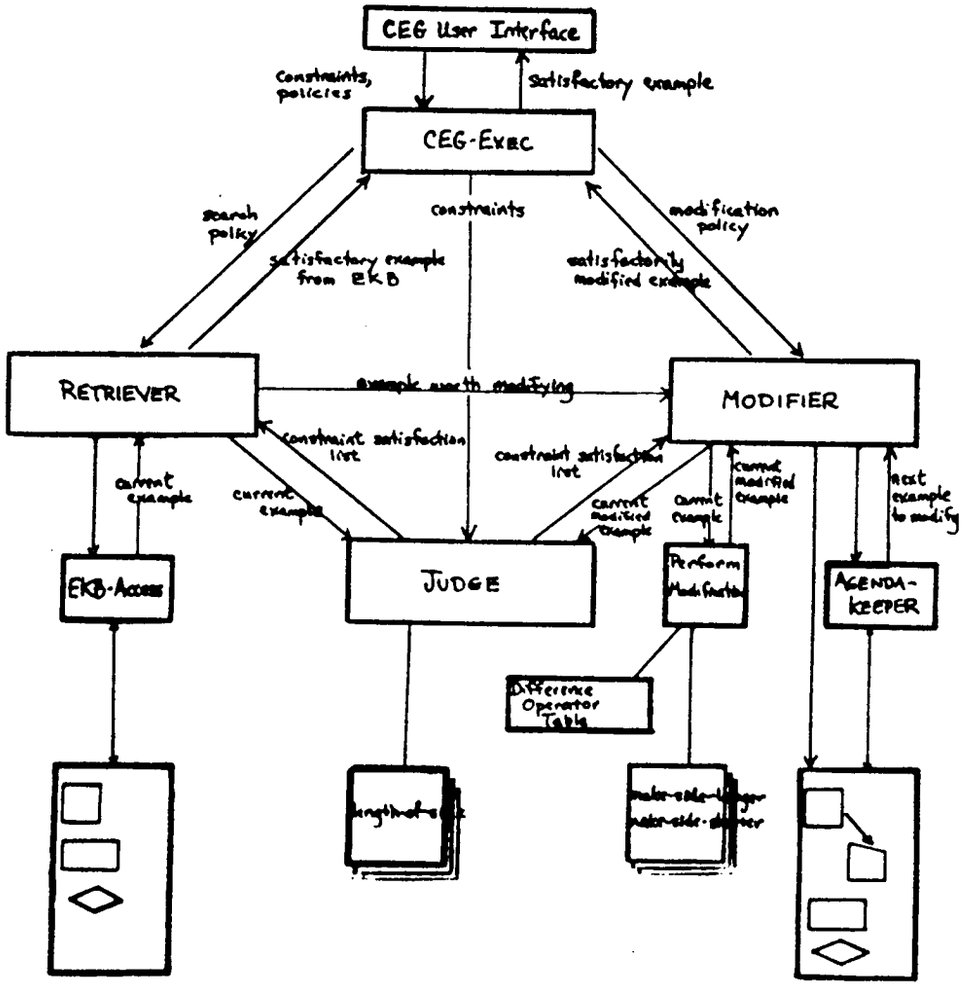
4. The MODIFIER contains domain-specific knowledge of (1) the modification procedures; and (2) when to apply them. The MODIFIER receives its "modification policy" from the EXECUTIVE. The MODIFIER is the executive process for the modification phase; it coordinates: calls to the JUDGE for evaluation, performance of modifications, storage of examples in the Alias EKB and the selection of the next example for modification. The modification policy is used by the subsidiary AGENDA-KEEPER. The modifications are performed by a sub-module that uses a difference operator table as in GPS [Newell and Simon 1963].

5. The JUDGE evaluates an example with respect to the posted domain-specific constraints and passes the results of its evaluation to the RETRIEVER and MODIFIER in a "constraint satisfaction list". This includes information on the differences between the actual and desired values which is used in conjunction with the difference operator table.

