Chapter 11

Blackboard systems

Although many of the expert and knowledge-based systems which make use of some kind of reasoning maintenance are hand-rolled and heavily dependent on features of the domain for their design, there is one general framework that often embodies something like an RMS. In the early 70s an approach was worked out to handle those domains in which several relatively incompatible kinds of expertise were needed to be able to solve problems. The general framework is called a "blackboard system"; the metaphor is of a variety of experts co-operating by mapping out a solution on a blackboard, each expert contributing when he sees that the state of the blackboard is such that he can do something useful. The experts might contribute a step in the solution, or contribute an expectation – something that he (perhaps incorrectly) predicts given the current state of the problem, or something that he needs before he can again be useful.

11.1 The basic ideas

The prototypical blackboard system has three main components:

- *the blackboard* itself, a multi-dimensional data structure, made up of *entries*. Typically, an entry corresponds to a hypothesis or expectation about some aspect of the problem. Typically, one of the dimensions of the blackboard is 'representional level', running from the lowest, nitty-gritty level to the highest, most abstracted level; another of the dimensions represents the number of competing hypotheses about some aspect of the problem. Usually, the entries have a lot of associated dependency information, to show which hypotheses or expectations depend on which others. Changes in one thus lead to changes in those depending on it, just as in an RMS. There is often a lot of control information recorded on the blackboard too - sometimes just because it is a convenient place to put it rather than for any more principled reason.

- a number of *knowledge sources*, alias KSs. These are essentially kinds of stripped-down expert system (e.g. without any explanatory facilities), often expressed in the form of a set of production rules. Each KS has two major components:
  - a *precondition part*, specifying the conditions for which it is likely to be worth 'waking up' this KS. Typically, this specifies that certain new entries should have appeared on the blackboard, of certain types, or at a certain level; it also gives very loose estimates of such details as the amount of resources that the KS might consume were it to be invoked, and the likely benefit, e.g. the number of hypotheses it might add to the blackboard and how reliable they might be in very general terms. These are normally based on subjective assessments by the system builder, rather than the fruits of reasoning by the system itself.
  - an *action part*, which does the work hinted at by the precondition part when it gets the chance.
- a *scheduler*, which drives the whole thing. Typically, it uses a straightforward algorithm such as this:
  - see what new entries have been made on the blackboard recently (at least one KS has to start the activity by being externally triggered, of course)
  - find out which KSs might be interested by any of these changes. Rather than expensively polling every KS, this is usually found by consulting control information either hand-crafted by the system builder or generated early on by the scheduler itself as a result of inspecting all the KSs preconditions
  - construct an agenda of *knowledge source activation records* (KSARs) usually consisting of instances of the preconditions of all those KSs it might be worth invoking
  - order the agenda according to some algorithm, to decide which KS to invoke
  - invoke it
  - make any resulting changes to the blackboard
  - ...and so on ...

In some blackboard systems, there are KSs whose job is to control the activation of other KSs, e.g. to say how many hypotheses some other KS may generate at each invocation. For example, in some text-handling system, there might be a KS whose job is, once a mis-spelt word has been found, to generate hypotheses

about what the correct spelling was. Given a badly mis-spelt word, there might be many hypotheses possible about the right word. If the word is vital for comprehension, it is worth considering many, otherwise just a few. It might be the job of another KS to set the limit, by judging on other criteria how important it was to find the intended word.

A blackboard system could be thought of as a generalisation of a message-passing system (and thus less efficient, probably). The blackboard entries would be messages, which could be looked at by any KS (or in message-passing terms, potential recipient) but read by only a few. A blackboard entry is in effect an anonymous message; unlike most message-passing systems, there can be many recipients, or none.

11.2 HEARSAY-II

The first successful blackboard system was the HEARSAY-II system (see papers by Reddy, Erman, Lesser, McCue, all from CMU). Its function was to understand speech. It was created as a result of the US Department of Defense Advance Research Project Agency (ARPA) setting goals in 1971 for the creation of a speech understanding system. The aim set by ARPA was to create a system that should:

- accept connected speech
- from many co-operative speakers of the general American dialect (whatever that is)
- in a quiet room
- using a good quality microphone
- with slight tuning per speaker
- needing only natural adaptation by the speaker
- with a slightly artificial vocabulary of 1000 words
- with highly artificial syntax and highly constrained task
- providing graceful interaction (rather than truculent!)
- tolerating less than 10% semantic error
- in a few times real time on a 100 million instructions per second machine
- by the end of 1976

In September 1976, HEARSAY-II could perform as follows:

speakers:	1
environment:	computer terminal room (¿65db)
microphone:	medium-quality
speaker tuning:	20-30 training examples
task:	consulting an AI document database
vocabulary:	1011 words, with no post-selection
language constraints:	context-free semantic grammar, static branching factor of 10
test data:	23 "blind" utterances, 7 words long on average, average 2.6 seconds long, average fanout of 40
accuracy:	9% sentence semantic error, 19% sentence error
resources:	60 million instructions per second of speech on DEC-10

Figure 11.1 below shows the levels of the HEARSAY-II blackboard, and the names of the KSs of configuartion C2 of the system. The arrows show which levels each KS is concerned with: for example, WORD-SEQ examines hypothese at the word level, and creates or modifies hypotheses at the word-sequence level. The dimensions of the blackboard are level, time since the start of the utterance, and number of hypotheses at that level concerned with essentially the same slice of the utterance (i.e. competing hypotheses).

