A Framework for The Blackboard Model

Magnus Kempe

The *blackboard model* is a flexible problem-solving approach where the data and knowledge rules are strictly separated. The data generated during a problem resolution, from a problem statement to a solution through various stages of inferences, is centrally stored in many forms of objects. The knowledge rules are divided into independent experts; each expert contributes to the resolution with its knowledge of how to search for a (partial) solution of (parts of) the problem.

A *blackboard system* is composed of three main components: the blackboard, which is a database containing data and hypotheses; knowledge sources, which examine the state of the blackboard and create new data or hypotheses; and a control mechanism, which regulates the incremental hypothesizing and testing.

The blackboard model was first proposed by Newell in 1962, as a description of cooperative goal-directed action; the cooperative behavior is mediated by the state of the resolution on the blackboard. It is an opportunistic, as opposed to systematic, variant of rule-based expert systems. The blackboard model is best viewed as "knowledge assembly," not as "search for solutions" (contrary to a common view of AI.)

The blackboard model of problem solving became famous with the Hearsay speech understanding systems [Lesser75, Lesser77, Erman 80, Nii86]. A *blackboard application* is a particular problem-solving system developed on top of a *blackboard framework*. A *blackboard framework* is an architecture, or skeleton, embodying a *blackboard model*.

The blackboard model describes systems which support some difficult problem-solving techniques:

- *Dynamic control*: incremental formation of the solution is the norm; at each step, new rules may be triggered.
- *Focus of attention*: there is no rigidity in the direction of resolution; work proceeds at various levels of abstraction, interleaving steps of forward and backward chaining, and goals can vary over time.
- *Flexibility of control*: the resolution procedure can be codified in simple, independent control rules, or in complex control regimes ("the right knowledge can be applied at the right time").
- *Island driving*: the entire current best hypothesis is visible to the knowledge sources; promising parts of the resolution space can be treated as islands, with the additional goal of building bridges between such islands.
- *Concurrent resolution*: the use of multiple knowledge sources is conducive to parallelization of the code and the pursuit of multiple lines of reasoning (alternative solution paths).

Architecture of the Framework

Blackboard

The *blackboard system* (BB) contains a collection of *panel objects* (POs); this collection constitutes the domain of problem-solving, which we call the *domain resolution*. The BB also contains a collection of *knowledge sources* (KSs), which we call the *domain knowledge*. The panels objects are constantly modified and used by the knowledge sources, under the control of an *inference engine* and the guidance of a *strategy*.

The initial configuration of the BB is: the application's domain knowledge (the KSs), a strategy for the activation of KSs, and the problem statement embodied by the initial state of the domain resolution (the POs).

The general control loop of the BB is such that no polling of the KSs is necessary for the resolution. Each KS is self-activated: it notifies the BB when it has something to contribute to the resolution. Figuratively, KSs raise their hand, then the most promising KS is called on.

Domain Resolution

The domain resolution is the collection of POs. The number and purpose of POs varies with particular applications of the BB. The domain resolution is partitioned in domain levels (e.g. transistor, gate, slice, and device; or: sounds, phonemes, words, and sentences); each level corresponds to a *panel type*.

Such partitions are necessary to avoid turning each KS into a mini-expert system; ideally, a KS restricts its activities to one or two panel types. Panel types are also necessary to structure the levels of hypotheses and to control the amount of data on the blackboard [Carver 92].

Panel Object

A panel object has a name which serves as a unique identifier; a set of interested KSs, to notify them in case the PO is modified; and a panel type (domain level) which determines which KSs may be interested in its modifications.

In addition, a panel object will have items that vary according to application: a collection of named values (varies in number, structure, and names according to type of panel—i.e. record components in Ada).

Panel Type

A panel type has a set of interested KSs to notify in case one of its POs is modified. Usually, each panel type will correspond to one subclass of POs.

Domain Knowledge

The domain knowledge is the collection of KSs. The number and purpose of KSs varies with particular applications of the BB.

A very special KS is the strategy KS, which in effect regulates the activities of the other KSs. Since it is so special, we single it out as one particular object (the strategy, described later).

Knowledge Source

A knowledge source is an expert of some specific part of the domain. Relative to the strategy, a KS is a specialist (or a tactician); it doesn't see the big picture. The domain knowledge needed to solve a problem is partitioned into knowledge sources that are kept separate and independent.

A KS takes the current state of the domain resolution and updates it as encoded in its specialized knowledge. Only KSs are allowed to change the domain resolution.

The KSs are represented as procedures, sets of rules, or logic assertions. Very broadly speaking, KSs normally embody two elements: a local inference engine, and a knowledge base. The activity of a KS is defined by preconditions (what triggers its interest), and actions (processing its rules or procedural knowledge when triggered).

