

ATOME : a blackboard architecture with temporal and hypothetical reasoning

Hassan Laasri, Brigitte Maitre, Thierry Mondot, François Charpillet, Jean-Paul Haton

▶ **To cite this version:**

Hassan Laasri, Brigitte Maitre, Thierry Mondot, François Charpillet, Jean-Paul Haton. ATOME : a blackboard architecture with temporal and hypothetical reasoning. [Research Report] RR-0855, INRIA. 1988. inria-00077180

HAL Id: inria-00077180
https://inria.hal.science/inria-00077180

Submitted on 29 May 2006

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

INRIA

UNITÉ DE RECHERCHE
INRIA-LORRAINE

Institut National
de Recherche
en Informatique
et en Automatique

Domaine de Voluceau
Rocquencourt
BP105
78153 Le Chesnay Cédex
France
Tél.:(1) 39 63 55 11

Rapports de Recherche

N° 855

ATOME : A BLACKBOARD ARCHITECTURE WITH TEMPORAL AND HYPOTHETICAL REASONING

Hassan LAASRI
Brigitte MAITRE
Thierry MONDOT
François CHARPILLET
Jean Paul HATON

JUIN 1988

RR-0855

LAASRI Hassan, MAITRE Brigitte, MONDOT Thierry,
CHARPILLET François, and HATON Jean Paul

INRIA/CRIN-Nancy

SYCO Project
Blackboard Group

B. P. 239, 54506 Vandoeuvre-Les-Nancy Cedex

Title : ATOME : A Blackboard Architecture with Temporal and Hypothetical Reasoning[1].

Abstract:

To cope with high level AI-based applications as signal understanding, process control or decision making, an AI system must take into account various knowledge sources and reasoning schemes.

In this paper, we propose a blackboard-based architecture which achieves opportunism and efficiency while controlling multiple knowledge sources. In addition it integrates temporal and hypothetical reasoning to deal with applications evolving in time, and manipulating noisy or errorfull information.

Titre : ATOME : une architecture à base de blackboard intégrant le raisonnement temporel et hypothétique.

Résumé :

Pour développer des applications avancées d'IA dans des domaines tels que l'interprétation de signaux, la commande de processus industriels ou l'aide à la décision il est nécessaire de mettre en oeuvre des sources de connaissances et des schémas de raisonnements variés.

Dans ce papier nous proposons une architecture fondée sur le modèle du "blackboard" qui permet de faire coopérer plusieurs sources de connaissances avec des stratégies de résolution efficaces et opportunistes.

Cette architecture intègre en plus des mécanismes de raisonnement temporel et hypothétique, ce qui lui permet de s'adapter à des situations évolutives dans le temps et de pouvoir traiter des données bruitées ou erronnées.

[1]This report is an expanded version of a paper presented at the European Conference on Artificial Intelligence (ECAI), Munich, August 1988 [1].

PAPIER RÉCUPÉRÉ ET RECYCLÉ

1 Introduction

Advanced AI-systems require the integration of various and heterogeneous knowledge sources and multiple reasoning schemes such as temporal reasoning to resolve applications evolving in time, or hypothetical reasoning to maintain coherence and drawing plausible inferences from incomplete data in large knowledge bases.

In this paper, we present ATOME, a shell for building blackboard-based systems currently developed at CRIN. The second section of this document describes ATOME 's blackboard(s), domain and control knowledge sources representations and functionalities. Section 3 describes temporal functionalities integrated in ATOME while section 4 points out the hypothetical and multiple hypothesis handling reasoning schemes under development. Section 5 gives an overview of current applications being developed with ATOME and its further developments.

2 The Architecture of ATOME

2.1 The Blackboard Model

A blackboard-based architecture consists of a number of individual computation agents, known as *knowledge sources* or KSs, which communicate with each other through a shared global database, known as the *blackboard* containing all necessary domain information called solution or hypothesis elements. A *controller* mediate the execution of enabled KSs.

KSs have a *precondition-action* format. The precondition describes situations in which the KS can contribute to the problem-solving process while the action specifies the KS's behavior. Only those KSs whose preconditions are satisfied can actually perform their actions.

The blackboard model corresponds to a high level description, and says nothing about how it can be computationaly implemented. As a result, blackboard-based systems differ widely, and the label *blackboard architecture* applies to diverse range of AI-based systems. For instance, in HEARSAY-II [11]. DVMT [8,9,25], HCVM [12] and OPM [15] KSs are procedures while in HASP/SIAP [32,31] KSs are rules bases, BB-1 [13] uses an agenda-based controller n while CRYSALIS [10,34], ATOME [14,18,20,19,28] use a hierarchical control and HEARSAY-II uses a sophisticated scheduler. For more details on blackboard-based architectures see [15,30,29,20].

2.2 The Specialist KSs

The ATOME 's specialists are the *domain* KSs which infer, modify or delete hypothesis elements on the blackboard(s) during the problem-solving process. A specialist in ATOME is a local expert for some aspect of the problem. It contributes to the overall solution by matching contextual information maintained on the blackboard(s).

The precondition of a specialist is a state-based predicate required for execution. The action of a specialist can be either a set of production rules (in this case, the interpreter is given by ATOME) or a program written in various languages (Lisp, C,...) as well as an expert system. At that time, a specialist could dispose of a local working memory on which it could consult and/or post its specific information, and when interactions with the other KSs is necessary, the specialist will read and/or write data in the global blackboard(s).

2.3 The Blackboard(s)

An AI system developed with ATOME (i.e. an ATOME-system) may have multiple blackboards to organize its domain.

Each level in a blackboard is represented as an object class and is characterized by a set of attributes and links. An hypothesis element corresponds to an instance of one of these classes. An hypothesis element attribute can have various alternative values resulting from different specialist KSs actions. Each value is weighted by a certainty coefficient, i.e. a number or the symbol *undefined* and eventually a validation period of time which indicates the portions of time during which the value is valid. At present certainty coefficients are not manipulated by the control component. The corresponding mechanisms will be added in a second step. Links between two nodes of the same blackboard or of different blackboards are also represented as attributes. A solution element link contains the nodes with which it is linked, and have automatically a reverse link.