What happens, in broad terms, is that an utterance is low-pass filtered and sampled at 10kHz, and a simple classification is made of the features of the input signal to extract the 'parameter level' information. This happens in real time. Then SEG, triggered by the appearance of this information on the blackboard, creates segment hypotheses. WORD-CTL, triggered by the start of processing, creates a hypothesis to control the amount of hypothesising that MOW will eventually do. It may be triggered again later if processing seems to be getting moribund without reaching any of the higher levels. Likewise, WORD-SEQ-CTRL controls the amount that WORD-SEQ will do. POM is triggered by the apearance of segment hypotheses, and creates syllable-class hypotheses. MOW, triggered by these, creates as many word hypotheses as WORD-CTL allows it to. At this point, the real blackboard activity starts. In an earlier version of HEARSAY-II, the real activity, with all KSs competing for the scheduler's attention, started with the creation of 'parameter level' information, but this proved to be very inefficient. Instead, the activation of KSs is strictly bottom-up so as to get a useful collection of initial hypotheses at the word level; only then do KSs get the chance to compete.

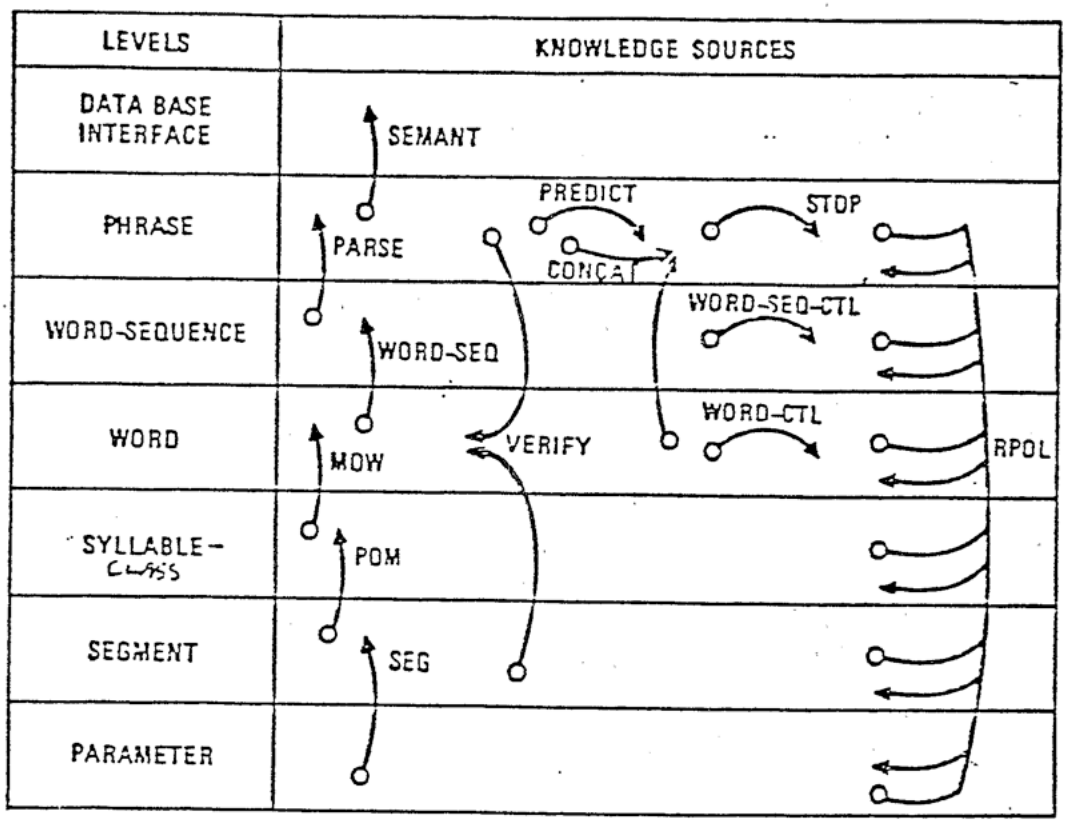

Figure 11.1: Levels and KSs of configuration C2 of HEARSAY-II (Erman/Lesser)

The other main KSs are as follows:

- WORD-SEQ generates a small set of word-sequence hypotheses, based on a 1011 × 1011 matrix of word-pair possibilities (e.g. "a" is never followed by "the" in this domain) and acoustic/phonetic information about word junctures.

- PARSE does parsing to generate phrase hypotheses, given word-sequence hypotheses.

