Multi Modal Methods of Information Transmission

Jim Brander
Interactive Engineering Pty Ltd
Harris Park
Sydney
Australia
inteng@ozemail.com.au

Abstract

Different methods of transmission of information are described within a network formalization of the analytic operators. Examples are given of where switching occurs between the methods. Control of switching is taking place at the lowest possible level, the operators themselves. It is suggested that, for complex high level tasks, no simple means of problem decomposition and assembly of separate dedicated subsystems is possible, nor is one needed.

Introduction

The desire for different modes of reasoning to attack different problems seems reasonable when the limitations of various current methods are assayed. The obvious problem with different modes is that someone had to understand the problem completely to choose the mode - not very likely for reasonably hard problems. Success depends very much on the level of reasoning being attempted - are we attempting a rapid response to a complex scene, or the slow and painful accretion of new concepts over a period of years. Even at these extreme ends of the spectrum, some underlying flexibility of form and capability for self -modification is required if we are to use the term "reasoning". There have been many attempts to decompose problems into disparate elements to suit preconceived notions of reasoning methods or efficient algorithms. Except in narrow areas, the decomposition and reassembly process is rarely successful without destroying the essence of the problem, or requires a large effort on the part of the human users to hold all the pieces together in phase.

It might be worthwhile to analyze what is meant by multi-modal reasoning. Are there different forms of reasoning, exemplified by Expert Systems and Constraint Handling Systems, or are they merely narrow interpretations of more general reasoning. Are we searching for a common substrate, or for glue that will bind the methods that have been implemented. The search for a "glue" is reminiscent of the early descriptions of Expert Systems. An algorithm for combining rules would be described, followed by a description of the properties of the Help system to be implemented later that would explain the ES results. The flexibility and reasoning power of the Help system would far surpass the limitations of the Expert System algorithm, to the point where if the Help system ever became available, one could throw away the Expert System.

Similarly, if we could find a glue or "super-algorithm" which understood enough about the operation of each of the different modes of reasoning to be able to assign tasks, it could do all of the reasoning, perhaps not as efficiently but certainly more completely. Each of these modes of reasoning has made certain assumptions to allow operation in its specific domain. While these assumptions are at least partially understood by the users, enthusiasm for a particular mode often outweighs careful analysis of the potential variability of the overall problem structure.

We might visualize the different modes of reasoning as different forms of information transmission among nodes, as

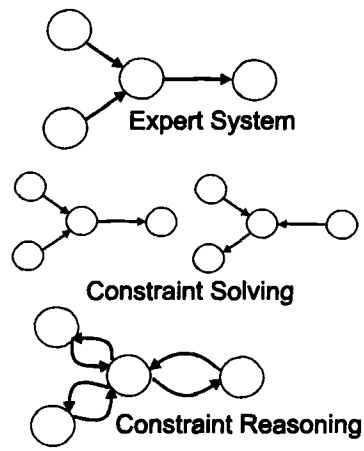

Figure 1

In the Expert System case, we might use goal directed searching or dataflow. In the Constraint Handling case, we allow alternatives at the boundary nodes, and turn the constraints around. In the case of Consistent Reasoning, we use the pathways within the statements, and any of the other forms of reasoning is a subset, with operations on the information and its direction of propagation being determined at the operators.

Perhaps we should look more closely at the analytic operators we use. It turns out that in our current implementations, we can hardly combine two analytic operators before we are forced to start stripping away inferences. With each inference lost, it becomes more important that a human user understand the intended use of the implementation. Our methods of mathematical and logical analysis are incomplete, and have inconsistencies needing to be papered over by a human user, particularly at the interface between logic and numbers, or where existence of objects is relevant, or where topological change can occur during analysis.

A network implementation of analytic operators will be used to highlight some difficulties with a mode integration approach. The network is intended for use in high level planning and design, where there can be no artificial distinction among rules, relations and constraints. Planning requires that a potential structure be assembled, evaluated perhaps using consistent reasoning, and then the planning system move to a new state, inconsistent with the last, and continue. As different forms of reasoning are needed, different modes of information transmission through a common structure are utilized. This paper will attempt to show through examples how the integration of different methods of information transmission through a common realized structure seems essential if we are to find a way of combining different modes of reasoning.