6. The AGENDA-KEEPER uses the modification policy to maintain an agenda of examples to be modified. An example of a modification policy is "if there is improvement with a modification operator apply it again but not more than four times in a row".

7. The CONSTRUCTOR in our implementations is the module with the weakest analogy to the CEG model of the previous section; it is capable only of instantiation of model examples. Thus, the CONSTRUCTOR in our program possesses the domain-specific knowledge to instantiate model examples (e.g., code "templates" [Soloway and Woolf 1980]) from the EKB.

The CEG System for Problem Geom1



This schematic shows the CEG system endowed with knowledge to work problems involving constraints on the lengths of sides of polygons in a retrieval+modification manner:

1. an EKB of polygons (e.g., the unit-square which is a reference example, a non-square rectangle, a rhombus);
2. judgement routines for evaluating the length of a side of a polygon;
3. modification routines for changing the length of a side;

Embedded in the code for these routines would be additional domain-specific knowledge about the scaling of vectors, counting the number of sides of polygons, etc.

6. FURTHER DISCUSSION OF HUMAN CEG

There are many aspects of CEG that our model and its implementation do not address, like esthetics. In this section we present additional fragments and synopses of protocols to illustrate more of the complexity and richness of human CEG. Our problems are taken from elementary function theory of the kind that a first year calculus student can solve.

Some of the aspects of CEG to be illustrated are:

1. Pure Retrieval
2. Retrieval versus Construction
3. Esthetics in Judging
4. Divide-and-Conquer plus Merging
5. Variety in Representation
6. Over- and Under-constraining
7. Default assumptions

In the following problems, all functions are real-valued functions of one real variable; in particular, their domain is the real numbers. Thus, the setting for these problems is "R", the reals, or FCN[R,R], the functions from R to R.

6.1 Pure Retrieval

Although most of our discussion has concentrated on generation by retrieval+modification and construction, some problems can be solved by pure retrieval; for instance:

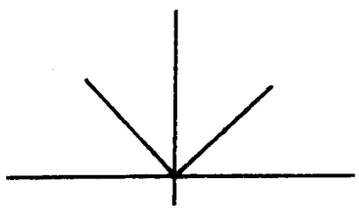
Problem Fcn1:

Give an example of a function that is continuous at a point but not differentiable there.

Solution 1:

$$Y = |X|$$

or its graphical representation:



Comments

Most subjects who have had some calculus handle this problem with Solution 1, the reference example of the absolute value function (at the origin). Responses (in a post-calculus population) are usually immediate, indicating that the retrieval was made very readily. For a subject who does not know the calculus well, this example might involve much more work to generate since it might not yet have become known, or be known as a reference example.

The two principal representations used for Solution 1 are algebraic and graphical. In other problems and solutions, one also finds "kinesthetic" representations, in which the subjects use their hands to trace out a curve or employ language with words like "climbing" or "turning". (Such language also entered into the geometric problems, e.g., "walking" in Problem Geom2.) Each representation has its own advantages and drawbacks; for instance, pictures are easy to generate, but are hard to use in some situations like differentiation or integration; formulas are

easy to use in such situations but are difficult to generate in the first place. In fact, it takes considerable skill to translate from one representation like pictures to another like equations [Clement et al 1981]. (Such a translation difficulty will be seen in the third problem, Fcn3.) One surprise in these protocols is the subjects' lack of "binding" to any one representation mode in individual problems, let alone across problems; subjects are opportunistic and choose whichever seems easiest.

6.2 Retrieval versus Construction, Esthetics

The next problem was solved by retrieval and by construction. Even though the retrieved examples were clearly well-known to all subjects, some subjects chose to construct a new example.