- PREDICT looks at phrase hypotheses and predicts preceeding and following phrases. VERIFY verifies words in phrases: they must pass a word-juncture test. Although, in a prototypical blackboard system, the KSs are meant to be wholly independent, in HEARSAY-II VERIFY can check MOW's internal database to see if there are any as yet unhypothesised words that might fit. VERIFY also looks at the segment net to see if its word predictions are plausible.

- CONCAT marries phrases to create longer ones.

- STOP halts the competition between KSs when either a sufficiently credible phrase spanning the whole utterance has been hypothesised, or when processing has become moribund.

- RPOL is run after every other KS runs. It gives a credibility rating to newly created or modified hypotheses, derived from ratings of the stimuli that triggered their creation or modification. Every hypothesis has attached dependency information which shows the other hypotheses that it depended on, so that a change in rating of one hypothesis causes changes in the ratings of all those that depend on it.

- SEMANT is run, once, after STOP has run. Either it finds the complete phrase or it attempts to form one, given the possible phrase hypotheses.

- A final KS, DISCO, takes the winning phrase, turns it into a database query and gives the answer back to the user.

Figure 11.2 shows the segment-level hypotheses (at (c)) and syllable-class hypotheses (at (d)) after HEARSAY-II has successfully recognized the utterance "Are any by Feigenbaum and Feldman?". Levels (a) and (b) show the signal from the microphone, and the utterance itself, purely for reference purposes.

Figure 11.3 shows the hypotheses at the word level (e+f), word-sequence level (g) and phrase level (h). For each entry, its box shows the interval of the utterance about which it is a hypotheses. Within the box are details, e.g. 14: WEIZENBAUM 70 showing that this was a hypothesised word created at

