Research report 103

CONFER: A KNOWLEDGE SYSTEM FOR BIOPROCESS CONTROL

Iain D Craig & David H Wilson

(RR103)

Abstract

The control and monitoring of fermentation processes is a knowledge-intensive acitvity which requires the application of a considerable body of expert knowledge. This paper describes and motivates CONFER, a knowledge system for real time bioprocess monitoring and control. CONFER contains a blackboard problem-solving system which is augmented by a declarative knowledge base and a collection of process models. The system uses the knowledge base and models to improve its problem-solving performance. CONFER is designed to be accountable, so detailed explanations of its activities are essential. Unlike other systems, explanation is the basis of the system design and partially motivates the inclusion of the knowledge base and the inclusion of the knowledge base and process models.

Department of Computer Science
University of Warwick
Coventry
CV4 7AL, UK

June 1987

CONFER: A KNOWLEDGE SYSTEM FOR BIOPROCESS CONTROL

Iain D. Craig and David H. Wilson

Department of Computer Science,

University of Warwick,

Coventry CV4 7AL, UK

1. INTRODUCTION

The control and monitoring of fermentation processes is a knowledge-intensive activity and requires the application of a considerable body of expert knowledge. The CONFER system is concerned with the real-time control of a batch fermentation process, informing the user of the state of the process and recommending an optimum time to harvest the product. In addition, CONFER can justify its actions and explain what is happening in the process at various levels of detail. In order to perform this task, it is necessary for the system to include extensive knowledge of the fermentation process, the kinds of data available from the fermentor and the expert knowledge required to use this data to infer the current state of the process.

The system uses the blackboard model of problem-solving (Erman, 1975; Feigenbaum, 1978; Hayes-Roth, 1983; Nii, 1986a; Craig, 1987a) to perform the monitoring and control functions. The blackboard model was chosen because it is one of the most powerful and flexible problem-solving architectures known. The architecture allows reasoning processes to be interfaced to external processes such as data-gathering and actuator control, and it also permits extensive and explicit representation of control and other information. The choice is also motivated by the fact that fermentation control is a problem which demands

the ability to reason using many, different kinds of knowledge. Additionally, it allows explicit access to all decisions made and actions taken during the problem-solving process: such explicit access is required for explanation.

The CONFER prototype is being designed to control and monitor the production of β-galactosidase from a mixed medium. This process was selected because it is well understood and because it embodies the problems generally found in fermentation processes. Although there are no direct commercial applications of this particular process, its fundamental properties are universal to this class of problem.

The structure of this paper is as follows. In the next section, we outline the enzyme production process at operon level. Section three briefly describes the blackboard architecture. Section four is concerned with the role of explanation in CONFER. The fifth section presents an overview of the architecture of CONFER. The final section recapitulates the main points of the paper.

1.1. Acknowledgements

The authors would like to express their gratitude to Anthony Richards of the Department of Biological Sciences, University of Warwick, our domain expert without whom this project would not be possible.

2. STRUCTURE, FUNCTION AND BEHAVIOUR OF THE LAC OPERON

During the growth of the bacterium *Escherichia coli* on a minimal medium containing lactose, the enzyme β-galactosidase is induced to change lactose into a form that can be metabolised by the *E.coli* cells. The induction of this enzyme is controlled, both positively and negatively, so that it occurs in high concentrations in the cell only when required. For example, during growth on glucose media there is no requirement for the enzyme and its production is not induced. The mechanisms for inducing and for controlling the induction of the β-galactosidase enzyme, and of controlling its induction, are centred on a particular sequence on the DNA in each *E.coli* nucleus. The sequence of genes for encoding both the enzymes and the regulatory system is called an *operon*.