Reasoning Structure

The elements of the network structure are the analytic operators, ranging over objects, numbers and logic, and being close analogues of the common operators. The main distinction with previous attempts at network formalism is that an attempt has been made to fully model the basic properties of each operator so that no potential inference is lost. Another difference is the dynamic nature of the structure. A simple logical constraint may serve as an introductory example.

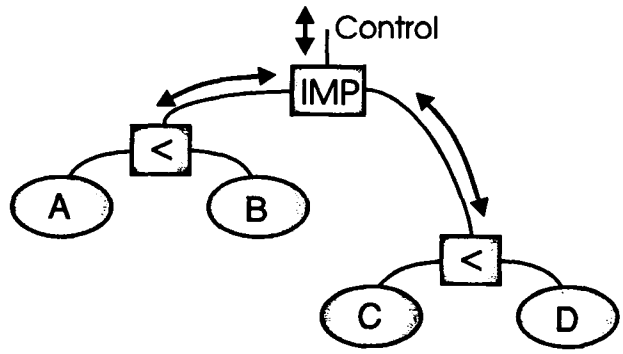

Figure 2

The structure implements

IF A < B THEN C < D

but contains far more than would be initially assumed from the textual form. The nodes and links may contain logical states, singular numeric values, alternative numeric values, lists of objects or lists of alternative lists. The arrows on the connections indicate the potential directions of information flow, in that information can flow in either direction, and, if consistency is maintained, can flow consecutively in either direction. If A is actually less than B, then a TRUE flows out of the less than operator to the implication operator. If its control pin is asserted, a TRUE flows out of the consequent pin to the other less than operator. However, if C had been greater than D, then a FALSE would have flowed the other way. If control had not been asserted, a FALSE may have flowed out of the control pin. The structure is clearly a logical model, and its application to constraint reasoning and rule based reasoning should also be clear. An essential point is that it is not using Boolean logic. To do so would destroy too many potential inferences. Only if the problem is perfectly static can we be sure the inferences discarded will never be needed. This is an example of where the methods of analysis we choose can serve to defeat us. Logic, both in our heads and in a complex problem, is better described by a squirming mass of connections than it is by numbers, which sit as a thin layer on top of logic.

If we wish to combine methods of reasoning where the implementations have destroyed many inferences, either the system combining them may need to recreate the inferences, or we should not have destroyed them in the first place.

The conversion between text form and network structure is not always direct, as inferences may exist for us in our understanding of the text form that would not survive direct conversion to network form with its operator independence.

One such case is Figure 3.

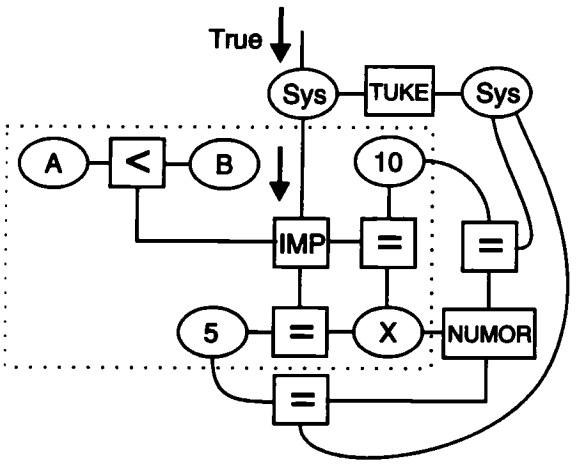

Figure 3

IF A < B THEN X = 5
ELSE X = 10

We can easily conclude from this statement that X may have the values 5 or 10 (the range 5,10). When a statement such as this is converted to network form, additional structure is added to allow the inference (and structure to control the inference is also built - the statement itself may not be unconditionally true).

One other element needs explanation for the following discussion. A cluster of operators forming a statement itself becomes an element attached to a logical spine, over which control may be exercised, the spine beginning at a logical variable. If the spine is not activated, an Unknowable state is used to control the invalid statements, as a False state would imply inversion.

The realized network form allows extensibility, by a user and by the operators of the structure acting directly upon it to change the topology or accrete new structure. This extensibility is in clear distinction to other methods which rely on a stack, up which they may become stuck.