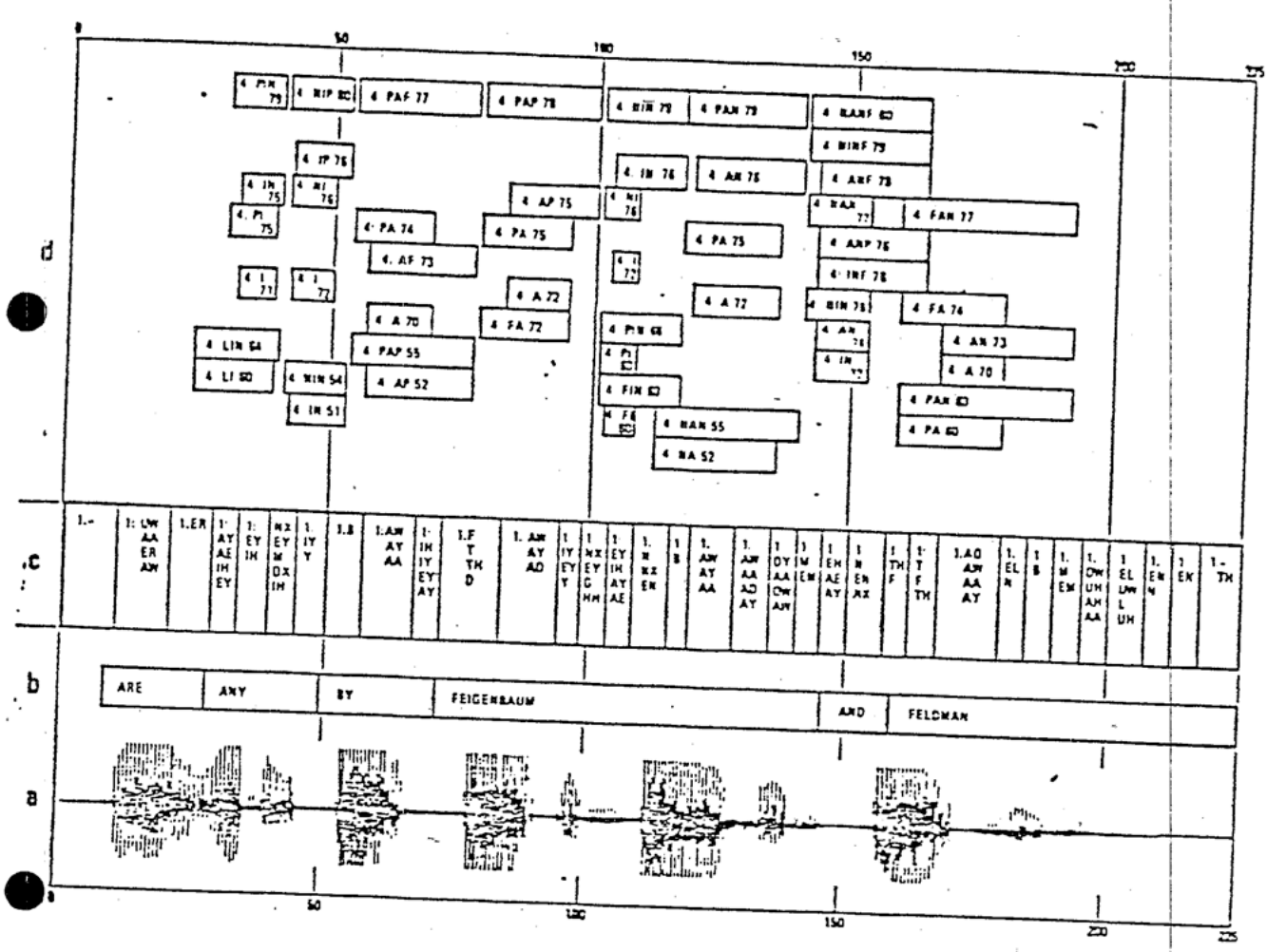

Figure 11.2: The lower levels of the blackboard at the end

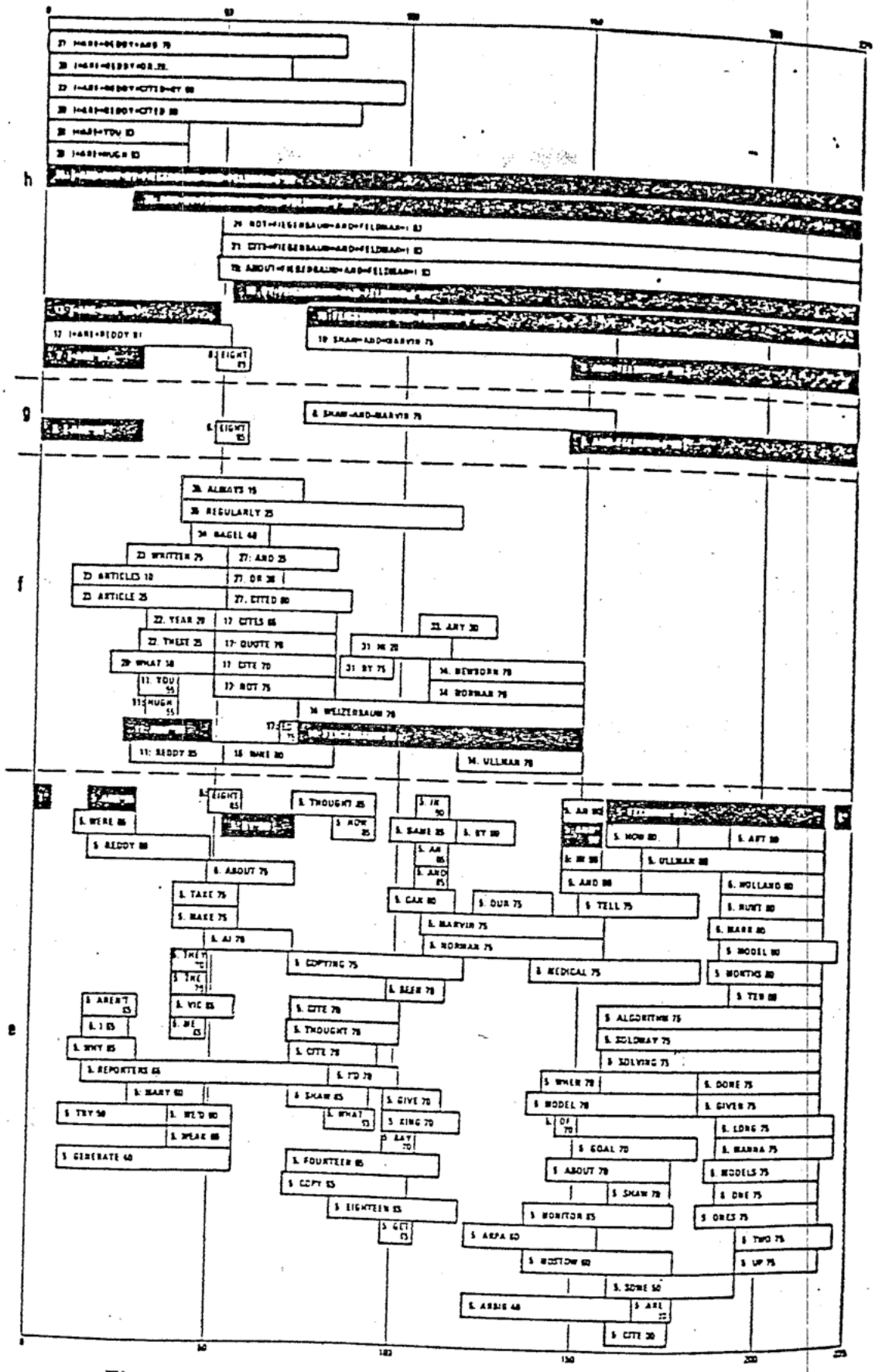

Figure 11.3: The higher levels of the blackboard at the end

the 14th cycle of the scheduler, and its credibility rating was 70, on a scale of 0-100. The black boxes are the correct hypotheses; the majority are wrong.

11.3 HASP/SIAP

HASP was a blackboard system developed at Stanford University in the mid-70s, under a military contract. It was formerly called SU/X. In due course its development was taken over by a company called Systems Control Technology Inc who had been involved with it from the start; it was then renamed SIAP. See papers by Nii and Feigenbaum, for example in AI Magazine, Spring 82.