A KS has a set of interesting panel types and a set of interesting POs.

A KS can: register interest in a panel type; register interest in a specific PO; accept to be triggered on a PO creation/modification/destruction; and update the domain resolution (i.e. the POs)—if given permission by the strategy.

Each KS uses POs at one level (of analysis, its input) and then generates and/or modifies POs at another level (its output). (These levels may correspond to levels of abstractions, e.g. sounds, phonemes, words, sentences, syllogisms...)

A KS may work by making a hypothesis and then verifying it (backward-chaining) or by going from specific assertions to general assertions (forward-chaining). The blackboard model's incorporation of both kinds of KSs is what makes it opportunistic.

Stimulus

A stimulus records changes in the domain resolution. It is used to indicate to interested KSs that a PO has been created, changed, or deleted.

Response

When a KS requests to be triggered, it creates a response to the stimulus it has received. A response has the KS as creator and the stimulus as cause.

Inference Engine

The pilot of the framework is the inference engine, a control mechanism. It mediates the interactions between domain resolution and domain knowledge (registering potential triggers, by sending out stimuli, and storing the corresponding responses). It submits the responses to the strategy, triggers one KS picked by the strategy, and finally prepares the next batch of triggers by logging the modifications of POs performed by the triggered KS.

The inference engine implements the following problem resolution cycle:

- a KS modifies, adds, or deletes a PO in the DR; this PO has a panel type and a set of interested KSs;
- interested KSs are notified of the change (the stimulus) and may ask to be triggered (by "raising their hands", with a response);
- each response is transmitted to the strategy;
- the strategy ranks or ignores the responses of KSs with "raised hands", thus

updating an agenda;
- the strategy picks a best response in the agenda;
- the strategical KS is triggered;
- the process ends when no more KSs are triggered (a solution was found, or none could be found).

Strategy

The strategy (some sort of super KS) is a heuristic for priority ordering to set up the sequence of KSs which will be triggered next (on the basis of their responses). The strategy is a kind of co-pilot, which serves to rank the most interesting directions of exploration. The strategy also determines the permissions of KSs to be triggered (censorship).

The strategy simply determines the order in which, and whether , interested KSs are triggered. This is done on the basis of an agenda of responses from interested KSs, as received from the inference engine. The strategy determines whether a solution has been reached (or whether to stop searching).

The communication between KSs, inference engine, and strategy follows a pattern of stimulus-response interactions. A stimulus describes a changed PO; it is sent to interested KSs. Interested KSs may return a response which indicates that they are trigger-happy and which records some strategical information about their interest. The strategy uses the responses to rank the trigger-happy KSs, thus making an agenda. The information contained in responses may vary from application to application (according to the nature of KSs).

Bibliography

[BB 88] *Blackboard Systems*. Ed. by R. Engelmore and T. Morgan. Addison-Wesley, 1988.

[BB 89] *Blackboard Architectures and Applications*, Perspectives in Artificial Intelligence (vol. 3). Ed. by V. Jagannathan, R. Dodhiawala, and L.S. Baum. Academic Press, Inc., 1989.

[Booch 91] G. Booch. *Object-Oriented Analysis and Design*. 2nd ed., pp. 413-444.

[Buttner 89] K. Buttner, D. Sriram, and M. Freiling. *An Object-Oriented Blackboard for Model-Based Diagnostic Reasoning*, in [BB 89], pp. 403-431.

[Carver 92] N. Carver and V. Lesser. *The Evolution of Blackboard Control Architectures*. Technical Report UM-CS-1992-071, University of Massachusetts Amherst. October 1992. URL ftp://ftp.cs.umass.edu/pub/techrept/techreport/1992/UM-CS-1992-071.ps

[Erman 80] L. Erman, F. Hayes-Roth, V. Lesser, and D.R. Reddy. *The Hearsay-II Speech-Understanding System: Integrating Knowledge to Resolve Uncertainty*, in *Computer Surveys* 12(2):213-253, 1980. Also in [BB 88], pp. 31-86.

[Lesser 75] V. Lesser, R. Fennell, L. Erman, and D.R. Reddy. *Organization of the Hearsay-II Speech Understanding System*, in *IEEE Transactions on Acoustics, Speech, and Signal Processing*, ASSP-23, 1975, pp. 11-24.

[Lesser 77] V. Lesser and L. Erman. *A Retrospective View of the Hearsay-II Architecture*, in *Proceedings of IJCAI-77*, 1977, pp. 790-800. Also in [BB 88], pp. 87-121.

[Nii 86] H.P. Nii. *The Blackboard Model of Problem Solving*, in *The AI Magazine* 7(2):38-53 and 7(3):82-106, 1986.