The structure which performs the reasoning is both capable of change to its topology, and recovery from that change, unlike other methods which impose directional barriers as a result of either a misunderstanding of the meaning of analytic operators or as an artifact of their implementation.

Information Transmission and Structural Change

Transmission of information through the structure varies among

Searching
Dataflow with killing
Consistent dataflow
Dataflow with structure growth
Structure assembly and activation

depending on the purpose to which the structure is currently being put.

Searching moves through a structure initially empty of information, attempting to return to the search starting point with useful information. Searching is recalled if dataflow overrides the need. Searching may instigate structural change.

Dataflow with killing causes information to be pushed down applicable pathways. If new information is to replace old, the old information is killed (the relevant part of the structure is emptied of information) before new information is transmitted.

Consistent dataflow allows new information to overlay old without killing, where the new information is consistent with the old, as a new range of 3..7 is consistent with an existing range of 2..10. Constraint dataflow is not limited to numbers, objects also being provided as alternatives.

Dataflow with structure growth implies that arrival of information at a node causes growth of new structure, through which information is again transmitted, either from the originating node, or from the other structure to which the newly created structure becomes attached.

Structure assembly and activation describes where existing structures are sought and selected based on particular properties, they are connected and then logically activated by driving logical states to their spines.

Backtrack can occur on structure creation and change, all of the logical structure being made out of the same stuff.

All of these methods can be used together in any parallel combination or sequence. A search through a structure may identify an area suitable for consistent reasoning, which on completion or failure causes killing of information or creation of new structure. Constraint reasoning on some area of the structure may allow transmission of information which then permits constraint reasoning in another area.

Some Examples

Bridging Methods Of Reasoning

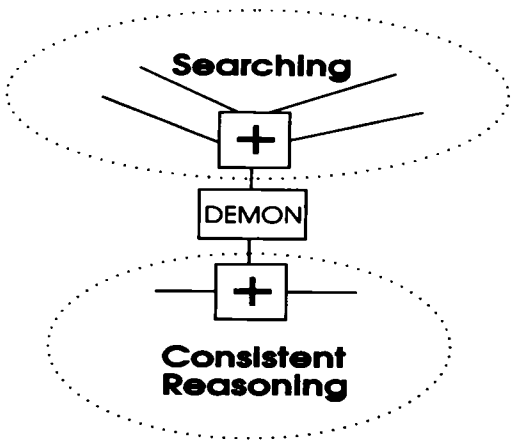

Figure 4

Some constraint handling tools (CHARME) provide a Demon operator which will allow constraint solving to fall back to a programmatic means when appropriate - that is, when a change in the range occurs, a procedure is run. The analog for a network where programmatic means does not exist is a DEMON operator which can be built into the network structure, as shown in the diagram. Here the structure is identical in form on either side of the operator, the only difference being the method of transmission of information Control can be exercised so that bridging only occurs when the range has fallen to a low level, or when a unique value is presented. A fall back to a programmatic means is an admission of failure of the reasoning paradigm, implying the initial means is insufficiently extensible for the problem.

List Indexing

Within the network, lists may be indexed, as ASD[X], where X is singular, or has alternatives. Figure 5 illustrates the structure obtaining where there are two alternative values for the index, and the objects indexed are numbers.

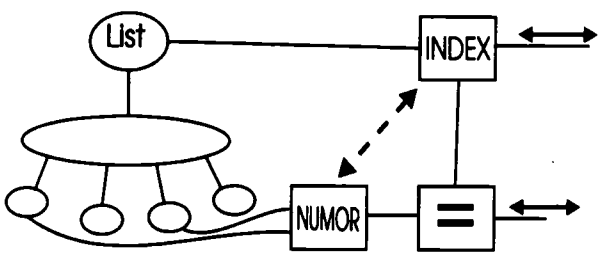

Figure 5

The index may itself be a list, as

ASD[{BFG[GHJ[X,{FGH[3]}],4],PQR}]

The [ ] indicates the index value, the { } indicate a list structure.