The result of this control is that, under optimum conditions, there is a dramatic change in the concentrations of the β-galactosidase in the cell. If the substrate provided for growth is glucose or glycerol, five copies of the β-galactosidase enzyme are present in each cell. If the substrate is switched to lactose, the

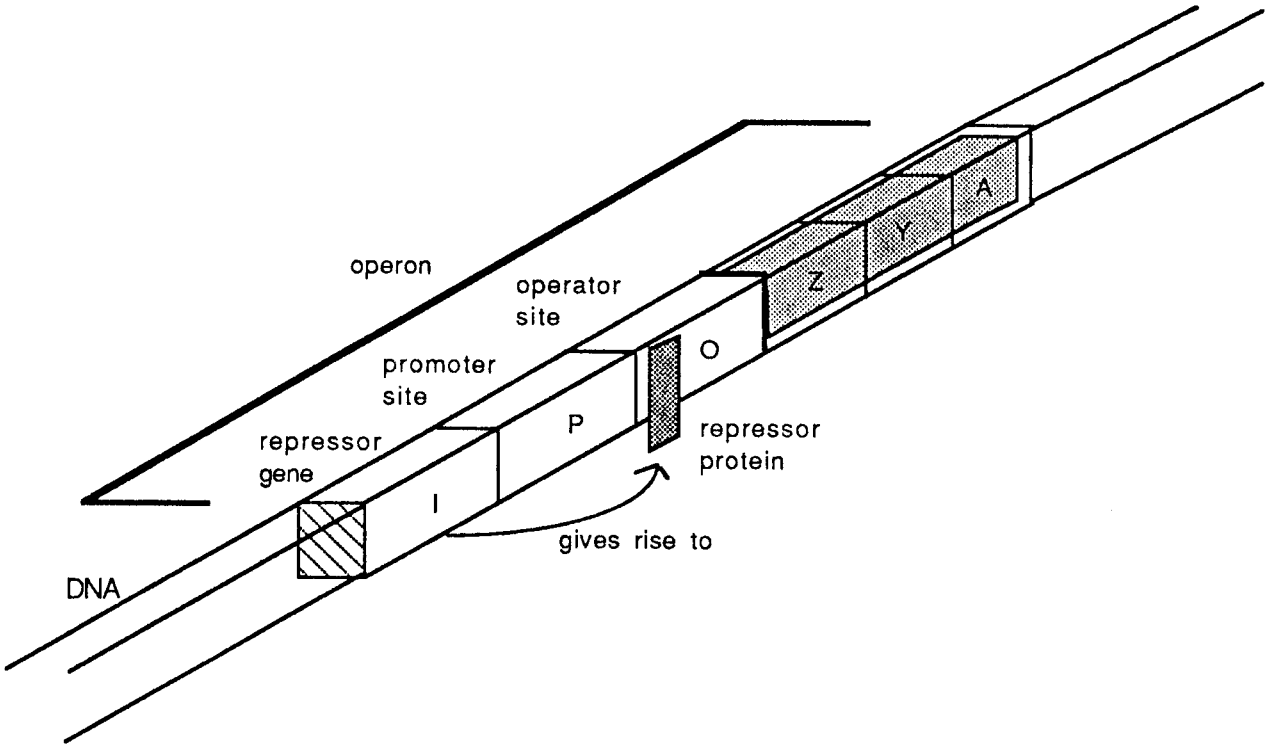

Figure 1. Structure of the lac operon

concentration of the enzyme increases by a factor of 1000 - 10000 (Jacob, 1961).

2.1. Operon Structure

The *lac* operon is responsible for the production of three proteins involved in promoting growth on a lactose substrate. These are:

(i) β-galactosidase - responsible for splitting the lactose into glucose and galactose (*Z*),

(ii) permease - responsible for increasing the levels of lactose uptake by the cell (*Y*), and

(iii) transacetylase - role remains unclear (*A*).

The genes for these proteins form the operon which responds to the presence of lactose (or some other chemically similar β-galactoside). The ability of these genes to transcribe messenger Ribonucleic Acid (mRNA) is controlled by the state of the Operator (*O*), Promoter (*P*) and Repressor gene (*I*) which precede the operon on the DNA chain (see figure 1). The Operator is inhibited by a repressor protein which prevents activation of the operon unless lactose is present in the medium.

