A HYBRID STRUCTURED OBJECT AND CONSTRAINT REPRESENTATION LANGUAGE*

David R. Harris
Sanders Associates
95 Canal Street
Nashua, New Hampshire 03061

ABSTRACT

SOCLE is a hybrid representation system in which cells of constraints are identified with slots of frame networks. Constraint formulas are maintained with respect to slots in frame networks and in turn provide for the dependency regulation of values on the frames. This paper illustrates the use of SOCLE and outlines the control structure decisions made for its design and implementation.

I. INTRODUCTION

This paper describes SOCLE** (Structured Object and Constraint Language Environment), a hybrid system in which cells of constraint networks are identified with slots of frame networks. The hybrid system contains a structured object component (essentially FRL [Roberts, Goldstein, 77]) and a constraint component (based on the Constraint Based Programming Language of Steele [Steele, 80]). As such SOCLE's benefits include the following: representation for both structure and formulas, constraint propagation, default reasoning, requirement enforcement, contradiction resolution, and explanation of computations.

While the notion of adding constraints to structured objects is not new (for example [Morgenstern, 84],[Batali, Hartheimer, 80]), SOCLE fully integrates the key features of mechanisms from the constraint and frame paridigms and its architecture offers insights into communication and delegation between the two components. Recently several papers including, [Rich,85], [Vilain, 85], and [Brachman, Gilbert, Levesque, 85] have reported on the advantages of such hybrid solutions for the representational needs of intelligent systems. It is our hope that SOCLE can serve as an additional data point for investigations of the space of hybrid solutions.

A. Background

A frame representation language is a programming language which supports a partitioning of knowledge into both "type" and "part" hierarchies. Individual frame objects are created containing slots which define the object and indicate relationships to other structured objects. The types of computation typically performed by frame based systems include subsumption, defaults, and procedural attachment.

A constraint based language is used to express formulas and dependencies. The underlying mechanism supports (i) bi-directional propagation of values through constraint networks, (ii) recording of dependencies (to be used in support of contradiction resolution, retraction, and explanations of the history of computations) and (iii) persistence of certain types of values.

As we will illustrate with an example below, SOCLE adds computational power to both structured object and constraint paridigms.

B. Motivation for SOCLE

SOCLE was motivated by our work on intelligent engineering assistance programs. Such programs must contain representations for highly structured engineering knowledge and must, in a mixed initiative mode, provide assistance when an engineer changes his or her mind in trying out solutions to complex problems.

It was important for us to develop a completely integrated solution in which typical computation in one component invokes the correct response in the other component. A weak link between components would not have succeeded. The critical consideration which forced the hybrid approach was the fact that formulas relate variables only as they fill a particular role with respect to application objects. This is true for two reasons. First, multiple instances of formulas may be used in describing the same object. For example, a description of a function might make use of a "range constraint" (minimum value + range = maximum value) applied to both the ordinate and abscissa. Each instance requires the instantiation of a separate constraint network. The knowledge about the number of constraint networks to be declared can be stored in the structured object component. Secondly, engineering formulas often are approximations which are not universally applied. Hence

* (c) Copyright Sanders Associates, Inc., 1986. Work reported here was developed under internal research and development at Sanders Associates.

** An architectural term for a projecting foundation piece.

enforcement, dependent on the context, can be expressed declaratively in the structured object component.

In the ensuing paragraphs, we will present SOCLE at the knowledge level and at the implementation level. At the knowledge level we illustrate the need for the hybrid approach with an example and then talk about the expressive power of SOCLE. At the implementation level we discuss issues of control structure: communication and division of labor between components.

II. KNOWLEDGE LEVEL DISCUSSION

A. An Example

This example is motivated by the use of a simple "distance = rate x time" formula in a design problem for air traffic control systems. Figure 1 shows a decomposition of the AIR TRAFFIC CONTROL SYSTEM using "input-from", "objects-tracked", "tracker", and "geographic-coverage" slots.