When the specialist KSs process the objects on a level of a blackboard, they generate *events* that correspond to the creation of a new hypothesis element, or to the alteration or the deletion of the values of one or several attributes of an existing hypothesis. They can also represent the realization, the alteration or the deletion of links between solution elements. Events are placed on the top, on the queue or at one specific rank in the event-lists (cf. 2.4.1) depending of their importance.

2.4 The Control Component

The control in ATOME is carried out by *control* KSs which work as meta-knowledge sources. Unlike BB-1, ATOME 's control is hierarchical and is realized at two levels : *task* and *strategy* levels (cf. Figure 1).

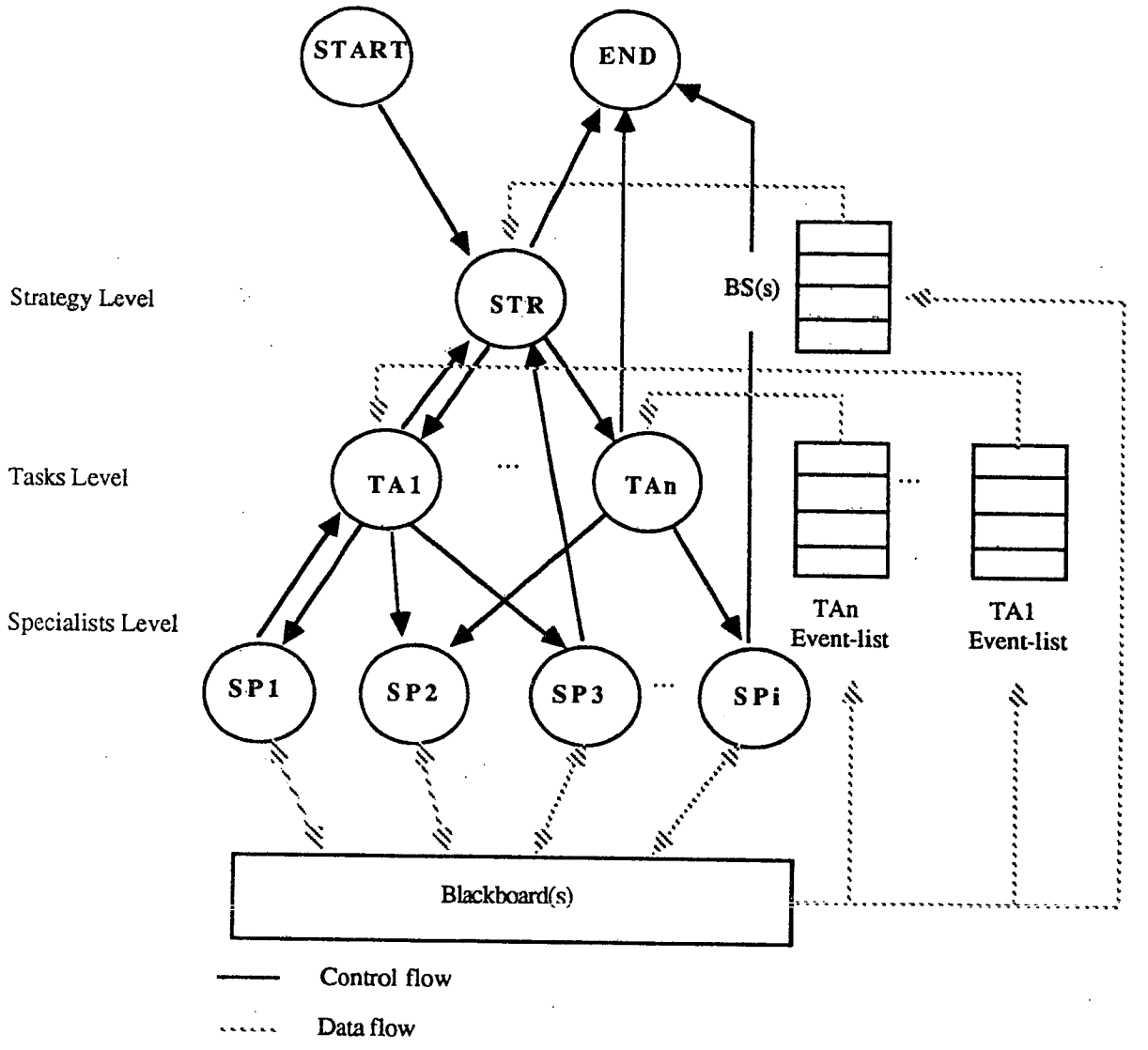


Figure 1 : ATOME's Architecture (without the agendas).
The strategy KS examines the BS(s) to select one or a set of tasks to execute sequentially. The current task scans its local event-list to activate one or more specialists sequentially or opportunistically depending of its type.
The current specialist consults and modifies the blackboard(s). As a result, they create events which will be placed onto the event-lists of tasks interested by these changes. The BS(s) is(are) then updated.

2.4.1 Tasks Control Level

A task is a control knowledge source which contains information on the specialist KS(s) capacities to solve a particular sub-goal. When activated, this control KS will decide which specialist KSs to activate during the problem-solving process. One specialist can be under the control of multiple tasks.

A task is composed of production rules in which the left hand side tests the existence of particular events and the state of global variables or task bindings (i.e. the left hand side of a task rule corresponds to the trigger notion introduced in BB-1), whilst the right hand side contains a list of one or more specialists to be executed *sequentially* or *opportunistically* depending on the type of control.

Associated with each task is a data structure called *the event-list* which contains all changes on the blackboard(s) not yet taken into account in the problem-solving process. This event-list is examined by the task in order to detect events which can induce specialist KSs activation. Each change on the blackboard(s) is immediately memorized in the event-list of the tasks interested by this change.

In ATOME, we distinguish between three types of tasks :

1. *event-driven tasks*

   The interpreter of an event-driven task takes the event with the highest priority (i.e. the first event) of its local event-list and scans the entire task rules in attempting to fire them.

