Goal Relationships in Blackboard Architectures

Victor R. Lesser
Daniel D. Corkill
Joseph A. Hernandez
Robert C. Whitehair

COINS Technical Report 88-19
March 1988
Supercedes COINS Technical Report 87-24

Abstract

How can we ensure that systems employing a blackboard architecture make appropriate focus-of-control decisions? This is a complex problem requiring the system to relate the predicted results of future activities to existing hypotheses and to stimulate activities along promising problem solving paths while inhibiting activities found to be redundant. This paper shows that mechanisms for achieving these capabilities can be introduced as natural extensions to a unified data-directed and goal-directed control framework. These mechanisms are based on adding new *goal relationships* and a new goal type, *inhibiting-goal*, to the unified framework. These additions improve the system's ability to evaluate potential activities. We provide examples demonstrating the benefits of these mechanisms.

This research was sponsored, in part, by the National Science Foundation under CER Grant DCR-8500332, and by the Office of Naval Research under University Research Initiative Grant Contract N00014-86-K-0764, and under Contract N00014-79-C-0439.

1 Introduction

Making appropriate focus-of-control decisions in a blackboard-based problem solver is difficult because the full range of interrelationships among partial results is not readily observable. The blackboard architecture permits the same partial results to be used in many contexts. Thus, producing a specific result may affect many alternative solutions. The problem space is represented at multiple levels of abstraction on the blackboard, and multiple solution paths for the same result may be available. This provides the problem solver with flexibility in choosing problem-solving activities. However, by providing many solution paths, it becomes possible to rederive results along one of the alternative paths without recognizing the redundancy until the final step. Furthermore, the asynchronous, opportunistic style of problem solving leads to situations where it is unclear whether a solution is missing due to a lack of data, in which case the solution will never be generated, or due to a lack of processing, in which case additional work will drive up the low-level data needed to produce the solution. Thus, exploiting the richness of problem-solving capabilities in a blackboard-based system and at the same time making intelligent control decisions is a formidable task [1,2,3].

Several years ago, we presented extensions to the cooperating knowledge source architecture of Hearsay-II [4] that unified data-directed and goal-directed control [5]. This was a first step toward developing the needed interrelationships among actions and results necessary for making intelligent control decisions. In the interim, we have gained considerable experience with this control architecture. In particular, we have identified the need for a new type of goal and for local relationships among goals. These extensions allow us to more accurately relate the predicted results of future activities to existing results in order to make more informed tactical control decisions [6,7]. This is in contrast to the work by Durfee and Lesser [8,9] and Hayes-Roth [10] that involve interrelationships that are more global in character and relate to strategic control decisions.

In Section 2 we briefly review the unified data-directed and goal-directed control architecture. Section 3 describes the new relationships we have defined to implement the desired control capabilities. Section 4 outlines how these new mechanisms work and Section 5 is a brief presentation and discussion of our experimental results.

2 A Review of Goal-based Control

Figure 1 presents a high-level schematic for the integrated data-directed and goal-directed control architecture as implemented in the DVMT [11,12]. The basic Hearsay-II architecture is modified to include a *goal blackboard* and a *goal processor*. The goal blackboard, which mirrors the data blackboard in dimensionality, contains goals representing *intentions* to create particular results on the data blackboard. Goals provide an abstraction over the potential actions for achieving a particular type of result and allow the system to reason about its intentions independently of the particular knowledge source (KS) actions at its disposal.

The two general classes of goals are *data-directed* and *goal-directed*. The blackboard monitor uses domain knowledge to create data-directed goals in response to the addition or modification of hypotheses on the data blackboard. Each data-directed goal specifies the range of hypotheses that could result if the triggering hypotheses were extended at the same blackboard level or abstracted to the next higher level.