Here there is a mixture of single values, alternatives and lists. The index operator need understand nothing of the structure of the list, instead restricting itself to the operation of taking the element relevant to it from the list, then creating a new index operator and passing the remaining index to the new structure. If there are ranges in the index value, these ranges cause new operators to form, with initial connection to all the alternative objects. Later pruning either on the index or the output list reduces the number of alternatives, destroying the operator supporting the alternatives if the alternatives reduce to a singular value.
The same transmission mechanism that transmits lists in goal directed searching and dataflow also transmits alternatives in Consistent Reasoning, so it doesn't matter what mixture the index is of different "modes of reasoning". There may be areas of the model where there are too many alternatives to handle, so this part of the model will wait for other modes of reasoning to reduce the variability to the point where consistent reasoning can begin - another example of the essential intertwining of different modes of reasoning.

It might be claimed that this form of list indexing is no more than a crude form of parsing, except that it should be noted that the result of all this generational and transmission activity is to finally connect two or more objects, without any knowledge about direction of information flow, or if information flow is currently possible, and to have the method of connection reversible, both by killing and backtracking.

Start Hint

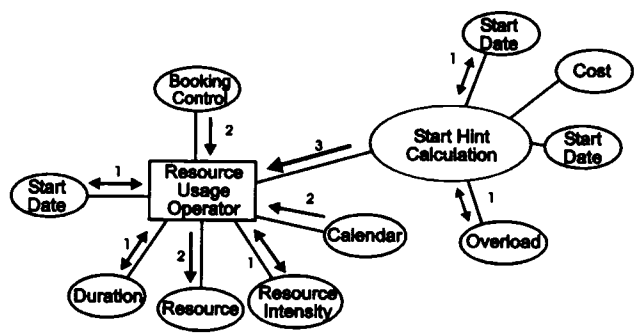

Figure 6

Resource usage is an important aspect of project planning. A resource usage operator in a planning network has consistent information on most of its connections, but also needs to handle information which is inconsistent with that already sent. A Start Hint is typical - there is some calculation that says where resource booking should preferentially occur, and the calculation changes its output value depending on the range and position of the available bookings. This new value, inconsistent with the last, must be transmitted through a network which everywhere else is using consistent dataflow. Figure 6 illustrates the different types of transmission - links marked with a 1 are using bidirectional dataflow (that is, new values flow back and forth, overlaying the old), links marked with 2 have a value that is transmitted once, and the link marked with 3 has information that is repeatedly killed and re-transmitted. The change in method of data transmission is occurring at the lowest possible level, at each operator. In this case, the receiving operator must understand its role. If killing escaped this single connection, all other information in the network would be destroyed. The example in shown in Figure 5 may not be defining, but it is a good example of how sterile a single method of "reasoning" would be.

Application Area - High Level Planning

Business or project planning is typified by the construction of a model which simulates the expected behavior of the real activity. The more complete and extensive the simulation, the more that people can come to understand the issues involved and the consequences of their actions. There will usually be constraints on the plan, there may be options which serve as cases, and there will usually be rules that need to be built into the model.

Consistent - Inconsistent - Consistent

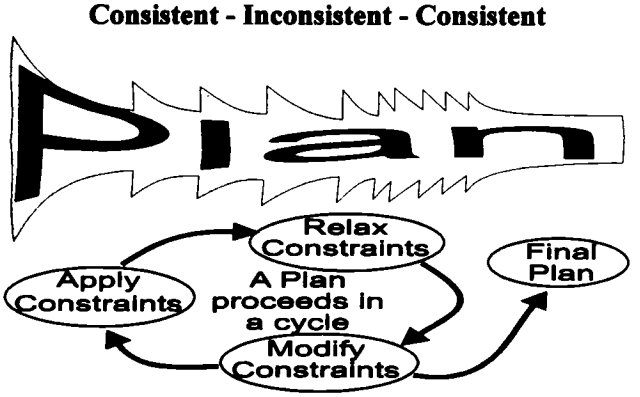

Figure 7

The values used in Constraint Reasoning remain consistent as a way of ensuring that no constraint is violated. This works well until a situation develops where a change is necessary - there just isn't enough resource to do the job on time. Most plans go through many iterations before stabilizing. If we look a little more closely at each iteration, we can see a multiplicity of methods in action.