   The action of a rule firing consists in triggering the specialists in its right hand side. These specialists will be *sequentially* executed in the order in which they are specified in the rule.

   The triggering event will be passed as a focus to the activated specialist KSs.

2. *rule-driven tasks*

   In attempting to fire the most important rule, the interpreter of a rule-driven task scans its entire local event-list to find the events which match its left hand side. Like an event-driven task rule, a rule-driven task firing rule triggers the specialist KSs enumerated in its right hand side. These specialist KSs will also be *sequentially* executed in the order in which they are specified in the rule.

   The events which have matched the firing rule conditions will be passed as a focus to the activated specialists.

3. *opportunistic tasks*

   The interpreter of an opportunistic task scans its entire local event-list and triggers all specialists KSs specified in the right hand side of the firing rules. Each triggering specialist creates a *SPecialist Instance* or SPI which is placed in the *Triggering-SPIs-Agenda* which groups all potential actions. Each triggering SPI which meets the precondition of its corresponding specialist is posted into the *Executable-SPIs-Agenda* which groups all concurrent and feasible actions. From the last agenda, the interpreter uses the *heuristics* proposed by the strategy KS to select one of the pending feasible actions (i.e. an executable SPI) to execute next. In addition, a SPI can be removed from the Triggered-SPIs-Agenda and Executable-SPI-Agenda by the satisfaction of their obviation conditions (cf. Figure 2).

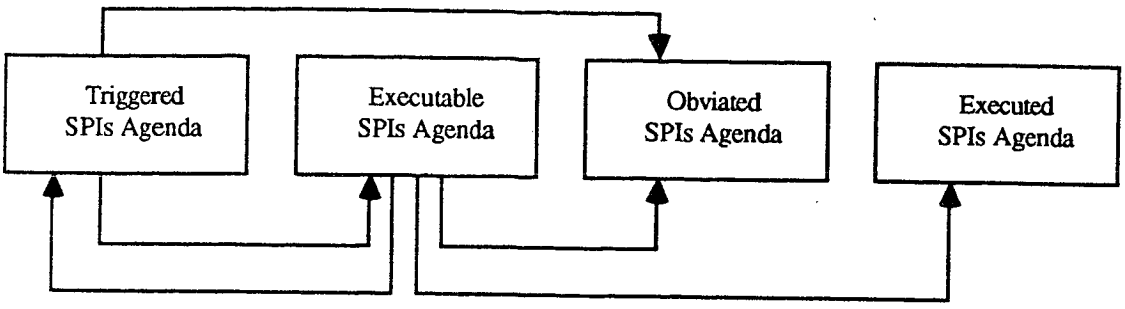


Figure 2 : Agendas used by an opportunistic task.
A SPI moves from the Triggered-SPIs-Agenda to the Executable-SPIs-Agenda when the precondition of its corresponding specialist is satisfied.
A SPI moves from the Triggered-SPIs-Agenda to the Obviated-SPIs-Agenda when the obviation conditions of its associate specialist are satisfied.
A SPI moves from the Executable-SPIs-Agenda to the Triggered-SPIs-Agenda when the precondition of its corresponding specialist becomes unsatisfied.
A SPI moves from the Executable-SPIs-Agenda to the Executed-SPIs-Agenda when the action of its corresponding specialist is executed.
A SPI moves from the Executable-SPIs-Agenda to the Obviated-SPIs-Agenda when the obviation conditions of its associate specialist are satisfied.

Rule-driven tasks are useful when their corresponding specialist KSs require multiple events as a context while event-driven tasks are useful when their specialist KSs require the most important event as a context. These two types of tasks are *efficient* but require that the designer has to precisely describe how to solve conflicts between specialists, they are therefore interesting when the control of the specialist KSs is well known.

Opportunistic tasks present a high overhead of computational cost but they are more *flexible* than the two other types. They are more preferable to use when the control of the specialist KSs is less known. An opportunistic task can therefore be used as a *learning* mechanism for a control KS, and once the instant and the order of the KSs activations are known it is recommended to translate it into an event or rule-driven task.

2.4.2 Strategy Control Level

The strategy KS works as a meta-level control knowledge source. It analyzes the quality of the current solution to determine the region of the data/hypothesis space on which to focus and the set of tasks to process on this region. Consequently the strategy KS chooses a set of sub-problems and the ways to solve them.

The strategy KS is made up of a set of production rules. The left hand side of these rules looks for some hypothesis elements on the blackboard(s) judged as important or necessary by the knowledge engineer and the expert in the problem-solving process. The right hand side contains one or more tasks to execute *sequentially*. A *cycle* corresponds to a rule execution.

For sake of efficiency, each blackboard is associated with a data structure called *the blackboard summary* or BS which memorizes all solution elements considered as important or necessary by the application designer or the user. Instead of examining the entire blackboard(s), the strategy KS will only scan the(se) BS(s) to save time.

2.4.3 ATOME's Basic Control Loop

The problem-solving cycle of an ATOME-system is as follows :

1. The strategy KS interpreter chooses the most important rule with its left hand side satisfied : in order to do this, it scans the BS(s) to find important or necessary solution elements which verify the rule left hand side conditions.

   When a rule fires, the interpreter activates *sequentially* the tasks whose names are specified in the rule right hand side, executes Lisp functions and/or requests the system to stop. The activated tasks can access the nodes selected by the strategy KS.

2. The current task interpreter activates *sequentially* or *opportunistically* specialist KSs depending on its type (event-driven, rule-driven or opportunistic), executes Lisp functions and/or stops the current task, the current cycle or the system itself.

   The current task can focus on the events associated with the nodes selected by the strategy KS.

   The activated specialist KSs can access to the nodes selected by the strategy and to those associated with the events selected by the current task.

3. The current specialist action is executed : it can create, modify or delete solution elements on the blackboard(s), executes programs and/or stops the current specialist, the current task, the current cycle or the system.