HASP's job was the processing of signals from arrays of hydrophones and intelligence information from various sources in order to build and maintain a good conceptual picture of the sea traffic in a certain part of the ocean. To understand the nature of the problem, consider an analogous task: listening to the signals coming from an eavesdropping device at a cocktail party. From the babble of voices it is desriable to sort out individual conversations; the most interesting ones are usually the quietest ones.

Hydrophone arrays have some directional resolution: they 'look at' a widening channel along the axis of the array. The sounds of interest come from ships' propellor shafts, from the propellors themselves, and from various kinds of machinery on board each ship. Each source radiates sound on certain fundamental frequencies and certain harmonics (multiples) of those frequencies. The many sources thus combine to produce a sonogram, showing sound frequencies and energies over time arriving at a hydrophone array. It is a very highly skilled job to interpret these sonograms. What complicates the task is that factors such as salinity gradients or temeprature gradients can affect the apparent direction and strength of sounds, in ways that depend on frequencies. It is also possible for all the sound from a source to fade out completely for some while (e.g. 10 seconds to 30 minutes), or for certain of its characteristic fundamental frequencies and/or harmonics to fade out for a while.

The raw data consists of sets of sonogram lines (akin to spectral lines, but showing frequency against time, with strength of signal represerted as intensity on a visual display of the sonogram). HASP's job is to construct a 'Current Best View' of part of the ocean, and maintain it as time passes. Figure 11.4 (all diagrams are from the paper mentioned above) shows the general structure.

Figure 11.5 shows the various levels of the system's blackboard, and a few of the KSs. Entries on the blackboard consist of hypotheses about what is or was or is likely to be true at some point in time at that level. For instance, at the Vessel level a hypothesis might indicate the presence of a certain vessel, with a given class, location, speed, course and destination. Each element of such an entry has a separate credibility rating. Any entry may be produced by abstraction

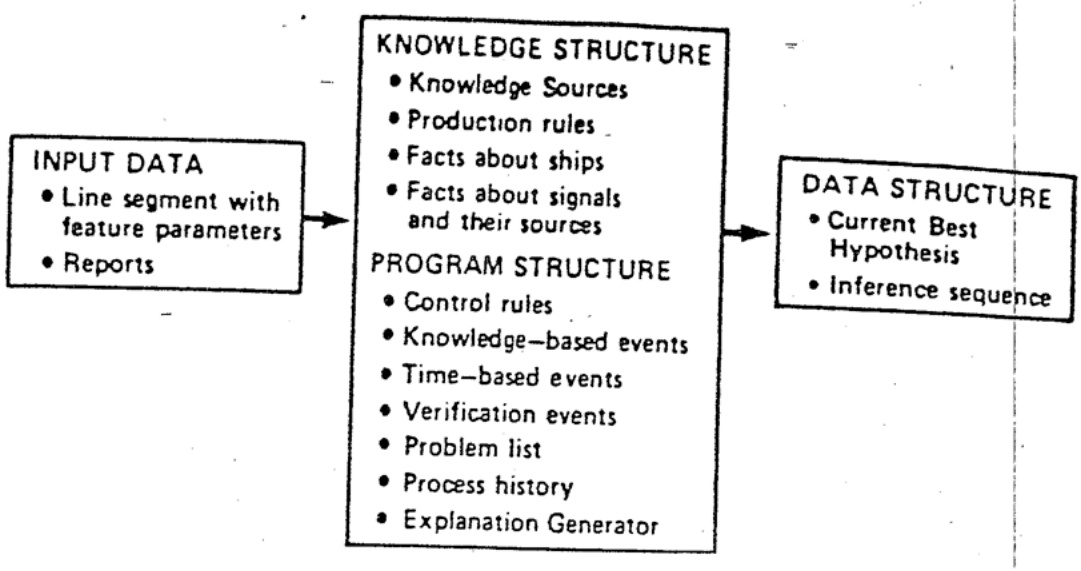

Figure 11.4: The general structure of HASP/SIAP

from lower level hypotheses or may be produced by expectation from higher level hypotheses.

Figure 11.6 gives a general outline of the system components; CBH is the 'current best top-level hypothesis about what is happening', and consists of a conjunction of hypotheses about vessels.

The system operates cyclically: new data goes onto the Event List once it has been electronically processed and turned into low-level symbolic data. Each cycle runs like this:

- the Clock-events list is checked to see if there is anything outstanding to do at this cycle, e.g. check for confirmation of a source heard earlier which had faded out and then may have reappeared. If there is anything to do, the appropriate KS is called.

- the Expectation-driver is run to see if any events added to the Event-list can be used to account for anything on the Problems list of unresolved questions. If so, the appropriate KSs are run, and the Problem list suitably pruned.

- the Event-driver is run, to process all new events and changes to the CBH on the previous cycle, by calling the appropriate KSs. They change the CBH as necessary, and possibly add new events to the Event-list.