Since the creation of a goal does not guarantee sufficient information on the data blackboard to execute a KS to satisfy the goal, the goal processor runs a *precondition procedure* for the applicable KSs to make a detailed examination. When results indicate that a KS has sufficient information to satisfy the goal, the goal *triggers* a KS instantiation (KSI). The scheduler assigns the KSI a priority rating and places it on the *scheduling queue*. The scheduler assigns priority by first determining the set of goals that may be satisfied by a KSI's predicted output. It then computes the KSI's rating as a weighted average of the ratings of the potentially satisfied goals and the credibility of the predicted results. If sufficient information is not available

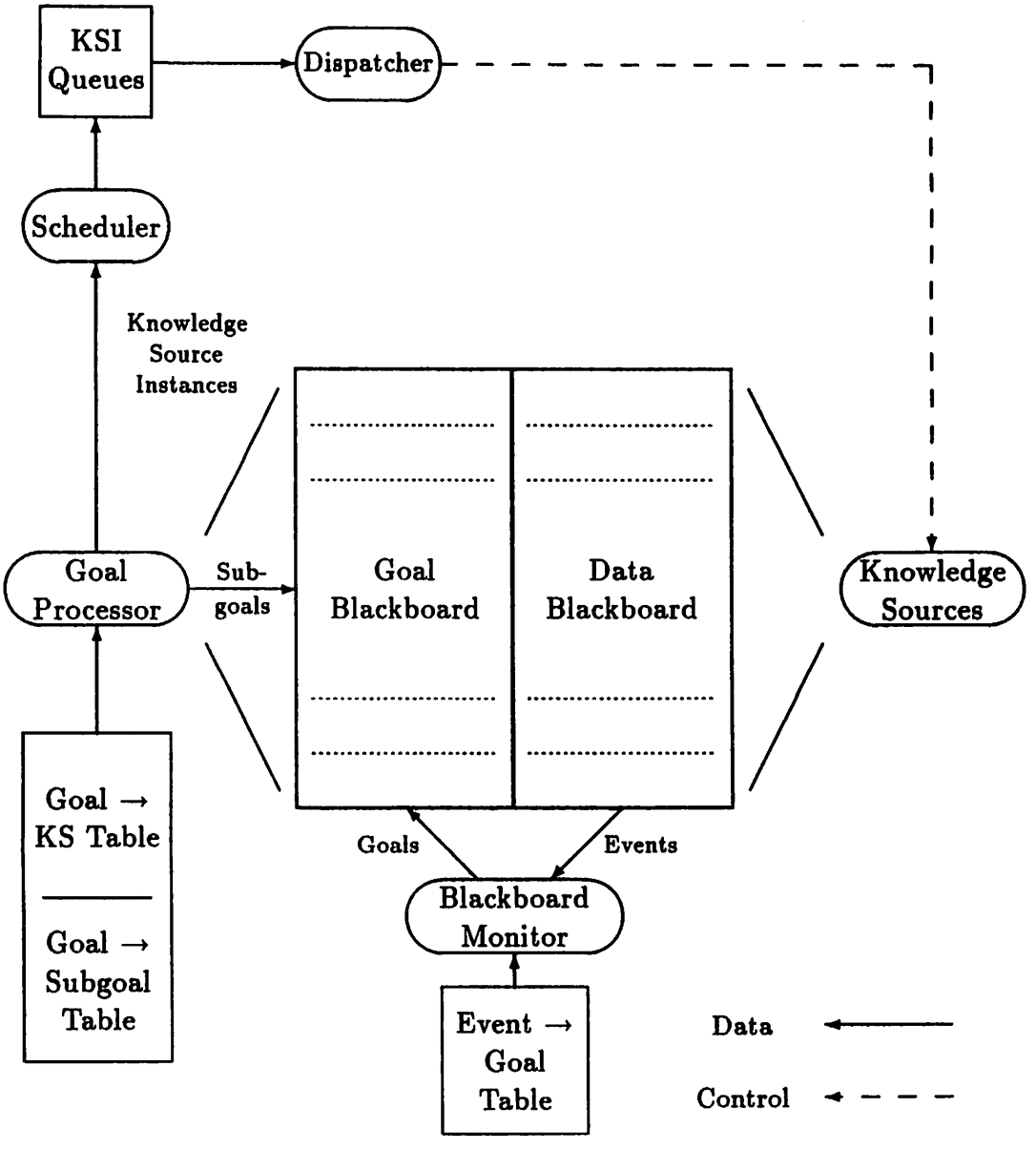


Figure 1: Integrated Data and Goal-Directed Control

to run a KSI, the goal processor creates *goal-directed goals* (subgoals) for driving up low-level data to be used to satisfy the original goal. Subgoals are rated the same as the original high-level, highly-rated goal. Thus, the ratings of low-level KSIs that could lead to satisfaction of the high-level goal will be increased.