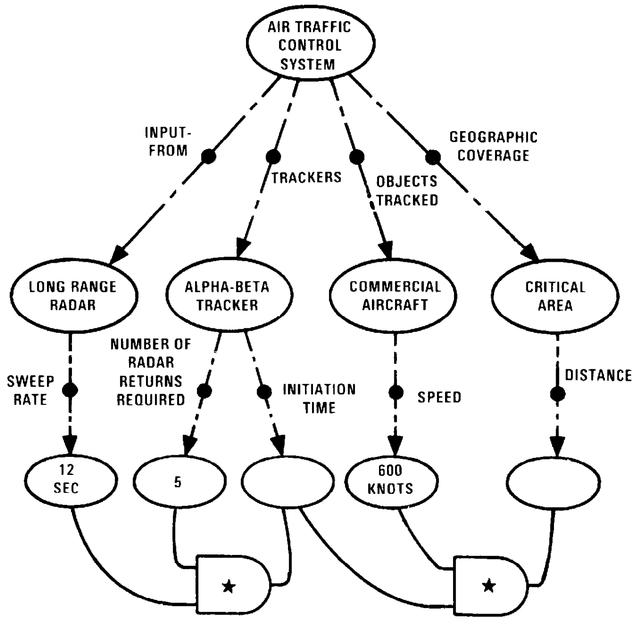

Figure 1. Structured Object Decomposition for AIR TRAFFIC CONTROL SYSTEM Concept.

Slot fillers for each of these parts are LONG-RANGE-RADAR, COMMERCIAL-AIRCRAFT, ALPHA-BETA-TRACKER, and CRITICAL-AREA respectively. Each of these is a structured object which can inherit values, defaults, procedural attachments from generalized concepts.

In such a decomposition, slots which are related through formulas may be functionally far apart. For example, the distance that an aircraft can cover before a track is established is related to the speed of the aircraft, sweep rate of the radar, and the number of hits required for the tracker to establish a new track. Letting

T = initiation time for establishing a track,
N = number of radar returns required to establish a new track,
R = the sweep rate of the system radar,
D = distance covered by the aircraft before a track is established,
and S = speed of a commercial aircraft,

we can quickly establish the formulas:

$$T = R * N$$
$$D = S * T.$$

These formulas are illustrated on the diagram (and conceptualized) as wiring networks.

The advantage of a hybrid approach can be seen from two viewpoints. From the point of view of the structured object component, these wiring networks constrain the values placed on slots. If values are set for "sweep-rate", "number-of-radar-returns-required", and "speed" as shown, then values of 1 minute, and 10 miles can be propagated to the "initiation-time" and "distance" variables, respectively. If, in an exploratory design session, an engineer sets the distance to 5 miles, SOCLE will declare a contradiction and help to resolve it by identifying premises and associated levels of confidence. From this view, what is significant is that variables located on structured objects are regulated using dependency information.

From the point of view of the constraint component, the structured objects provide and maintain the context for constraint formulas. Constraints are only enforced for values that fill particular roles in structured object networks. Subsequent engineering changes can result in modification to these structured object networks, and constraint networks must be adjusted accordingly. Hence in the example, if the "input-from" slot filler is replaced by a radar with a sweep rate of 10 seconds rather than 12 seconds, SOCLE will move the constraint network to the new radar, disconnect the 12 second value from formulas, retract any values for which the 12 seconds was a premise, assert the fact that a new value of 10 seconds is to be used, and propagate appropriate values in the constraint network. This enforcement of formulas between values only as they fill slots in structured object networks is a key feature of the hybrid system.

B. Expressibility

In addition to the somewhat standardized vocabulary of frame systems, SOCLE includes functions which mix notions from the structured object and constraint paridigms. For example, levels of confidence (DEFAULT, SUPPOSITION, BELIEF, and CONSTANT) can be stated for values at particular locations in structured objects.

The declaration of formulas is an important aspect of .using SOCLE. Two methods are available for this. In both methods, the functions for declaring formulas work with pathnames (i.e., sequences of slots whose fillers are frames) for variables. This idea is also used by Morgenstern, [Morgenstern, 1984], in declaring constraint equations in semantic networks.

First, a priori formulas can be declared on generic structured objects. The mechanism for installing constraints starts with structured object based inferencing. We profit from inheritance by declaring the formula on a CONSTRAINT slot of the most general concept appropriate. When an individual structured object is instantiated, procedural attachments are placed along the path to the variable referred to in the constraint. These procedural attachments are charged with installing and maintaining the constraint network when changes are made in the participating structure.