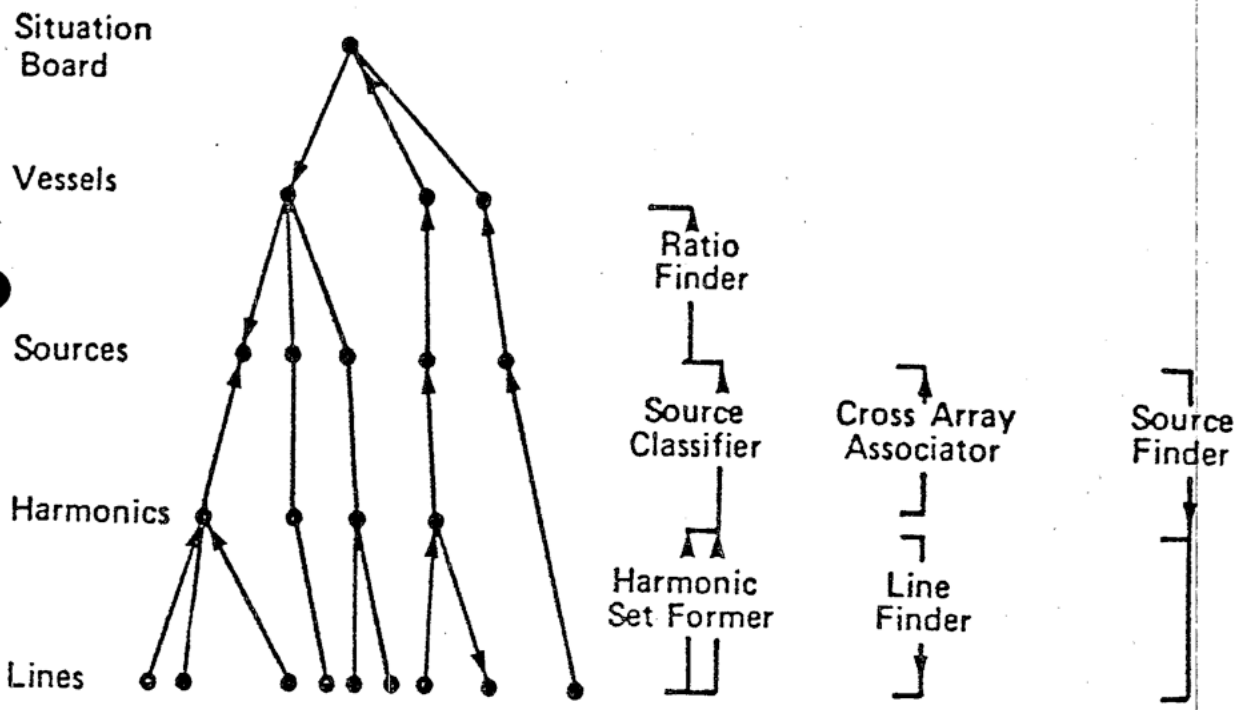

Figure 11.5: The levels of the HASP/SIAP blackboard

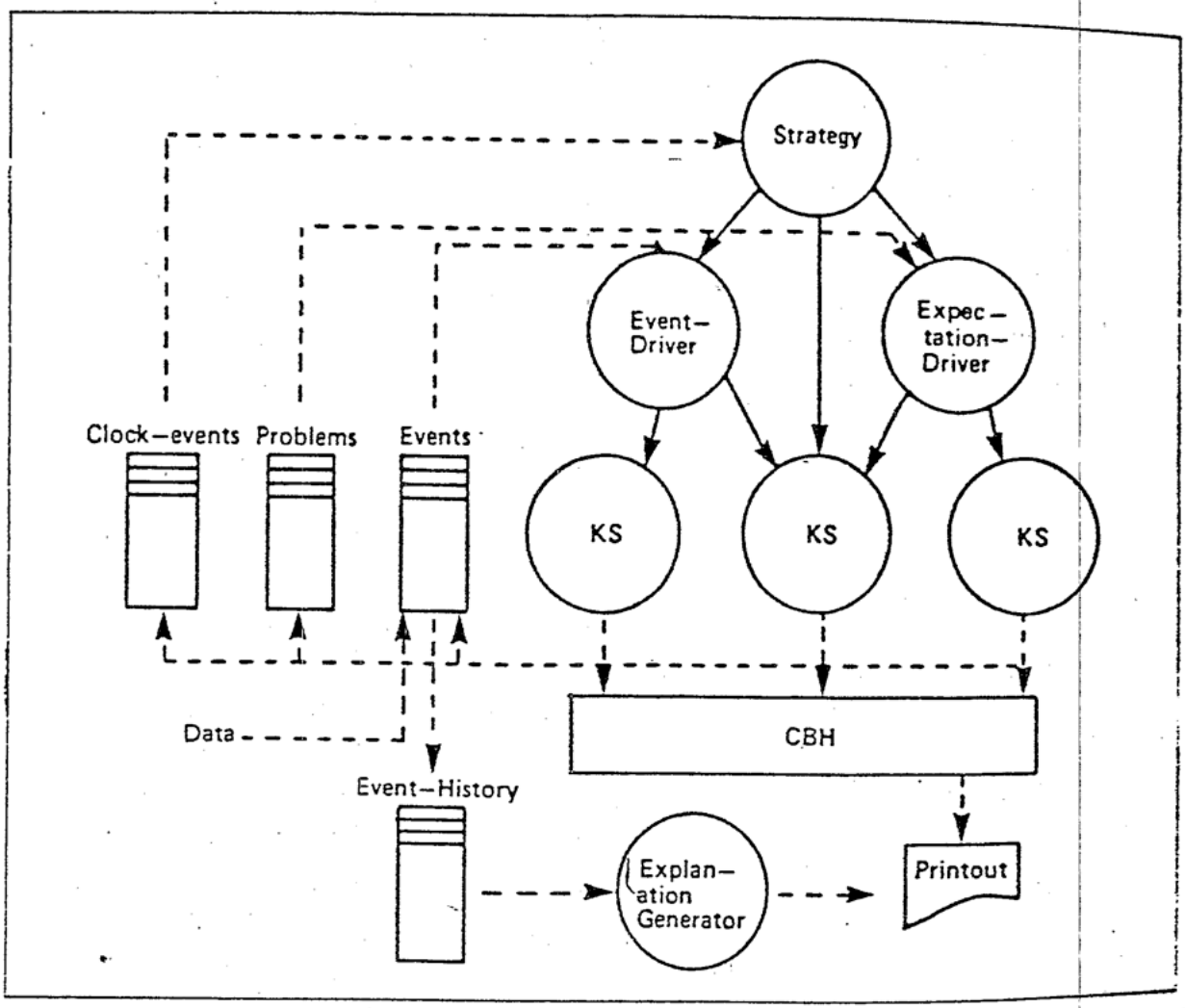

Figure 11.6: The components of HASP/SIAP

- the cycle ends when the Event-list is empty.

Though the system operates in real time, this is not so amazing as it seems; each sonogram represents a 5-minute picture, so a cycle is allowed to take minutes rather than seconds or less to complete. The system has been judged in field trials to have expert performance.

In Britain, a blackboard system shell called MXA has been developed by SPL, a part of Systems Designers International. It has been applied to a similar kind of task as HASP, namely the construction and maintenance of a current best view of what is happening in the airspace around a ship. It too operates in real time, on a VAX, but is perhaps more impressive since events can happen much faster. MXA is very unusual: it is doubly compiled. MXA itself takes a description of a blackboard system and all its KSs and compiles it to a Pascal program, which is then compiled to produce a very fast running product.

11.4 OPM

This is a blackboard system with a difference: the scheduler's control algorithm, by which the agenda gets sorted so as to decide which KS should be activated within a scheduler cycle, is itself determined by a set of KSs which reason about control issues. See papers by Barbara Hayes-Roth in Artificial Intelligence, 1983.

OPM's aim is to plan how to satisfy as many of a set of desirable tasks as possible, given resource constraints such as limited time which may imply that not all the tasks can be tackled. The example given by Barbara Hayes-Roth is:

> You have just finished a session at the Health Club. It is 11am. You have the rest of the day to acheive some of the following tasks, but you need to be at a garage at a certain place in the city by 5:30, to collect your car to drive home. You would like to see a film, at one of two cinemas; both have showings at 1pm and 3pm. Also, you ought to:
>
> - pick up some medicine from the vet's office
> - buy a new fan belt for your fridge
> - look at a few houses, since you're planning to move
> - meet a friend for lunch at some restaurant you choose
> - buy a toy for your dog
> - pick up your watch from the repairers
> - order a book from a certain bookshop
> - buy some fresh vegetables

- buy this month's issue of a gardening magazine
- go to a florist to get some flowers sent to a friend in hospital.

Humans do this kind of task in a sensible way. They cluster the activities by area of the city, and move from cluster to cluster rather than rushing back and forth across the city. When in a location, they see if they can achieve any tasks that aren't bound to one particular location, and make sensible trade-offs between the constraints (but you knew all that, didn't you?). The basic knowledge sources of OPM were designed after analysing a good number of verbal protocols given by volunteers who were set such tasks. For example, here are two of the KSs, in an ersatz formalism:

```
KS 452: DESIGN-REFINER

CONDITION:
        There is a design whose specification covers task B
        There is a new Procedure which specifies:
                Go from task A to task B
        There are Procedures that locate each unplanned task
          and measure its distance and direction from task B
ACTION:
        Identify the best unplanned task as the one whose
          direction from task B fits the design and which
          minimises the distance from B
        Create a Procedure that specifies:
                Go from task B to the best unplanned task

KS 324: CLUSTER-RECOGNIZER

CONDITION:
        There is a new Procedure locating task A
        There are Procedures locating at least two other
          tasks in the vicinity of A
ACTION:
        Create a Design that identifies a cluster of tasks,
          consisting of A and the other tasks in its vicinity
          and that identifies the region in which they lie.
```