Goal-based control does not require that control decisions be made in a top-down, *goal-directed* manner. We use goals to make both data-directed and goal-directed control decisions, and the classical data-directed/goal-directed dichotomy is represented in our approach by the relative ratings among goals. By adjusting its KSI rating computations, the scheduler can bias the system towards goal-directed or data-directed control. Goal-based control attempts to incorporate domain data to build an appropriate control abstraction that will predict the type of results which can possibly be generated. This permits the system to develop non-local focus-of-control strategies that take into account the interactions of work on data in different parts of the problem-solving space.

3 Goal Relationships

We have found that by reasoning about the interrelationships among the goals we can achieve more sophisticated control. Since goals represent an abstraction over a set of possible hypotheses, they capture desired results independently of the actions available to the system for achieving these results. In this way three important questions necessary for effective control can be answered by relating goals to each other:

- Can the same results be obtained by working on different goals?
- Will working on two distinct goals generate distinct partial solutions?
- Will work on a goal allow the system to differentiate between mutually exclusive solutions?

We use the following goal relationships to answer these questions.

assistance: One goal, g1, is said to assist another goal, g2, if satisfaction of g1 implies satisfaction of g2. The assistance relationship identifies those goals which represent alternative approaches to generating a particular solution.

competition: Two goals, g1 and g2, are competing if they both specify the same blackboard level and there is no possible hypothesis that will satisfy both goals. By checking if two goals are competing, the system can determine if the knowledge sources they have triggered will generate distinct results.

subsumption: Goal g1 subsumes a second goal, g2, if the specifications of g2 are completely encompassed by the specifications of g1.

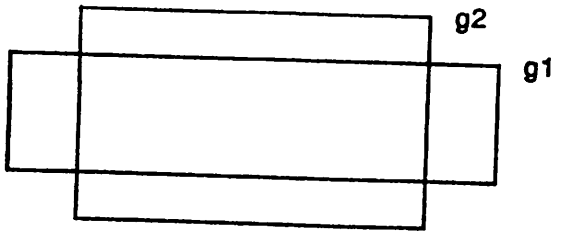
subsumption-inhibition: We have added a new type of goal, called an *inhibiting-goal*, to identify redundant work that can be eliminated. Goal g1 completely inhibits a second goal, g2, if g1 is an inhibiting goal and g1 subsumes g2.

assistance-inhibition: Goal g1 partially inhibits goal g2 if g1 is an inhibiting-goal and g1 assists g2. The assistance-inhibition relation limits work to those areas of g2 not encompassed by g1.

cooperation: Two goals are cooperating if none of the previously defined relations hold and it is possible for the goals to produce information that may be incorporated into a single result at some point in the future.

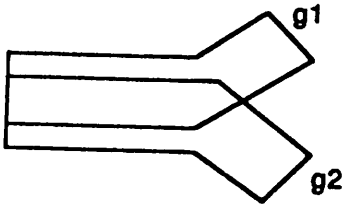
independence: Two goals are independent if none of the previously defined relationships apply.

To clarify these definitions, assume we represent a goal as a two dimensional rectangle, $[(x_a, y_b)\ (x_c, y_d)]$, and a result that satisfies it as a sequence of points $((x_a,y_0)...(x_{(a+k)},y_k)...(x_c,y_{(x_c-x_a)}))$ where $y_b \leq y_i \leq y_d$. The diagrams in Figure 2 show the different goal relationships.