In the example, the "distance = rate x time" formula, referred to above, could be declared on the AIR-TRAFFIC-CONTROL structured object as follows:

```
(air-traffic-control (ako ($value (system)))
     (constraint ($value
     ((multiplier   (at* tracker initiation-time)
                    (at* objects-tracked speed)
                    (at* geographic-coverage
                          distance))))***
```

Formulas can also be declared for variables found only on specific individual structured objects. In this case, the maintenance of context along structured object links is forfeited. As an example, one might declare the formula between "sweep-rate", "number-of-radar-returns-required", and "initiation-time" by invoking:

```
(multiplier    (at radar-43 sweep-rate)
               (at tracker-21 number-of-radar-
                    returns-required)
               (at tracker-21 initiation-time))
```

C. Assumptions

Two assumptions of the current implementation should be mentioned. First, SOCLE supports numbers, symbols, sets, and number unit pairs as slot values to be tied to constraints (Internally, constraint primitives employ functions which understand number conversions and dimensional analysis). Second, it is assumed that all slots which participate in constraint formulas are single valued (i.e. x is the slot filler of s on frame f means s(f) = x).

III. IMPLEMENTATION LEVEL DISCUSSION

This section is organized to describe the control structure issues outlined in Brotsky and Rich's paper [Brotsky, Rich, 1985] on hybrid systems.

*** The hyphenated-expressions indicate that the frames are instantiations of generic frames for RADAR and TRACKER. The difference between the AT and AT* functions is that the first argument to the AT function is the name of a frame, while the AT* function is evaluated when the frame name is bound to the frame on which the a priori formula is defined.

A. Communication

Communication between the two components of SOCLE is performed through a collection of cells which are attached to slots of frames. Frame generated values are pushed into these cells. Subsequently, the cells are used for setting values, retrieving values and explanations, and invoking procedural attachments.

1. Setting Values:

A value may be set through a frame based inference. For example, a request for a value may be answered by inheritance of a default. This value is returned and also stored on the cell attached to the slot. In addition, the confidence level of default is noted so that the value will behave as a default in constraint networks.

A value may also be remotely set through a constraint based inference. Computation in the constraint system proceeds by awakening constraints when new values are set for participating variables. If the variable is in fact the slot of a frame (this information is stored on the cell), then control is passed to the frame system to awaken procedural attachments that reside there.

2. Retrieving Values:

The frame-constraint boundary may need to be crossed to retrieve values. A frame based request for a value is honored by looking on the cell attached to a slot of a frame. From the other side, constraint computations may beg frame networks for values. This occurs in the course of contradiction resolution and retraction. When cells lose values, the frame location is determined and a frame based request for a value is made. If a value is available, it is immediately stored back in the cell as described above.

In summary, all communication between the two components takes place using cells. These cells know their place in both worlds and contain the current state for the variables of interest. The division of labor and the needs for crossing the frame-constraint boundary are the topics of discussion in the next section.

B. Division Of Labor

1. Strategies:

Before explaining the approach we have taken in SOCLE, we might pause to consider two extreme strategies for integrating structured objects with constraints.

On the one hand, it could be the responsibility of the constraint mechanism to perform all frame based inferences. Thus, for example, rather than having a frame retrieval function which uses subsumption to find a value, one could install "inheritance constraint" networks which link together values on all slots of two frames when one subsumes the other. This would lead to some difficulties, however.

Importantly, propagation and contradiction resolution strategies would need to be tailored to support exception links. Also, additional control structure would be required when inheritance is considered prior to formula computation.

On the other hand, one could move all the information stored in constraint nodes onto frame facets. In addition to default, type, and procedural attachments, one might have supplier, reason, and associated constraint pins as facets. The complexities involved in computations for local propagation, retraction, and contradiction resolution could be made the responsibility of frame representation language functions, but now all the checking related to constraint calculation would occur all the time whether or not there was ever any intent to tie a particular frame-slot to a constraint network. At issue, is the percentage of frame-slots in the application domain which can be expected to be tied into these constraint networks. In our work, only about 10% of the slots serve as variables for constraint networks.

We have deemed both of these strategies to be inappropriate. The first is inappropriate due to differences between inheritance and constraint propagation. The second is inappropriate due to the expectation that only a small percentage of slots will participate in constraint networks.

2. Features Of A Good Hybrid System For Intelligent Assistance:

In order to divide responsibility for computations it was necessary to look carefully at the union of the features provided by both mechanisms. With this in mind, we generated the following list of important benefits:

a. Representation of structured engineering knowledge and engineering formulas. This item falls into the province of a frame based system. Of note is the fact that formulas are expressed declaratively on an appropriate frame. When this frame is instantiated, the constraint function (e.g. MULTIPLIER in the example above) is invoked to install the constraint network.

b. Propagation of values which are constrained by underlying engineering formulas. This item is primarily the responsibility of the constraint component. If, however, values are lost in constraint network computation, then control is returned to the frame component to locate potential values there.

c. Default reasoning, wherein default values eagerly assert themselves in formulas, but immediately bow out when they have created a contradictory state. Default reasoning has semantics in both paridigms. An important consideration of the implementation was to ensure that the two notions worked correctly together.