4. Goto 1.

An ATOME-system stops when no strategy KS rule fires or when a domain or control KS calls for the stop of the system running.

Figure 3 enhances the opportunistic task activities whilst Figure 4 describes the basic control loop in ATOME.

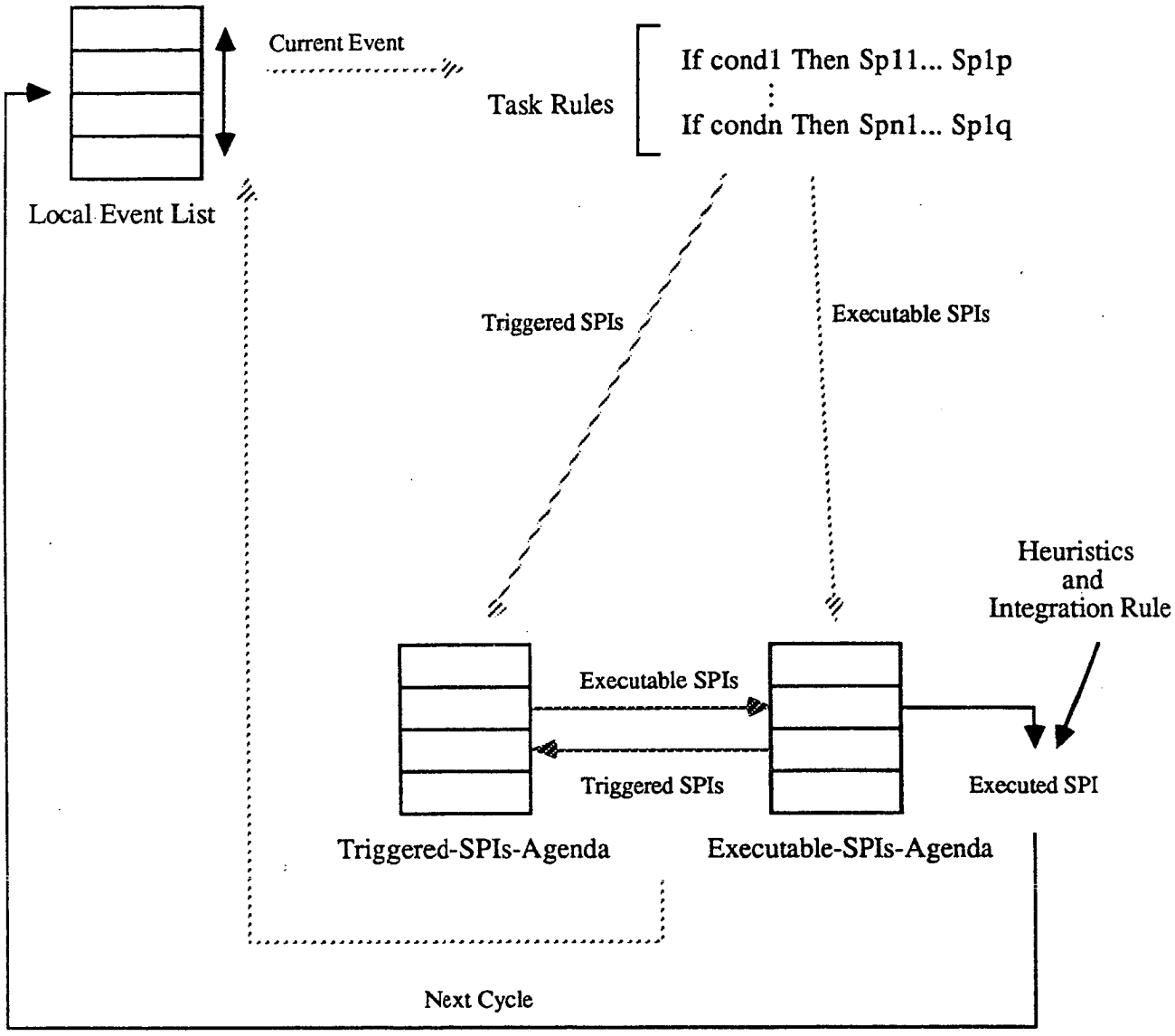


Figure 3 : The Opportunistic Mode Task.
Events of the event list of the task may cause specialists to become triggered by satisfaction of their triggering conditions. A triggered specialist creates instanciation of itself called SPecialist Instance or SPI which represents potential specialist applications on specific hypotheses to satisfy certain goals. Each specialist waits for its precondition to be satisfied, at which point it becomes executable. From the agenda of executable SPIs, the interpreter selects one whose actions appear to best fit the heuristics under the integration rule proposed by the strategy and executes these actions which in turn generate new events. In addition, a SPI can be exchanged from one to another agenda depending of the state of its precondition.