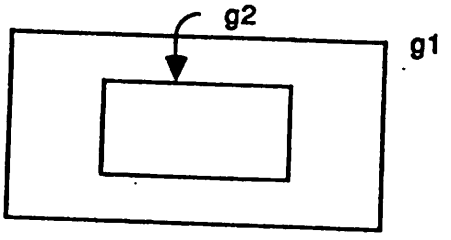
g1 assists g2

a. assistance



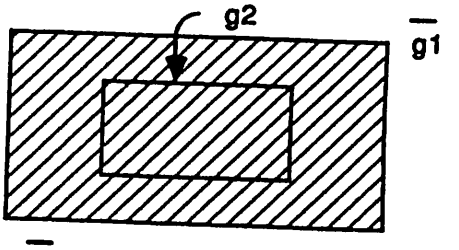
g1 competing with g2

b. competition



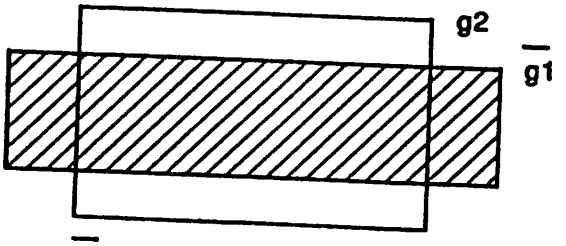
g1 subsumes g2

c. subsumption



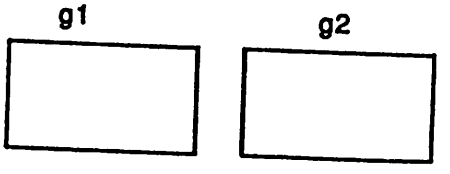
$\overline{g1}$ subsuming-inhibits g2

d. subsumption-inhibition



$\overline{g1}$ assisting-inhibits g2

e. assistance-inhibition



g1 cooperating with g2

f. cooperation

Figure 2: Goal Relationship Diagrams

4 Processing Goal Relationships

The goal relationships described in section three are used in two distinct ways, to prevent redundant processing through the use of inhibiting goals, and to efficiently schedule knowledge sources for execution based on their local context.

4.1 Inhibiting Goals

We have found it necessary to add a new goal type, described earlier as an inhibiting-goal, to augment the existing data-directed and goal-directed goal types. The original formulation of the goal-based control approach was unable to control redundant processing. After deriving a high-level result with sufficient confidence, the only way to minimize redundant activity was to decrease the ratings of the goal and subgoals that generated it. This method prevented additional work only on the original solution path used to derive the high-level result. It did not limit activity on any of the alternative paths that lead to the same result. In order to effectively control redundant processing, a separate mechanism was needed to eliminate derivation of *any* intermediate result that would eventually produce the high-level result. Thus, when a satisfactory, high-level result is produced, this new mechanism allows the system to work only on data it determines to be independent, competing, or cooperating in relation to the high-level result.

An inhibiting-goal and its associated subgoals are generated when the system determines that sufficient work has been done on refining a high-level result. All knowledge sources are then inhibited from producing results that are subsumed by the inhibiting-goal. The specification of the inhibiting-goal is taken from the characteristics of the high-level result. By specifying a tolerance around the inhibiting-goal, its characteristics can be generalized to extend the range of inhibition. This can eliminate solutions that are similar, though not identical. This is appropriate in environments where answers that have characteristics close to the correct answer are acceptable. The algorithm we have implemented is the following:

```
FOR each newly created hypothesis
   IF (hypothesis.level ≥ *inhibiting-goal-creation-level*) AND
      (hypothesis.rating ≥ *inhibiting-goal-creation-threshold*)
   THEN
         Create inhibiting-goal and associated inhibiting-subgoals
         For each stimulating-goal subsumed by the inhibiting-goal
            Terminate efforts to satisfy the subsumed goal
         For each stimulating-goal assisting the inhibiting-goal
            Restrict processing in areas encompassed by the inhibiting-goal
```

4.2 Local Context

Along with inhibiting activity based on high-level results, there is also a need to inhibit activity based on a more local context. For example, since a KSI's rating is based partly on the ratings of the goals it is predicted to satisfy, if any of those goals are satisfied before the KSI is invoked, and if the KSI cannot improve on the results used to satisfy the goal, then the KSI should have its rating decreased. Thus, how a goal is satisfied is an important issue. If the scheduler gave priority to the KSI with the most comprehensive triggering goal, its results might satisfy other goals and eliminate the need to execute their triggered KSI's. The following situation demonstrates this point.