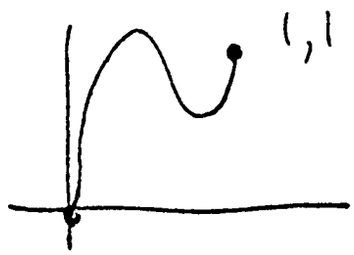
Problem Fcn2:

> Give an example of a function going from the point (0,0) to the point (1,1).

Solution 1:

Y = X was offered most of the time; sometimes it was represented as an equation, othertimes as a straight line or graph on (sketchy) Cartesian coordinates. Y = X**2 was also offered frequently.

Solution 2:



Comments

The functions Y=X and Y=X**2 are clearly products of retrieval; they are "favorite" examples. In fact, Y=X is perhaps the simplest example of a function. It is an example of many mathematical ideas (e.g., monotonicity, one-to-one, onto, identity, isomorphism). This diversity can be considered both a strong and a weak point; subjects constructing solutions like Solution 2 stated Y=X wasn't a "good" example because it showed too many things; some said they felt it was too simple and that something more complicated was more esthetically pleasing. This is related to their feeling that in these problems "an example" means "any arbitrary example". Thus an example should show generality by exhibiting many features, but it should not exhibit so many as to seem too special; this is the old "type-token" problem [<ref>]. This is also an interesting contrast to the usual view that "simpler is better" as embodied, for instance, in Occam's Razor [Russell 1945].

6.3 Divide and Conquer plus Merging

One can frequently solve problems by dividing them into subproblems, "conquering" each of them, and merging the subsolutions. The merging often involves "global" knowledge of the problem as a whole, which in CEG can involve interactions among the orginal constraints and imposition of additional constraints on how the subsolutions match up. In these simple function problems, such "matching up" constraints involve boundary

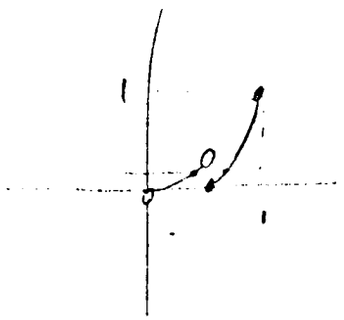
conditions on the functions. The point is that to do a successful "divide and conquer plus merging" solution involves additional calls to CEG, grouping of constraints, and imposition of additional constraints.

Problem Fcn3:

> Give an example of a function on the unit interval that goes from the point (0,0) to the point (1,1) and that is not one-to-one. {NOTE 4.}

Solution 1:

Use Y=X**2; this takes care of the (0, 0) requirement. "Cut and paste" it to take care of the not 1-1 requirement. Then stretch the right hand side to pass through (1,1). This looks like:



This was then translated (incorrectly) into the algebraic representation:

$$\begin{cases} Y = X^{**}2 \text{ when } X \leq .5 \\ Y = X^{**}2-1/4 \text{ when } X \geq .5 \end{cases}$$

Comments

In this solution, the sub-solutions are merged back together through the use of a picture or mathematical "curly" brackets. It is interesting to note that although the curly bracket construction is one of the best known and most successful techniques by which to construct new functions, it is hardly ever taught as such. Even in a standard calculus book [Thomas and Finney 1979] where there is a lengthy introductory discussion to

make explicit certain basic notions like 'function', there is no discussion of the bracket idea even though it is used. (Students rarely use it until they are told it is acceptable.)

The subject offering Solution 1 didn't quite finish things up in his algebraic expression of his answer: the third constraint of passing through (1,1) is not met. However, this can be handled by "fudging" the coefficients (perhaps by setting up and solving simultaneous equations) {NOTE 5.}.