This suggests, incidentally, that there were over 400 KSs. In fact, there were about 20 at most! The following shows an example agenda at cycle 31:

```
OPM Agenda: Cycle 31
```

```
((KSAR 103) (KS 452) (EVENT 40) (CYCLE 31) (PROCEDURE 095)
   (DESIGN 019) (TRIGGERWEIGHT .8) (LEVEL PROCEDURE)
   (INTERVAL 2) (KSEFFICIENCY .5) (KSCREDIBILITY .9)
   (CONTROL-OR-DOMAIN D) (KSIMPORTANCE .9))
((KSAR 102) (KS 324) (EVENT 40) (CYCLE 31) (PROCEDURE 095)
   (NEIGHBOURS ((PROCEDURE 088) (PROCEDURE 082)))
   (TRIGGERWEIGHT-.8) (LEVEL DESIGN) (INTERVAL (2 3))
   (KSEFFICIENCY .3) (KSCREDIBILITY .6)
   (CONTROL-OR-DOMAIN D) (KSIMPORTANCE .6))
((KSAR 101) (KS 249) (EVENT 35) (CYCLE 27) (PROCEDURE 076)
   (TRIGGERWEIGHT 1.0) (LEVEL PROCEDURE) (INTERVAL 4)
   (KSEFFICIENCY .9) (KSCREDIBILITY 1.0)
   (CONTROL-OR-DOMAIN D) (KSIMPORTANCE .9))
... and so on ...
```

The first item on the agenda is KSAR 103, an instance of KS 452 above, triggered by event 40 on cycle 31 (this one), and Procedure level entry 095 and Design level entry 019 of the developing plan are to be the B and A respectively of the invocation of KS 452. The system's cofidence in the triggering is 0.8 on a scale 0-1, it will generate a new entry at the Procedure level in the second solution interval, KS 452's efficiency is 0.5, it produces useful entries 90% of the time, it is a domain KS rather than a control KS and the entry it would produce would be very important.

The domain part of the blackboard has two important dimensions: the level, and the time interval for plan execution (in the above problem, it is 11am to 5:30pm). The levels are:

OUTCOME the highest level. Entries describe the specific problems to be handled, after massaging the general task such as the one above

DESIGN decisions about moving from cluster to cluster

PROCEDURE decisons about moving about within a cluster, e.g. go from florist to vet

OPERATION low level decisions, e.g. how to get from florist to vet ("go along X, turn right onto Y, then left at Z")

There is also a control part of the blackboard - you could happily think of it as a second blackboard. The important dimensions are scheduling cycles and level. The levels are:

PROBLEM general descriptions of the problem to be solved

STRATEGY general guidelines of how to tackle it, e.g. top-down

FOCUS restrictions to make on the system's attention, e.g. consider the DESIGN level of the domain blackboard for the moment

POLICY general decision criteria, e.g. (when there have already been a good number of cycles) consider KSs that are reliable first.

AGENDA an entry is an agenda at a specific scheduler cycle

KSAR the entry is the chosen KSAR indicating the KS to be triggered on that cycle.

There are a variety of control KSs, for each of the levels. Figure 11.7 shows an example of the control blackboard, with sample decisions (taken from the paper by B.Hayes-Roth):

The scheduler itself is represented as three KSs. This allows the system, and the system builder, a high degree of flexibility about how the system works. The three KSs are:

```
KS 150: AGENDA-UPDATER

CONDITION:
        There is a new change to the most recent Scheduled-KSAR,
          KSAR-n, indicating that it has executed
ACTION:
        Create a new Agenda = the most recent prior Agenda
        Remove KSAR-n from the new Agenda
        If the execution of KSAR-n produced changes on the
          blackboard,
          Then remove from the Agenda and existing KSARs whose
            triggering conditions are invalidated by the change
          And add to the Agenda a new KSAR for each knowledge
            source whose triggering conditions are satisfied
            as a result of the change

KS 151: SCHEDULER

CONDITION:
        There is a new Agenda, Agenda-n
ACTION:
        Choose a KSAR from Agenda-n by applying the evaluation
          function to operative Foci and Policies for all KSARs
        Create a new Scheduled KSAR to record the chosen KSAR
          and the factors that influenced the decision to
```

Figure 11.7: An example of the OPM control blackboard

```
            schedule it

KS 152: KSAR-INTERPRETER

CONDITION:
        There is a new Scheduled KSAR, KSAR-n
ACTION:
        Interpret and execute KSAR-n
        Change KSAR-n to record pointers to all the resulting
          changes on the blackboard
        Change KSAR-n to indicate that it has executed
```

The following is an example of one of the (other) control KSs:

```
KS 191: PROBLEM-REFINEMENT

CONDITION:
        There is a new Problem
        Its type is "multiple-task planning"
ACTION:
        If estimated (task-time + travel-time) is less than
          the plan execution time
          Then let the Problem goal be "all"
          Else let the Problem goal be "many"
            And let the Problem goal-criterion be "importance"
```

This kind of framework, with explicit control KSs, appears to be sufficiently powerful to allow relatively straightforward replication of HEARSAY-II and HASP.