Consider the pending activities KSI1 and KSI2, generated from work on two different derivation paths. If executed, KSI1 would produce result R1 which would subsume R2, the result of executing KSI2. Thus, executing KSI1 first would make KSI2's results redundant, so the scheduler should give KSI1 priority. However, from a local, data-directed perspective, KSI2 might be given higher priority, even though KSI1 is the more promising of the two. This can occur if the scheduler incorporates an average of input data credibility in its KSI rating function, and if KSI1 uses lower rated data in addition to highly rated data used by KSI2. For example, KSI1 may generate R1 by extending highly credible data into areas of weak data, and KSI2 may use only the highly credible data to produce the highest rated component of R1.

Our earlier approach to rating KSIs tried to balance the quality of the predicted

result with its scope. However, we found that the right balance seemed to be situation dependent. Too much priority to scope had the undesirable consequence of making the problem solution search too depth-first, while too much priority to quality led to redundant activity as illustrated in the above example. Instead, we found that, to choose among the pending KSIs, we need to explicitly take into account the relationships among their predicted outputs.

Using the goal relationships specified in the previous section, the system can form a local understanding of why a KSI is scheduled to be invoked and may instead invoke a different KSI which produces the same results more efficiently. In general, before executing a KSI, the Local Context mechanism examines the KSI's triggering goals and searches for a more comprehensive KSI which would also satisfy these goals. If this more comprehensive KSI produces an actual result that is as good in the subsuming area as that expected from the less comprehensive KSI, the subsumed results are removed from the output set of the less comprehensive KSI. This is implemented as a combination of the following two mechanisms:

```
FOR every goal in a KSI's triggering-goal list
    For each satisfying hypothesis
        IF the hypothesis subsumes any element of the KSI's predicted outputs
            AND the predicted output can not improve the hypothesis in any way THEN
                Remove the subsumed item from the KSI's set of predicted outputs
                Recalculate the KSI's rating

Prior to invoking the highest rated KSI
    IF the KSI's assisting goal list is non-nil THEN
        Invoke the KSI triggered by the most comprehensive assisting-goal
    ELSE
        Invoke the original KSI
```

5 Experimental Results

The Distributed Vehicle Monitoring Testbed (DVMT) simulates a network of vehicle monitoring nodes, where each node applies simplified signal processing knowledge

Table 1: Experiment Summary.

Exp	Env	Mech?	KS ex	Hyps	Goals
E1	1	no	184	246	488
E2	1	yes	64	157	443
E3	2	no	207	335	661
E4	2	yes	130	297	712
E5	3	no	806	938	1894
E6	3	yes	437	728	1714

Abbreviations

Exp: Experiment
Env: The problem solving environment;
1) single vehicle track, no noise
2) single vehicle track, random noise
3) crossing vehicle tracks, random noise
Mech?: Whether the Inhibiting-goal and Local Context mechanisms are used.
KS ex: The number of knowledge source executions required to find the solution(s)
Hyps: Number or hyps generated during problem solving.
Goals: Number of goals generated during problem solving.

to acoustically sensed data in an attempt to identify, locate and track patterns of vehicles moving through a two-dimensional space. A node is responsible for a specific area and attempts to recognize and eliminate errorful sensor data as it integrates the correct data into an answer map. Each node has a blackboard architecture with knowledge sources and blackboard levels of abstraction appropriate for vehicle monitoring. Knowledge sources perform the basic problem solving tasks of extending and refining hypotheses (partial solutions). As described earlier, data-directed and goal-directed goals are used to control problem solving activities.

We have implemented the mechanisms described in this paper and carried out experiments to test their effectiveness in a single node system. The results are summarized in Table 1. Three environments were used for testing; a simple environment with a single vehicle track and no sensor noise, an environment with the same vehicle track but with random noise added, and a complex environment with two crossing vehicle tracks and a significant amount of noise. The features used for comparison

were number of knowledge source executions required to produce the solution(s), number of hypotheses created, and number of goals created. As shown in Table 1, the system performed more efficiently with the new mechanisms. For the environment with a single track, 36% fewer hypotheses and 9% fewer goals were produced and the system required 65% fewer KS executions to compute the answer. In the second environment, 11% fewer hypotheses and 8% more goals were produced and the solution was found with 37% fewer KS executions. Finally, in the complex environment, 22% fewer hypotheses and 10% fewer goals were produced and the solutions were found with 46% fewer KS executions.