Using the mathematical curly brackets often introduces bugs at the "join" point as in the following solution which doesn't account for x = .5:

Solution 2:

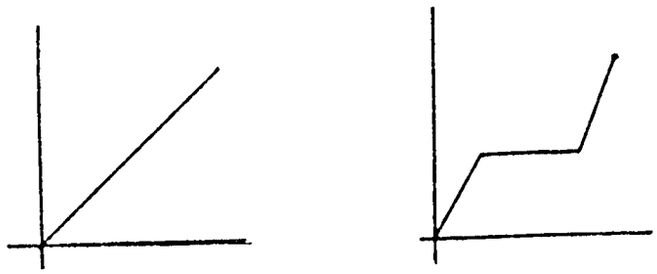
$$f(x) = \begin{cases} 0 & x < .5 \\ 1 & x > .5 \end{cases}$$

Note that this solution is made up by combing two simpler examples: the 0 and 1 constant functions. Using 0's and 1's, as for instantiation of a parameter, is a very important mathematical strategy, the 0-1 "mega-principle" [Rissland 1978a,b].

The following solution of the retrieval+modification variety is interesting in that it is similar to the function used as the first step in the sequence of functions in the construction of the Cantor function, a very important function in real analysis [Gelbaum and Olmstead 1964].

Solution 3:

Take a function such as Y=X on the interval [0,1], and "make a flat spot":



Solution 4:

The subject who offered Solution 2 to problem Fcn2, contentedly said "see the above".

Clearly, this is an example of retrieval of a recent "success" from an updated EKB.

6.4 Overconstraining, Underconstraining and Defaults

The following problem shows two different approaches to dealing with constraints: (1) over-constraining, then relaxing; (2) under-constraining, then tightening up.

Problem Fcn4

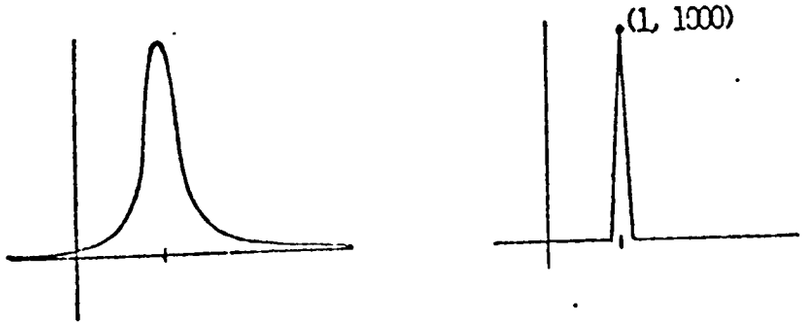
> Give an example of a non-negative, continuous function defined on the real line with the value 1000 at 1, and with area under its curve less than 1/1000.

Solution 1:

Start with the function for a "normal distribution". Move it to the right so that it is centered over X=1. Now make it "skinny" by squeezing in the sides and stretching the top so that it hits the point (1,1000).

"I can make the area as small as I please by squeezing in the sides and feathering off the sides. But to actually demonstrate that the area is indeed less than 1/1000, I'll have to do an integration, which is going to be a bother.

"Hmmm. My candidate function is smoother than it need be: the problem asked only for continuity and not differentiability. So let me relax my example to be a "hat" function because I know how to find the areas of triangles. That is, make my function be a function with apex at (1,1000) and with steeply sloping sides down to the x-axis a little bit on either side of X=1, and 0 outside to the right and left. (This is OK, because you only asked for non-negative.) Again by squeezing, I can make the area under the function (i.e., the triangle's area) be as small as I please, and I'm done."



Solution 2:

Another subject said, "Take the zero function, which clearly satisfies the less than 1/1000 constraint and is non-negative and continuous, and the function which is 1000 at X=1 and 0 elsewhere (a "characteristic" function) which clearly meets the (1,1000) constraint, and smooth them together."

Comments

Solution 1 starts off with an implicit over-constraining: the example retrieved, a normal distribution, is a reference example from the class of infinitely differentiable functions which are not only continuous but also have continuous derivatives, that is, they are exceedingly "smooth", in fact much "smoother" than mere continuity requires. Because of the difficulties in demonstrating satisfaction of the area constraint, the subject subsequently relaxed his function to a hat function, which is continuous but whose derivatives are not; the hat function has no gaps or breaks but does have "corners".