2.2. Operon Behaviour

Normally the Repressor gene (the *I* gene) produces mRNA for a protein which specifically regulates the operon. This is the repressor protein which is normally bound to the Operator, preventing mRNA transcription of the genes *lac Z Y A*. The Inducer enters the cell and binds to the repressor. This reduces the ability of the repressor to occupy the Operator site. Released from the repressor, the Operator is now available to the protein which builds RNA. This is the RNA polymerase which builds mRNA as specified by the DNA code. The mRNA sequence is specified by the *Z*, *Y* and *A* genes of the operon. The mRNA moves to the synthesising machinery of the cell where the code it carries is translated into small amounts of the three proteins (enzymes). Translation occurs through the binding of free amino acids to each other in an order specified by the mRNA nucleotide sequence.*

In order that large amounts of the desired enzymes be produced, the Promoter must also be bound. The Promoter is bound by a Catabolite Activator Protein (CAP). This CAP binding is itself dependent on

* For a more detailed account of operon behaviour, see (Darnell, 1986) and (Stent, 1971).

	Lactose present	Lactose absent
Glucose present	cell cAMP levels low so no CAP binding to Promoter. Inducer bound to repressor so Operator active. Result: low enzyme production	cell cAMP levels low so no CAP binding to Promoter. repressor remains bound so Operator inactive. Result: no enzyme production
Glucose absent	cell cAMP levels high so CAP binding to Promoter. Inducer bound to repressor so Operator active. Result: rapid enzyme production	cell cAMP levels high so CAP binding to Promoter. repressor remains bound so Operator inactive. Result: no enzyme production

.Figure2. Mixed medium effects on enzyme production

there being high levels of cyclic Adenosine Monophosphate (cAMP) in the cell, and this in turn depends on there being low levels of catabolites (such as glucose) in the medium. Given low glucose levels, the CAP and cAMP bind and, together with an RNA polymerase, activate the Promoter. The bound Promoter increases the Operator activity which increases the rate of operon transcription of mRNA, and hence ultimately increases the levels of enzyme in the cell. High levels of glucose, a lactose catabolite, prevent the process of CAP binding by interfering with the conversion of Adenosine Triphosphate (ATP) into cAMP. The inhibition of a process by a catabolite is termed *catabolite repression.*

In addition to this control of the Promoter by contingent CAP binding, there is a degree of overlap between the Operator and the Promoter such that if the repressor is bound to the Operator, it also effects CAP activation of the Promoter. This has the result that even if the glucose levels are low, if lactose levels are also low, such binding will not occur because the binding site on the Promoter remains partially occupied by the repressor on the Operator.

Negative control of the operon is manifested in the production of the repressor protein by the *I* gene, resulting in inhibition of the operon. The positive control of the operon is achieved by the interaction of the CAP/cAMP and RNA polymerase causing the Promoter to increase enzyme transcription. The interaction of these forms of control is summarised in figure 2.

2.3. System Function

The growth of *E.coli* on a batch culture of a glucose/lactose medium is diauxic. That is, growth displays two distinct phases (see figure 3). Initially, the bacteria metabolise the available glucose and multiply rapidly. Once the glucose becomes exhausted, the specific growth rate reaches a plateau (see figure 3). After a period, rapid growth resumes as the *E.coli* operons respond, enabling the cell to utilise the lactose in the medium. To be metabolised, the lactose must be taken into the cells and split to produce glucose and galactose. The resulting glucose can then be metabolised to promote further growth. To perform this function, the *E.coli* cells produce large quantities of the permease, transacetylase and β-galactosidase enzymes. Production of these enzymes is triggered only when glucose levels are low and lactose levels are high. High levels of these enzymes are maintained for the period of growth on the lactose substrate.