For frame based computation, defaults are used only when values are not present or can not be inherited. In constraint networks, there are potentially two dimensions to be considered. First defaults can be thought of as being the weakest confidence level for assertions. In this sense, the notion in the two paridigms is the same. In addition, however, there is a notion of persistence associated with constraint based computation. Values which are persistent must actively force themselves into formulas when they can.****

In SOCLE, we have placed responsibility for maintaining default state information on the frame component. When defaults are declared they are eagerly pushed onto instantiated frames and hence out into attached constraint networks. Also, when a value is lost (through retraction of a supporting premise perhaps), the transfer of control back to the frame network described in the paragraph above will of necessity discover and re-assert default values.

d. Enforcement of requirements imposed by both structures and formulas. The enforcement of requirements on values is of course exactly what constraint networks are all about. However, one can also declare explicit requirements on a frame. For example, a requirement that speed be within a valid range (imposed by the laws of physics) might properly be placed on a MOVING-OBJECT structure independently of constraint networks. On the other hand, a value for the speed of a particular aircraft may be regulated by a formula which is enforced by a constraint network. SOCLE permits the assertion of values only as they are consistant with both types of requirements.

e. Contradiction resolution based on recordings of premises for inferences. Resolution takes advantage of annotations for the level of confidence that an engineer has in the premise. Contradiction resolution is performed in both the frame and constraint mechanisms. The determining factor is whether or not a new value being set is intended to be a premise or is remotely established through other premises in constraint networks. In the first case, a preliminary investigation can compare the levels of confidence between the new assertion and the old. For example, a default, would bow out to a supposition. The second case, can only be resolved in the constraint network. Values dependent on the old value are retracted, the new value is asserted and an attempt is made to settle out the state of the constraint network. When SOCLE can not automatically resolve the contradiction, it informs the user of the problem and requests that the user either retract a premise or declare a formula's application to be invalid.

f. Explanation of values based on history of computations. Explanation is handled totally by the constaint network, although values to be explained are referenced by their location in the frame network.

IV. CONCLUSIONS

In summary, SOCLE embodies the power of

**** While this feature can be associated with any confidence level, we have chosen to associate it with the weakest level. This is consistent with Steele's implementation.

the above six items: representation for structure and formula, propagation, default reasoning, requirement enforcement, contradiction resolution, and explanation. It is a generally useful knowledge representation language in application areas which contain highly structured knowledge including formulas which tie together variables from the structures.

SOCLE is currently being used on several projects at Sanders. These include projects in system and software requirements analysis, automatic test equipment reprogramming, and reliability simulation.

I would like to thank Chuck Rich for his suggestions and encouragement on this effort. Important contributions to the design and implementation of SOCLE were made by Andy Czuchry, Terry Ginn, and Lynne Higbie.

REFERENCES

[1] Batali, Hartheimer, "The Design Procedure Language Manual". MIT/AI Memo 598, 1980.

[2] Brachman, Gilbert, Levesque, "An Essential Hybrid Reasoning System: Knowledge and Symbol Level Accounts of KRYPTON", Proc. IJCAI-85, Los Angeles, California, Aug. 1985, pp 532-539.

[3] Brotsky, Rich, "Issues in the Design of Hybrid Knowledge Representation and Reasoning Systems". Proc. of the Workshop on Theoretical Issues in Natural Language Understanding, Halifax, Nova Scotia, May, 1985.

[4] Morgenstern, "Constraint Equations: A Concise Compilable Representation for Quantified Constraints in Semantic Networks", Proc. AAAI-84, Austin, Texas, Aug., 1984, pp. 255-259.

[5] Rich, "The Layered Architecture of a System for Reasoning about Programs", Proc. IJCAI-85, Los Angeles, California, Aug. 1985, pp 540-546.

[6] Roberts, Goldstein, "The FRL Prime", MIT/AI Memo 408, 1977.

[7] Steele, "The Definition and Implementation of a Computer Programming Language Based on Constraints", MIT/AI Technical Report 595, 1980.

[8] Vilain, "The Restricted Language Architecture of a Hybrid Representation System", Proc. IJCAI-85, Los Angeles, California, Aug. 1985, pp. 547-561.