Solution 2 starts off with an explicit underconstraining. A "smoothing" operator is then used to tighten up his solution. The "smoothing" operation is very powerful indeed; it involves knowledge on how to make functions continuous, a very non-trivial matter.

It was observed in all the protocols that subjects make implicit default assumptions about the symmetry of the function (i.e., about the line X=1) and its maximum (i.e., occurring at X=1 and being equal to 1000). There is no specification about either of these properties in the problem statement. It is as if one's "frame" for a function includes symmetry as a default. This may be related to effort: it is less work to use a copy of one side for the other than to specify the two sides separately or distinctly.

The symmetry default was also seen in the quadrilateral problems of Section 3. For instance, in problem Geom1 there is no reason to "move" two sides symmetrically as in Solution 1.

There were also instances of over-constraining and relaxing: trying to use equal, or nearly equal, sides in solutions to problems requiring equal angles and vice versa (Problems Geom1 and Geom2). The non-solvability of Problem Geom3 shows that such over-constraints must necessarily be relaxed. Attempted Solution 2 to Problem Geom3 can be viewed as a tightening up ("be careful") of the cascading solution to Problem Geom2.

8. CONCLUSIONS

In this paper, we have described the process of constrained example generation and illustrated it with protocols from mathematics. We have described a model for CEG in terms of processes of retrieval, modification, construction, judgement and control and a knowledge base of domain-specific knowledge, particularly a knowledge base of examples. We then presented some of the details necessary to build a computer implementation of our model.

In our investigations of examples in other fields such as law and linguistics, we have found our CEG model to provide a good description of the CEG problem-solving behavior we have observed. We have also found our implemented CEG system to be robust and easily adaptable to different domains of knowledge.

A related, and very important, problem to CEG is the constraint generation problem. That is, how does one decide what one wants in an example, i.e., in our terminology, what are the constraints? To answer this, one must examine the context in which the example is to be used and the purpose for which it is intended. In mathematics for instance, this involves looking closely at the "alternation process" and the conjecturing-proving-refuting cycle. Sometimes the constraints on the examples come from the proof of the theorem that the example is to illustrate; sometimes from the proof sketch of a conjecture that one is trying to refute. In computer science, the constraints can come from sources as diverse

as the intentions of the program user, the specifications, or the code itself. In law, the kind of example (e.g., hypothetical case) used depends on the doctrine or argument to be advanced.

We feel that the CEG model of example generation has captured many of the important features of example generation. Our current efforts are now focussed on the CEG in different domains like law -- the problem of generating hypotheticals -- and the larger issues of constraint generation, especially in the context of giving explanations. In conclusion, we feel that examples are important to many fields of endeavor and that CEG is but one example of the rich role they play in problem solving.

Acknowledgements

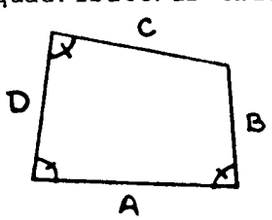
The author gratefuly acknowledges the significant contributions of E. M. Soloway. Thanks also to S. Waisbrot who helped in gathering protocols, R. S. Wall who implemented the second version of the CEG system, and to O. G. Selfridge who has provided valuable comments on CEG in general.

NOTE 1. Examples can be classified according to their role in the development and understanding of a theory. We use the taxonomy previously developed [Rissland 1978a, b, 1980]:

1. start-up examples are perspicuous, simple cases;
2. reference examples are standard, specific, ubiquitous cases;
3. model examples are general, paradigmatic cases;
4. counter examples are limiting, falsifying cases;
5. anomalous examples are exceptional, pathological cases.

Such a taxonomy is neither exhaustive nor exclusive. It is also not stationary in the sense that the members of the classes change as the development or understanding of the theory changes. In fact, we conjecture that the migration of examples between epistemological classes might be a way to gauge the development of a theory: examples first considered anomalies and counter-examples become references and even models as the theory evolves, especially during Kuhnian "paradigm shifts" [Kuhn 1970].