In each of the environments, the new mechanisms were effective in preventing redundant processing in areas where strongly believed, high-level results were found. This enabled the system to allocate resources for work in noisy areas and areas where the sensed signals were weak. Although the new mechanisms caused the system to generate additional goals, most noticeably in environment 2, the resulting improvement in focusing capabilities resulted in a considerable reduction in the number of knowledge sources executed.

6 Conclusion

We have shown that mechanisms for accurately controlling the flexibility provided by the multi-level, cooperative knowledge source model of problem solving can be built as natural extensions to the integrated data-directed and goal-directed architecture. We have introduced a new type of goal, the inhibiting-goal, and we have presented a taxonomy of generic goal relationships. These mechanisms have applicability to tasks in which combined data-directed and goal-directed control is appropriate and where it is possible to roughly predict the quality and characteristics of a KSI's output. Performance results were given demonstrating the effectiveness of this approach to control. Our future research plans include experimenting in more complex environments in order to gain insights into further uses of these mechanisms and the use of

these mechanisms for real time control.

7 Acknowledgements

We would like to thank Jasmina Pavlin for her thoughtful criticism of earlier drafts of this paper and Dave Hildum for assisting with the implementation.

References

[1] Victor R. Lesser and Lee D. Erman.
A retrospective view of the Hearsay-II architecture.
In *Proceedings of the Fifth International Joint Conference on Artificial Intelligence*, pages 790–800, Tiblisi, Georgia, USSR, August 1977.

[2] H. Penny Nii.
Blackboard systems: The blackboard model of problem solving and the evolution of blackboard architectures.
AI Magazine, 7(2):38–53, Summer 1986.

[3] H. Penny Nii.
Blackboard application systems: Blackboard systems from a knowledge engineering perspective.
AI Magazine, 7(3):82–106, Conference 1986.

[4] Lee D. Erman, Frederick Hayes-Roth, Victor R. Lesser, and D. Raj Reddy.
The Hearsay-II speech-understanding system: Integrating knowledge to resolve uncertainty.
Computing Surveys, 12(2):213–253, June 1980.

[5] Daniel D. Corkill, Victor R. Lesser, and Eva Hudlická.
Unifying data-directed and goal-directed control: An example and experiments.
In *Proceedings of the National Conference on Artificial Intelligence*, pages 143–147, Pittsburgh, Pennsylvania, August 1982.

[6] Frederick Hayes-Roth and Victor R. Lesser.

Focus of attention in the Hearsay-II system.

In *Proceedings of the Fifth International Joint Conference on Artificial Intelligence*, pages 27–35, Tiblisi, Georgia, USSR, August 1977.

[7] M. Vaughn Johnson and Barbara Hayes-Roth.

Integrating diverse reasoning methods in the bb1 blackboard control architecture.

In *Proceedings of the National Conference on Artificial Intelligence*, pages 30–35, Seattle, Washington, July 1987.

[8] Edmund H. Durfee and Victor R. Lesser.

Incremental planning to control a blackboard-based problem solver.

In *Proceedings of the National Conference on Artificial Intelligence*, pages 58–64, Philadelphia, Pennsylvania, August 1986.

[9] Edmund H. Durfee and Victor R. Lesser.

Incremental planning to control a time-constrained, blackboard-based problem solver.

IEEE Transactions on Aerospace and Electronics Systems, September 1988.

[10] Barbara Hayes-Roth.

A blackboard architecture for control.

Artificial Intelligence, 26(3):251–321, July 1985.

[11] Victor R. Lesser and Daniel D. Corkill.

The Distributed Vehicle Monitoring Testbed: A tool for investigating distributed problem solving networks.

AI Magazine, 4(3):15–33, Fall 1983.

(Also to appear in *Blackboard Systems*, Robert S. Engelmore and Anthony Morgan, editors, Addison-Wesley, in press, 1988 and in *Readings from AI Magazine 1980–1985*, in press, 1988).

[12] Victor R. Lesser, Daniel D. Corkill, and Edmund H. Durfee.

An Update on the Distributed Vehicle Monitoring Testbed.
Technical Report 87-111, Department of Computer and Information Science, University of Massachusetts, Amherst, Massachusetts 01003, December 1987.