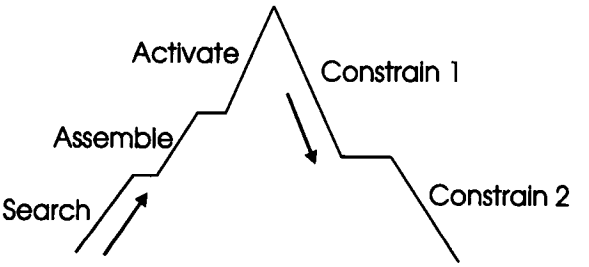

Figure 8

The planning system needs to find and assemble the elements of the plan, and then handle waves of constraint action. It can't start to constrain the use of resources until it is known how much resource is to be used. We don't know whether a range even exists until other areas in the reasoning model have stabilised - 0 does not describe existence. A complex plan has many levels and local areas at which iteration may be occurring at different phases, not the simple sequential phasing shown here..

A planning system that could not automate all of the planning cycle would be of limited use, requiring the human users to fill in the gaps in its analytic capabilities, and risk loss of phase among the components. The method of information transmission needs to smoothly change from Searching to Consistent Dataflow to Dataflow with Killing, so that a new state may be found that is inconsistent with the previous one. As killing proceeds, some of the structure may be taken apart and destroyed, and new structure built in its place.

As already mentioned, Boolean logic is inappropriate in a high level area because of its destruction of knowledge. It should also be evident that a destructive method of reasoning, reductio ad absurdum, is inappropriate where a suitable structure has still to emerge as part of the planning process.

As an example of rules, there may be a clause in the project contract describing incentive payments that apply to completion.

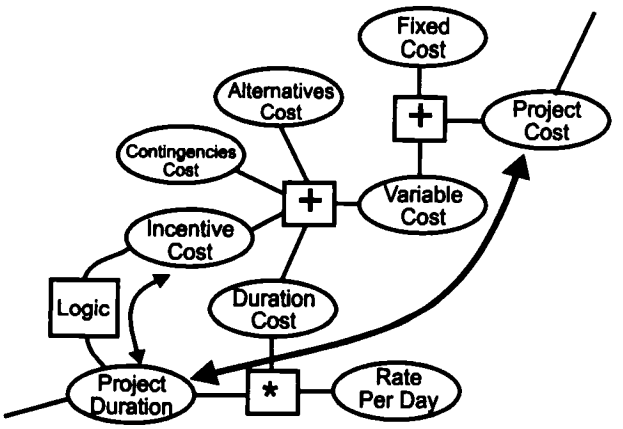

Figure 9

In simple form, a rule linking project duration and project cost might be

IF ProjectDuration < 200 THEN Incentive = 400
ELSE Incentive = 0

This rule (or is it a constraint) is built into the network model, allowing an undirected connection between duration and cost. Initially, the range of ProjectDuration spans 200. Additional constraints may be applied on either cost or duration, with resultant switching occurring at an operator within the structure of the rule to constrain ProjectDuration or Incentive. Some Constraint Handling systems have attempted a kludge to connect between Boolean logic conditions and constraints - each kludge pushes us further away from a seamless handling of complex problems. The rule shown here is simple, but may instead be of arbitrary complexity, and use any mixture of the techniques in its evaluation.

Conclusion

This brief overview of different methods of information transmission through a network structure may have illustrated how multi modal reasoning should have a common substrate.

Hopes that we can somehow implement large pieces of a problem using different methods of reasoning, then combine them at a high level are likely to prove false. Examples in high level planning illustrate that the planning problem would be destroyed if it were cut into elements seemingly suitable for different modes of reasoning. Put another way, the only level at which different forms of reasoning will be successfully combined is at the level of the analytic operators, because only they are close enough to the problem to decide how particular areas need to be handled.

The Expert System and Constraint Handling modes of reasoning, either in isolation or in macro combination, seem far too narrow to describe the methods of reasoning required in the fields of high level design and planning.

References

Cras, J-Y. 1993. A Review of Industrial Constraint Solving Tools. *A.I. Intelligence*, Oxford, U.K.

Brander, J. and Dawe, M. 1998. Use of Constraint Reasoning to Integrate Risk Analysis with Project Planning. *The International Journal of Project and Business Risk Management.* Forthcoming