NOTE 2. Assume such a quadrilateral exists. Mark the three equal angles:



Case 1. If A, B, D are the three equal sides, then C $\parallel$ A.
==> angle between C and B = angle between C and D,
which is a contradiction.
Case 2. IF A, D, C are the three equal sides, then B $\parallel$ D.
==> angle between C and B = angle between B and A,
which is a contradiction.
Case 3. If .....

NOTE 3. One can monitor such conflicts in a computational model by using global and local Constraint Satisfaction Counts (CSC's): one global CSC to keep track of overall progress (e.g., 1 of 2 constraints satisfied) and one local CSC for each individual constraint. Thus, this situation would change the local CSC for "three equal angles" from 0 to 1, the local CSC for "three equal sides" from 1 to 0, and cause no change in the global CSC.

There are many ways the examples can be scored, like a binary "yes or no", a discrete "success, better, no-change or worse" or some numerical scale. The range chosen affects the judgement, agenda and other control processes. The current implementation of the CEG system uses the S-B-NC-W scheme.

NOTE 4. A function f is said to be "one-to-one"
if when f(a) = f(b), then a = b.

To say it another way, if a and b are different points, their images will be different; i.e., the function does not repeat a value. Ways in which a function can fail to be one-to-one are to repeat values (e.g., y=x**2 on [-1,1]) or to have "flat" spots (e.g., step functions).

NOTE 5. The following, found by solving simulatneous linear equations, will solve the problem:

Y = X**2 when X < .5
Y = 4/3 X**2 - 1/3 when X $\geq$ .5

Solving linear equations can be thought of as a way of finding a point which is constrained to lie on two lines whose equations are given. Simultaneous equations can often be used to express constraints; for instance to find the parabola (y = ax**2 + c) with the suitably fudged coefficients to solve problem Fcn.3, it suffices to solve:

0 = a/4 + c
1 = a + c

While such mathematical methods are elegant and efficient when they work, they are not all that useful for most CEG problems. Furthermore, they are inadequate as a computational description of the CEG process.

References

Abelson, H., and A. DiSessa, Turtle Geometry. MIT Press, Cambridge, 1981.

Baker, C. L., Introduction to Generative-Transformational Syntax. Prentice-Hall, New Jersey, 1978.

Buchanan, B. G., "The Role of the Critic in Learning Systems". Presented at the NATO Advanced Research Institute on Adaptive Control and Ill-Defined Systems, Devon, England, 1981. To appear in Adaptative Control and Ill-Defined Systems, Selfridge, Rissland, Arbib (eds), Plenum Press, in press.

Byte, Vol. 6., No. 8, August 1981.

Clement, J., J. Lochhead, and G. Monk, "Translation Difficulties in Learning Mathematics". American Mathematical Monthly, Vol 88, No. 4, April 1981.

Collins, A. and A. Stevens, Goals and Strategies of Effective Teachers. Bolt Beranek and Newman, Inc., Cambridge, Mass., 1979.

Gelbaum, B. R., and J. M. H. Olmstead, Counterexamples in Analysis. Holden-Day, San Francisco, 1964.

Hayes-Roth, B., Projecting the Future for Situation Assessment and Planning. RAND Note N-1600-AF, Nov. 1980.

Hayes-Roth, B., and F. Hayes-Roth, "A Cognitive Model of Planning". Cognitive Science, Vol. 3, No. 4, 1979.

Hawkins, D., "The View from Below". For the Learning of Mathematics. Volume 1, No. 2, FLM Publishing Association, Quebec, Canada, November 1980.

Kuhn, T. S., The Structure of Scientific Revolutions. Second Edition. University of Chicago Press, 1970.