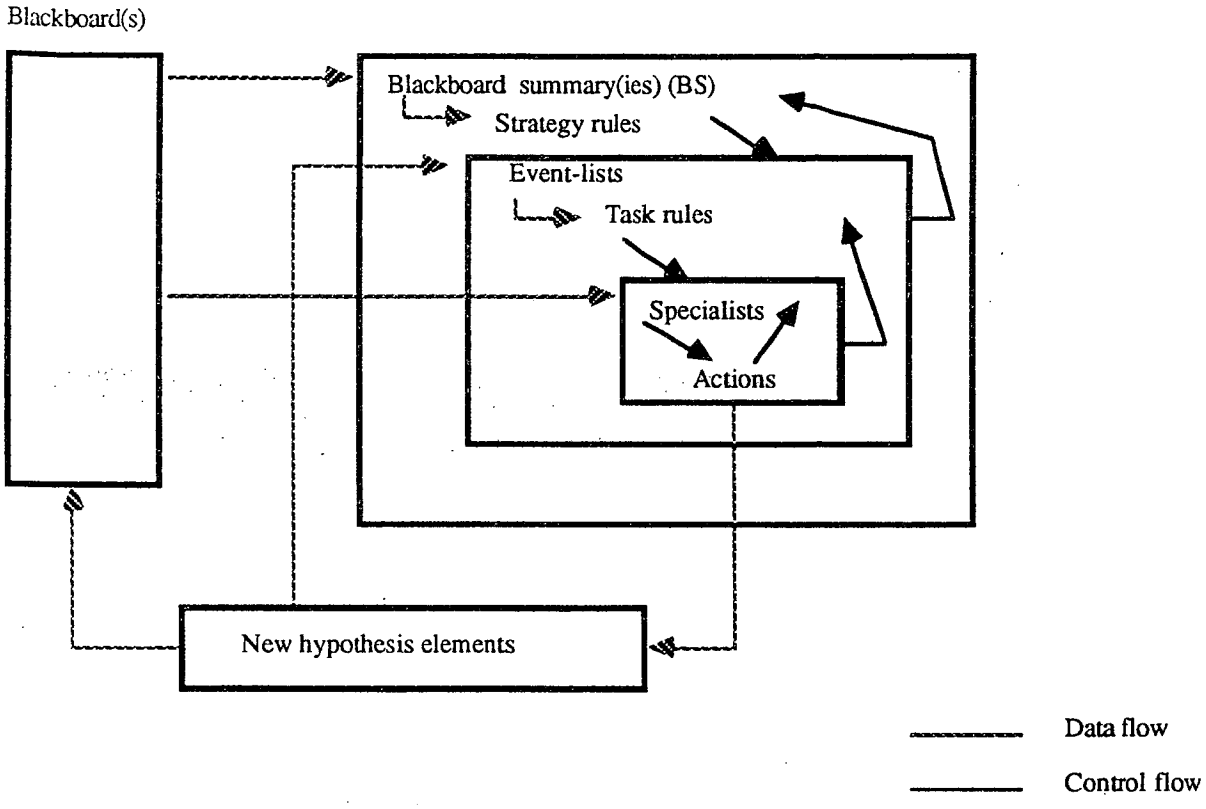


Figure 4 : ATOME's control structure (without the agendas).
The strategy KS examines the BS(s) to select one or a set of tasks to execute sequentially.
The current task scans its local event-list to activate one or more specialists sequentially or opportunistically depending of its type.
The current specialist consults and modifies the blackboard(s). As a result, events are created and be placed onto the event-lists of tasks interested by these changes. The BS(s) is(are) then updated.

For more details on the ATOME control component see [26,27].

3 Temporal Reasoning

As the aim of ATOME is to build large problem solving systems in realistic domains, we have implemented tools for representing and reasoning about time.

Most of applications need to refer to what has happened, is happening and might happen. But such temporal information is often incomplete (only partly ordered), metric (some durations and dates are well known) and in constant evolution (an assumption made at one moment in time may turn out to be untrue later on).

In order to allow the designer to deal with everyday problem solving situations, and because a single perspective of time is inadequate (especially in a multi-experts system), ATOME provides different kinds of temporal representations.

3.1 Dates and Periods of Time

In our system, the algebraic approach of time is privileged, but it is augmented with the advantages of symbolic relationships.

A *date* may be, as in [33], stamped on different calendars (commonly from year to second), with ##[$-\infty$] and ##[$+\infty$] representing the infinity in past and future. But it may be usefull for the designer to reason on a symbolic time step. Accordingly, for using *now* in a temporal expression, it is possible to call the real-time clock with the appropriate calendar (e.g. which returns a date such as ##[88 1] with the calendar of months) as well as a symbolic clock (e.g. which returns a date as t_{27}).

Likewise, a *duration* may be computed on a specific calendar, in which #$d[\infty]$ represents the infinite duration.

In order to reason on interruptable processes and repeatable facts, as argued in [22], we use the notion of *period of time* carried on by a fact or an event. A period of time is a list of ordered *convex intervals* defined on the same calendar where a convex interval is just a pair of dates.

Unfortunately, a large amount of information cannot be linked to a precise time, though the duration of facts is well known or these facts can be related to one another. Therefore, when an interval is not bound by precise dates, the system will use a structure composed of the following attributes :

- a start-date,
- an end-date and
- a duration, (each attribute is either a numerical value or a set of numerical constraints on this value).

Periods are the temporal objects used to index an information. In case of these periods are not composed by precise intervals, the system will use a similar structure composed of the previous attributes and completed with :

- a diameter (duration of the smaller convex interval which contains the period[1]), which is either a numerical value or a set of numerical constraints on this value, and
- a set of symbolic constraints (which are symbolic relationships with other periods as defined in section 3.2).

In section 3.3, we will see how these constraints may be propagated through the blackboard.

[1] In case of a convex period (composed by only one interval), diameter equals duration

3.1.1 Temporal Attributes

It is important for the designer to consider the fact that it is not necessary to memorize or predict all pieces of information. The designer must specify the temporal attributes.

To each temporal attribute of a level in the blackboard, is associated a calendar, numerical or symbolic, on which the periods of time of its values are defined. Moreover, the persistence of the values (extent in time of an information when discovered) is defined for this attribute. The default ($\#d[\infty]$) simply means that a value of an attribute will last just until something is known that make it become false, and may be modified during reasoning. This persistence clipping is achieved explicitly by the problem solver when a new value is detected thanks to a specific function ($$replace) available to the designer.

3.2 Temporal Functions

To enable reasoning on succession, simultaneity and duration of events, a set of numerical and symbolic functions are provided to the designer.

Concerning the numerical aspects of time, the following functions are available :

- computation on numerical dates and durations defined on different calendars,
- comparison of durations defined on different calendars,
- computation on precise periods (union, intersection, complementary) on different calendars, ...

When different numerical calendars are used in a function, an automatic transformation of the temporal objects on the more specific calendar is computed. It is not possible to refer both to symbolic and real-time clock in the same function.

ATOME provides the following symbolic relations for retrieving or adding relationships between periods, based on ideas proposed by Ladkin [23] :

- always, sometimes (associated with disjunctions of convex interval relations defined by Allen [1]),
- equals, disjoint from,
  precedes, follows,
  meets, met by,
  overlaps, overlapped by,
  contains, contained by,
  begins, ends,
  begins with, begins before, begins after, ends with, ends before, ends after,
  alternates, ...