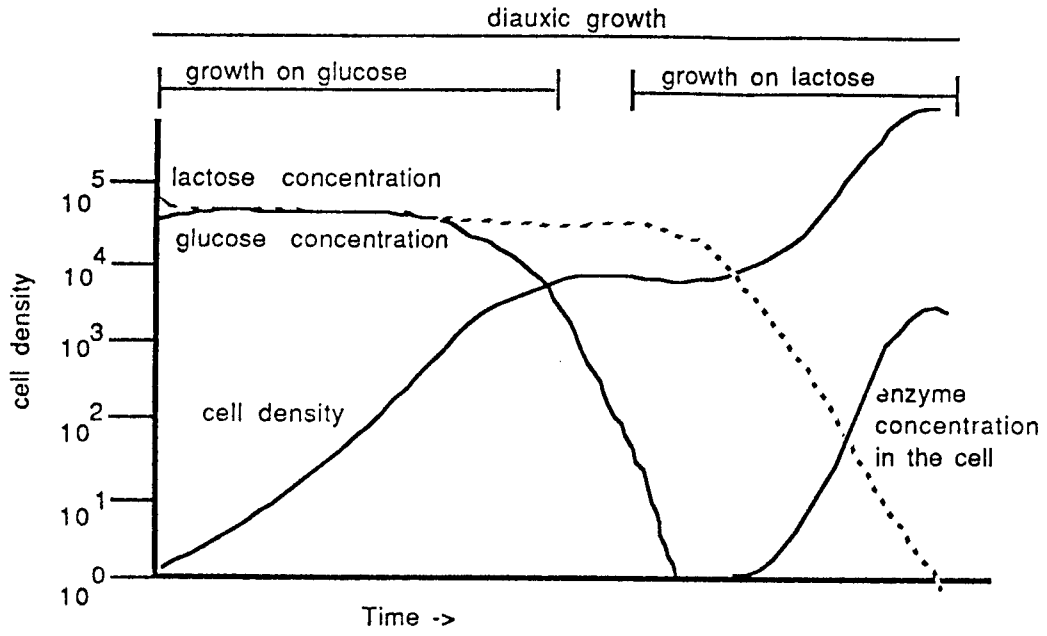

Figure 3. Generalised graph of the fermentation of b-galactoside from a mixed medium.

The primary goal of the CONFER system is the determination of the time at which the maximum concentration of the β-galactosidase enzyme is reached. Production of β-galactosidase is initiated when the glucose in the medium is exhausted. Production continues until lactose has also been metabolised, at which point the enzymes are rapidly broken down by the cell. Given only data about gas uptake and pH changes, the system must be able to infer the state of the process and what is occurring at the operon level in terms of operon control. Using this knowledge, the CONFER system explains why the process is behaving in a particular fashion at various levels of abstraction.

3. THE BLACKBOARD ARCHITECTURE

The blackboard architecture for problem-solving has been developed from work on a speech understanding system called HEARSAY-II (Erman, 1975; Hayes-Roth, 1977; Erman, 1980). The architecture has found favour as the basis for designing and building AI systems operating in complex domains, sometimes in real time. It is attractive because it provides a method for dealing with complexity, a property not shared by any other commonly used architecture, and because of its enormous power and flexibility. This section describes the architecture which is depicted schematically in figure 4. For further information, the reader is recommended to consult (Hayes-Roth, 1983; Hayes-Roth, 1985; Nii, 1986a; Nii, 1986b; Craig, 1987a; Craig, 1987b).

3.1. The Blackboard

At the centre of the blackboard architecture is the *blackboard*, a hierarchically organised database containing solution elements generated by the problem-solving process. The blackboard contains two or more *abstraction levels* which represent the problem space at different levels of detail. Solution elements at varying levels of abstraction represent the state of the solution, and are typically linked together into solution "islands", each of which represents a large-grained partial solution to part of the problem. The organisation of the blackboard into abstraction levels gives the architecture considerable power, for it is possible to make problem-solving decisions of varying extents: those in the more abstract regions of the blackboard cover larger portions of the solution space than do those at lower levels. By structuring the blackboard as an abstraction hierarchy, solutions can be generated by progressive refinement (top-down), by progressive