Kuhn. T. S., "Second Thoughts on Paradigms". In The Structure of Scientific Theories, Suppe (ed), University of Chicago Press, 1974.

Lakatos, I., Proofs and Refutations. Cambridge University Press, London, 1976.

Lenat, D. B., "Automatic Theory Formation in Mathematics", Proc. IJCAI-77.

Levi, E. H., An Introduction to Legal Reasoning. University of Chicago Press, 1949.

Minsky, M. L., "A Framework for Representing Knowledge". In The Psychology of Computer Vision, Winston (ed), McGraw-Hill, 1975.

Newell, A. and H. A. Simon, "GPS, A Program that Simulates Human Thought". In Computers and Thought, Feigenbaum and Feldman (eds),

McGraw-Hill, 1963.

Polya, G., How To Solve It. Second Edition. Princeton University Press, 1973.

Polya, G., Mathematical Discovery. Volume II. Wiley, New York, 1965.

Polya, G., Mathematics and Plansible Reasoning, Volumes I and II. Princeton University Press, 1968.

Rissland, E. L., "The Structure of Knowledge in Complex Domains". In Proc. NIE-LRDC Conference on Thinking and Learning Skills. Pittsburgh, 1980a, in press (Lawrence Erlbaum).

Rissland, E. L., The Structure of Mathematical Knowledge. Technical Report No. 472, MIT Artificial Intelligence Laboratory, 1978a.

Rissland, E. L., "Understanding Understanding Mathematics". Cognitive Science, Vol. 2, No. 4, 1978b.

Rissland, E. L., O. G. Selfridge and E. M. Soloway, "The Role of Experiences and Examples in Learning Systems". In Proc. Third Annual Conference of the Cognitive Science Society, Berkeley, 1981.

Rissland, E. L., and E. M. Soloway, Constrained Example Generation: A Testbed for Studying in Learning. IJCAI-81, Vancouver, 1981a.

Rissland, E. L., and E. M. Soloway, "Generating Examples in LISP: Data and Programs". In Proc. International Workshop on Program Construction. Bonas, France, September 1980a.

Rissland, E. L., and E. M. Soloway, "Overview of an Example Generation System". In Proc. First National Conference on Artificial Intelligence. Stanford, August 1980.

Rissland, E. L., E. M. Soloway, S. Waisbrot, R. Wall, and L. Wesley, "The CEG System", in preparation.

Russell, B., A History of Western Philosophy. Simon and Schuster, New York, 1945.

Sacerdoti, E., "The Non-Linear Nature of Plans". Proc. IJCAI-75.

Selfridge, O.G., Learning to Count: How A Computer Might Do It. Bolt Beranek and Newman, Inc, 1979.

Smith, R. G., T. M. Mitchell, R. A. Chestek and B. G. Buchanan, "A Model for Learning Systems". In Proc IJCAI-77.

Soloway, E. M., "Learning = Interpretation + Generalization: A Case Study in Knowledge-Directed Learning". COINS Technical Report 78-13, University of Massachusetts, 1978.

Soloway, E. M., and B. Woolf, "Problems, Plans and Programs". In Proc. Eleventh ACM Technical Symposium on Computer Science Education, Kansas City, 1980.

Stefik, M. J., Planning with Constraints. Memo HPP-80-2, Stanford University, 1980.

Sussman, G. J., A Computer Model of Skill Acquisition. American Elsevier, New York, 1975.

Thomas, G. B., and Finney, R. L., Calculus and Analytic Geometry. Fifth Edition, Addison-Wesley, 1979.

Wesley, L., "Learning Racketball by Constrained Example Generation". In Proc. IJCAI-81.

Winston, P. H., "Learning Structural Descriptions from Examples". In The Psychology of Computer Vision, Winston (ed), McGraw-Hill, 1975.

APPENDIX A: A Protocol for Problem Geom2

This is a verbatim transcription of one protocol for Problem Geom2. It shows many of the ideas discussed in the paper. The subject was very verbal and did a lot of thinking out loud. Remarks by the author are indicated by "ER:".