Notice that the 13 relations of Allen, specific to convex intervals, are included on the functions dealing with periods.

When the system retrieves a relationship between two periods, the numerical representation is considered first with eventually different calendars. If the date information is not precise enough, numerical constraints are used.

3.3 Constraints Propagation

A network search is not applied when retrieving a relationship. We have considered that the constraints propagation was performed by an appropriate propagation system. This system must not be called systematically as soon as a new symbolic or numerical constraint is added in the blackboard due to the amount of deductive activity implied [24].

Since some situations only require a minimum of constraints to be propagated (or no constraint at all) and others may have a high level of urgency, the problem solver must adapt the amount of propagation during the reasoning process.

We are presently developing functions which will enable the designer to build a special task dedicated to the control of constraint propagation according to the current situation and to the resolution of detected inconsistencies between constraints.

These functions include the following propagation facilities :

- computation of the minimum and maximum of one of the attributes (start-date, end-date, duration or diameter) of a temporal object, from the associated set of constraints;
- propagation of symbolic constraints through a level of a specific blackboard or from a level (up or down) to another one;
- propagation of symbolic constraints restricted to two specific objects of a blackboard;
- transposition of constraints from a symbolic to a numerical representation (as we focus on numerical representation of time, the reverse function is not considered);
- deletion or restoration of constraints.

But whenever a constraint is removed due to the appearance of an inconsistency (and more generally when the validity of a fact is stopped), the system has to determine how the set of beliefs is affected by these changes [6]. This essential function for temporal reasoning, i.e. *temporal reasoning maintenance*, is considered in section 4.

3.4 Historical Functions

ATOME, through the utilization of different blackboards and different levels in a specific blackboard, enables to partition the set of information in a syntactic basis.

Our idea is also to temporally partition the information.

Since a precise information is usually no longer available after a certain time, this partition may be done by removing old information from the current blackboard (called the short term memory, STM) and eventually storing it in another blackboard (called the long term memory, LTM) dedicated to long term analyses [17].

But creating a LTM does not consist only of a transfer of information since this information may need to be aggregated with others. Moreover, the frequency of memorization in the historical blackboard is not necessarily a well defined value. It may vary according to the current context. For example, the precise values of temperature are not to be remembered during all the life-time of a physical system. On the long term memory, the average between two important variations of the measurements may only be needed.

Historical functions in ATOME enable the designer to define the rate of transfer of information from the STM to the LTM according to the context and the action of aggregation the memorization system must apply on this information.

4 Hypothetical Reasoning

Most present knowledge base systems run without strong connections with the outside world. However, more and more industrial applications require a large numbers of input-output interactions, and outer information is necessary for maintaining the coherence with the real world of the deductions.

Such a framework must be supported by a set of tools and facilities providing a development environment. In fact, the following points constitute the main issues to be handled :

- drawing inferences from incomplete, inaccurate, ambiguous and/or conflicting data,
- introducing hypotheses,
- taking into account continuous flows of information,
- assuming data consistency with good efficiency,
- reasoning about time, and
- fusioning multi sources data.

In order to take into account these issues, we aim at providing ATOME with a shell which allows contradiction, hypothetical knowledge and data dependency in both the blackboard(s) and the knowledge sources. A way for solving these points consists in implementing a truth maintenance system. Two approaches can be considered :

- systems based on Doyle 's TMS principles which provide a tool for maintaining the data dependency and for dependency-directed backtracking [7];
- systems based on DeKleer's ATMS principles which are based on the manipulation of assumption sets, avoiding context switching and backtracking encountered in TMS.

We have chosen to develop a system based on the second approach. The objective of our work is to reason in a multiple world paradigm, in the sense of ART [2] which has several similarities with ATMS, but with the advantage of allowing negated assumptions (i.e. facts are retractable under specified worlds).

We are implementing a package of functions which can generate hypotheses (in the sense of blackboard systems) and assumptions(in the sense of ATMS). These functions draw conclusions, modify the blackboard structure and construct data dependencies.

4.1 Data Dependencies.

In ATOME, a specific mechanism keeps knowledge consistency by using data dependency. Instead of constructing justifications for each deduced fact or knowledge, it is possible to choose both the data which justify thanks to the function "justifier" and the data which are justified thanks to the function "justified". Therefore the user builds up a "while then" association between data, assuming that if a justifier is removed from the blackboard structure or from a local database, then the deductions which depend on deleted information are retracted from the worlds concerned by the deletion. Two cases have to be considered :

- the retracted information justified a data
- the retracted information justified a data's retract world.

In the first case, the data is removed; in the second case the data is modified by retracting the world belonging from the retract list concerned with the previous justification.

This approach makes it possible to use data dependency in procedural knowledge sources as well as in production rules. In a procedural KS, the user can collect the reading access from the blackboard structure and build up one or several justification lists then associated with the writing access. The use of functions "justifier" and "justified" therefore provides flexibility in constructing data dependency. The use of data dependencies is not necessary in an ATMS framework. De Kleer suggests [3,4,5] to make justification retractable by associating an assumption which represents its defeasibility. Indeed, a direct justification retraction should be inefficient in ATMS since the recomputation of all the nodes recursively invalidate by the retracted justification. However, the increase of assumptions is not always desirable. We have adopted a different implementation by duplicating the facts which hold in

alternative worlds. We are thus able to build a data dependency algorithm which is really efficient.

4.2 Multiple World Reasoning

In conventional blackboard systems all pieces of information present in the blackboard structure are hypotheses. If this approach is general enough for problem solving, it however does not provide any help for assuring the logical and semantic consistency of the deductions that are made. For that reason, we aim at providing the user with facilities for maintaining the coherence of the deductions done on the basis of assumptions.