Problem 1.2. Draw a quadrilateral with exactly three equal angles. Uh, I'm tempted to think of a square again. Um, so I will. And three equal angles. Now I'm going to draw in the three equal ones. As right angles. Now is there any way I can change that fourth angle to be not equal? I don't think there is. I'm thinking about stretching sides A and B to be longer, which would also stretch sides C and D. That isn't changing the angle. Stretching is the only thing I can think of that serves the...three angles. So I think that I'm going to try to start over. Draw a quadrilateral with exactly three angles. Exactly three equal angles. Oh. No, that won't work. I just thought that I might be able to move the right end of D down along C, and that that would work, but that changes two angles not three. So, I'm having trouble with this very simple-looking problem. Draw a quadrilateral with exactly three equal angles. I wonder if having right angles is hanging me up? I'm attracted to the idea of starting with a triangle. An equilateral triangle. And there are my three equal angles. Could I somehow get a fourth side in there? Well, I could if I bent in side A, but then that would change the angles. Um, I'm going to make the triangle into a quadrilateral. I'm going to try a right triangle. [Draws again.] Which does not have three equal angles. I'm starting to get embarrassed because this problem looks so easy. [Laughs.] But for some reason, I'm off on the wrong track. OK. Let me see if I can take a new approach. Uh, I'm going to think about quadrilaterals. I know about...sort of trial and error. A rectangle won't work. A trapezoid won't work. Um, and it's starting to occur to me, in fact--this may be more foolish--that this is impossible to do. But I sort of doubt that.

ER: Well, it's always a possibility.

But I sort of doubt that. OK. A parallelogram won't work. Uh...Oh. I just had an image of walking along a line and then making a turn and that reminded me of turtle geometry, and then making another turn that would be the same, and then making another turn that would be the same, and that would give me three equal angles. And then I would draw another line, and now I have five sides. Damn it! So. Could I get anywhere from that? What if I drew the line at point A instead of point...from A to C instead of from B to C? Let me number these: 1, 2, 3, 4, 5, I was drawing 6. I'm looking at the line between A and C. Let me draw that dotted. And I'm reconvincing myself that quadrilateral ACDE does not have three equal angles, and um, I'm sort of getting a blank. I don't know what to do next. Uh, well, I had one idea, which was to...I had an intuition that maybe doing sharper turns, at E and D, might help, and that if I picked the angle I was

turning at correctly, things might work out. But then would the thing...I mean, one obvious one is a right angle. But then you get a square. And we already showed that that wouldn't work. So I'm now attaching some increasing suspicion to the idea that maybe this is impossible.

ER: Do you want me to make any suggestions?

Uh, that's up to you.

ER: OK. Well, you've done a lot on it. Why don't I just say: 90 degrees is an angle that you've explored pretty thoroughly and you don't want to look at that. That's what you just said. And you said something about choosing a different angle, a smaller angle, and you talked about it, but you didn't draw a picture. Why don't you just try it? One last try.

OK.

ER: Your routine number 6, but pick a different sort of angle.

OK. So, I'm going to start going vertically to get things looking nice. Point C--this is drawing 7. And I'm not going to pick a right angle to turn at point E. I'm going to...say, I could pick...I'm anticipating picking less than a right angle for the interior angle and is that going to...? Ooooh! What if I had a polygon that was was concave? Bet that might work. So now I'm envisioning something like...oh, I drew it wrong in 7, so I'll draw it very...and I just noticed I'm tending to use equal sides. And I don't need to use that constraint. So, you have a sharp angle and then another angle that's going to be inside. It's going to hit line CE if I continue too far. And then draw it back to the point where I get a nice looking angle at C that's like angle E and F, drawing 8. This looks like it might work. Just visually, C looks like the same angle as E and F. But G...G is different, so that looks like a possible solution.