So the user creates an assumption or a world every time he wants to :

- explore alternatives,
- assess multiple solution strategies,
- merge different alternative solutions,
- deduce conditional data (for instance in contextual reasoning),
- shadow part of the deductions to focus the reasoning on chosen assumptions,
- represent successive or alternative states (for instance temporal states),
- resolve constraints satisfaction[16].

Our approach is similar to ART's contexts and viewpoints and we have chosen the same terminology. We adopt a similar representation by associating information present in the blackboard structure and knowledge source bases with a label. This label (or extend) defines the set of viewpoints within which a data holds. As an inheritance relation exists between a viewpoint and its descendants, the extend incorporates only the most general viewpoint V within which the data is considered to be true. We call this viewpoint the base viewpoint. As we allow data retraction, a fact could be retracted from viewpoints that inherit from the base viewpoint. In this case the label have also to contain the descendant of V under which the data is shadowed. We call these viewpoints the set of retraction viewpoints. So a label is a kind of interval applied on the viewpoint's structure. A label can thus be represented by the following structure :

label= base viewpoint|set of retraction viewpoints

4.2.1 Multiple Justification

As, we do not allow for disjunction of assumptions and systematic justification, it is necessary to duplicate data which hold in alternative viewpoints. Three cases have to be considered when a new derivation is inferred for an existing data.

Let F be a data which holds in
C1|R11 R12 — R1n(1)
C2|R21 R22 — R2n(2)
— — — — — — — — —
Cp|Rp1 Rp2 — Rpn(p)
where Ci represents the alternative base viewpoint of F and Ri1 — Rin(i) represents the set of retraction viewpoints of F observed from Ci. C|R1 — Rn is the new deduction.

1. for all i such that Ci is more general than C or equal to C :
   in this case the label Ci|Ri1 — Rin(i) is modify by incorporating to its retract list the new retractions R1 — Rn.

2. for all i such that Ci is more specific than C :
   in this case the old label is removed and replaced by C|(R1 — Rn) U (Ri1 — Rin(i)).

3. in the other cases the new inference is conserved.

4.2.2 Merging Viewpoints

All deduction are made by collecting various information belonging to single or alternative viewpoints. In a production rule system, the left-hand side of the rule gives the conditions under which the action part can be activated. In a procedural knowledge source, the reading access to the blackboard and to the local knowledge bases are used for computing new information which is then posted on the blackboard structure. In both cases the question consists in determining in which viewpoint the deduction is valid. Two cases has to be considered. Either, the user specifies explicitly the validity of the inserted deduction, or a default mechanism has to take care of this problem. In the second case, the only way for assuring the coherency is to decide that any information used for a deduction has to be believed in an unique viewpoint which becomes the validity field of the new information. If all information needed for the inference is not believed in the same viewpoint, the system tries to merge the different viewpoints in order to create the most general viewpoint containing all of them. Each time a new viewpoint is discovered the consistency is verified according to the contradiction knowledge.

Let A and B be data holding by : CA|RA1-RAn and CB|RB1-RBn.
The most general viewpoint containing A and B is defined by :
merge(CA,CB)|(RA1-RAn) ∪ (RB1-RBn).

A deduction may sometimes be conditioned by the non believe of a data. It is then necessary to compute the viewpoints in which a data is believed and in which an other data is not believed. The viewpoints in which A holds and B is not present are thus defined by CA|CB ∪ (RA1-RAn)
more precisely (CA,RBi)|RA1-RAn for every RBi in RB1-RBn

4.3 Contradiction Rules and KSs

For assuming the semantic coherence, it is possible to define contradiction rules or KSs in order to cancel the hypothetical worlds within which different hypotheses or deduced information are inconsistent. Such rules can specify constraint depending upon the problem and provide a means for controlling the generation of worlds.

4.4 Present Developments

Our approach of hypothetical reasoning is pragmatic and does not assure the consistency of the blackboard structure and of local knowledge bases in all cases. However, it gives a powerful help for developing applications.

The implementation of both data dependency and viewpoints is efficient, and guarantees a minimal computation time between two inferences. This aspect is very important for real time applications.

We are presently focusing on an efficient representation of viewpoints through the different levels of the blackboard structure and we also study a control model (for instance shadowing hypothesis to focus the reasoning on a chosen direction or controlling the viewpoint creation).

5 Conclusion

ATOME is a shell for building blackboard-based systems which provides to combine efficiency and opportunism. Its hierarchical control architecture emphasizes the efficiency of the system while the opportunistic tasks introduce flexibility.

ATOME is written in Le_Lisp and runs on a SUN worstation under UNIX system.

The implementation of temporal reasoning tools makes it easy to take into account data evolving in time whereas the implementation of hypothetical reasoning is currently under development and is intended to contribute to maintain coherence and to introduce deductions based on assumptions.

All these mechanisms are required for a multi sensor data fusion application on which we are working.

Acknowledgements

We thank all persons who take an interest in ATOME project. Special thanks to Cognitech knowledge engineers for the stimulating discussions we regularly have with them.

References

[1] J. F. Allen. Maintaining Knowledge about Temporal Intervals. *Communica-*

tions of the ACM, 26:832–843, 1983.

[2] B. D. Clayton. *A First Look at Viewpoints.* Inference Corporation, 1985.

[3] J. De Kleer. An Assumption-based TMS. *Artificial Intelligence*, 28:127–162, 1986.

[4] J. De Kleer. Extending the ATMS. *Artificial Intelligence*, 28:163–186, 1986.

[5] J. De Kleer. Problem Solving with the ATMS. *Artificial Intelligence*, 28:197–223, 1986.

[6] T. L. Dean and D. V. McDermott. Temporal Data Base Management. *Artificial Intelligence*, 32:1–55, 1987.

[7] J. Doyle. A Truth Maintenance System. *Artificial Intelligence*, 12:231–272, 1979.

[8] E. H. Durfee and V. R. Lesser. Incremental Planning to Control a Blackboard Based Problem Solver. In *Proceedings of the 5th National Conference on Artificial Intelligence*, pages 58–64, Philadelphia, Pa, August 1986.

[9] E. H. Durfee, V. R. Lesser, and D. D. Corkill. Cooperation through Communication in a Distributed Problem Solving Network. In *Distributed Artificial Intelligence*, pages 29–58, M. N. Huhns Editor, Pitman, 1987.

[10] R. Engelmore and A. Terry. Structure and Function of the CRYSALIS System. In *Proceedings of the 6th International Joint Conference on Artificial Intelligence*, pages 250–256, Tokyo, Japan, 1979.

[11] L. D. Erman, F. Hayes-Roth, V. R. Lesser, and R. D. Reddy. The HEARSAY-II Speech Understanding System : Integrating Knowledge to Resolve Uncertainty. *ACM Computing Surveys*, 12:213–253, 1980.

[12] M. R. Fehling, S. Forrest, and B. M. Wilber. The Heuristic Control Virtual Machine. In *Proceedings of the Workshop on Blackboard Systems : AAAI-87*, Seattle, Wa, July 13, 1987.

[13] A. Garvey, M. Hewett, M. V. Johnson Jr., R. Schulman, and B. Hayes-Roth. *BB-1 User Manual - Common LISP Version 2.0.* Technical Report KSL-86-61, Knowledge Systems Laboratory, Computer Science Department, Stanford University, August 1987.

[14] J. P. Haton, B. Maître, H. Lâasri, and T. Mondot. ATOME : Another TOol for developing Multi-Expert Systems. In *Proceedings of the Workshop on Blackboard Systems : AAAI-87*, Seattle, Wa, July 13, 1987.

[15] B. Hayes-Roth. A Blackboard Architecture for Control. *Artificial Intelligence*, 26:251–321, July 1985.

[16] V. Jagannathan, L. Baum, and R. Dodhiawala. *Constraint Satisfaction and Reasoning in the Boeing Blackboard System.* Technical Report, Boeing Computer Services, P.O.Box 24346 Seattle, WA, 98124, 1986.

[17] J. L. Kolodner. Maintaining Organization in a Dynamic Long-Term Memory. *Cognitive Science*, 7:243–280, 1983.

[18] H. Lâasri, B. Maître, and J. P. Haton. ATOME : Outil d'Aide au Développement de Systèmes Multi-Experts. *Actes du 6ème Congrès AFCET-RFIA*, Antibes, France, 16-20 Novembre 1987.

[19] H. Lâasri, B. Maître, and J. P. Haton. *Opportunism and Efficiency in Meta-Level Blackboard Architectures : ATOME Case Study.* Technical Report 88-R-007, CRIN/INRIA-Lorraine, 1988.

[20] H. Lâasri, B. Maître, and J. P. Haton. Organisation, Coopération et Exploitation des Connaissances dans les Architectures de Blackboard : Cas de ATOME. *Actes des 8ièmes journées internationales sur les systèmes experts et leurs applications*, Avignon, France, 30 Mai-03 Juin, 1988.

[21] H. Lâasri, B. Maître, T. Mondot, F. Charpillet, and J. P. Haton. ATOME : A Blackboard Architecture with Temporal and Hypothetical Reasoning. In *Proceedings of the European Conference on Artificial Intelligence*, Munich, W. Germany, August 01-05, 1988.

[22] P. B. Ladkin. The Completeness of a Natural System for Reasoning with Time Intervals. In *Proceedings of the 10th International Joint Conference on Artificial Intelligence*, pages 462–467, Milan, Italy, 1987.

[23] P. B. Ladkin. *Two Papers on Time Representation.* Research Report KES.U.86.5, Kestrel Institute, 1986.

[24] C. Le Pape and A. Collinot. Controlling Constraint Propagation. In *Proceedings of the 10th International Joint Conference on Artificial Intelligence*, pages 1032–1034, Milan, Italy, 1987.

[25] V. R. Lesser and D. D. Corkill. The Distributed Vehicle Monitoring Testbed : A Tool for Investigating Distributed Problem Solving Networks. *AI Magazine*, 4(3):15–33, Fall 1983.

[26] B. Maître and H. Lâasri. *Manuel d'Utilisation de ATOME : Version 1.0.* Rapport Technique 87-T-099, CRIN/INRIA-Lorraine, Novembre 1987.

[27] B. Maître, H. Lâasri, and T. Mondot. *Manuel d'Utilisation de ATOME : Version 1.1.* Rapport Technique 87-T-110, CRIN/INRIA-Lorraine, Novembre 1987.

[28] B. Maître, H. Lâasri, T. Mondot, and J. P. Haton. ATOME : Advanced Tool for Multi-Level Knowledge Organization. *Second International Conference on Expert Systems and the Leading Edge in Production Planning and Control*, Charleston, SC, May 3-5, 1988.

[29] H. P. Nii. Blackboard Systems : Blackboard Application Systems, Blackboard Systems from a Knowledge Engineering Perspective. *AI Magazine*, 7(3):82–106, 1986.

[30] H. P. Nii. Blackboard Systems : The Blackboard Model of Problem Solving and the Evolution of Blackboard Architectures. *AI Magazine*, 7(2):38–53, 1986.

[31] H. P. Nii. *Signal-to-Symbol Transformation : Reasoning in the HASP/SIAP Program.* Technical Report HPP-83-44, Heuristic Programming Project, Computer Science Department, Stanford University. Stanford, Ca, December 1983.

[32] H. P. Nii, E. A. Feigenbaum, J. J. Anton, and A. J. Rockmore. Signal-to-Symbol Transformation : HASP/SIAP Case Study. *AI Magazine*, 3(2):23–35, 1982.

[33] S. F. Smith. *Exploiting Temporal Knowledge to Organize Constraints.* Technical Report CMU-RI-TR-83-12, Carnegie Mellon University, July 1983.

[34] A. Terry. *The CRYSALIS Project : Hierarchical Control of Production Systems.* Technical Report HPP-83-19, Heuristic Programming Project, Computer Science Department, Stanford University. Stanford, Ca, 1983.

Imprimé en France
par
l'Institut National de Recherche en Informatique et en Automatique