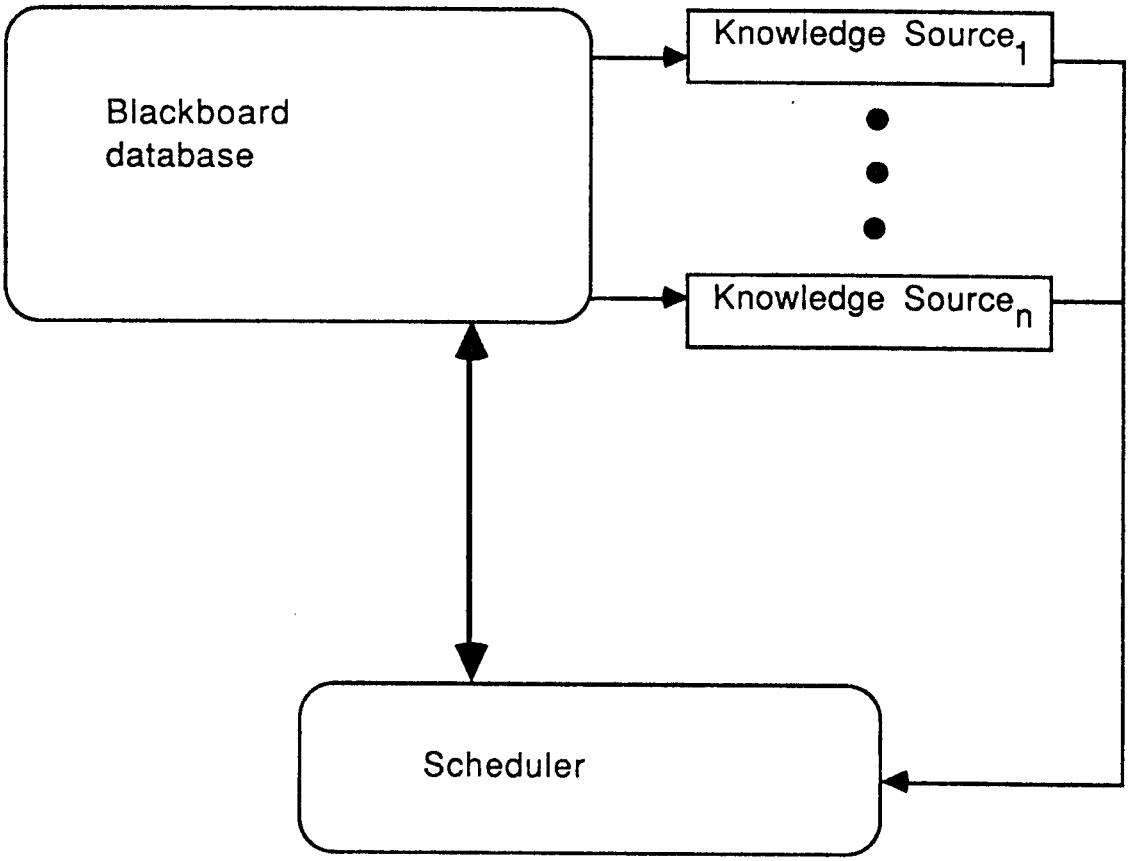

Figure 4. The Blackboard Architecture

abstraction (bottom-up) or a combination of the two. A blackboard system is, therefore, able to view the problem to be solved in different ways at the same time.

3.2. Knowledge Sources

Solution elements are placed on the blackboard by *Knowledge Sources.* Knowledge Sources (KSs) represent large-grained "chunks" of problem-solving knowledge: they can be thought of as "mini-experts", each knowledgeable in some highly restricted part of the problem domain. For example, in the speech understanding task, there is a Knowledge Source which can determine the words which can be hypothesised from a set of syllables. Knowledge Sources are independent modules which respond to events on the blackboard. A blackboard event is either the addition or modification of a solution element. When an event occurs, some or all Knowledge Sources in the system respond to it (this is referred to as *triggering*) and become candidates for activation. If a KS is selected for activation, or *scheduled*, it can update the blackboard causing more blackboard events. Knowledge Sources may, though, only respond to blackboard events and may only indirectly communicate with each other by placing or modifying solution elements on the blackboard -- they are not permitted to engage in any direct communication with other KSs.

Knowledge Sources respond to events at different levels of abstraction and may act upon the same or upon other levels. That is, if a KS triggers on a particular abstraction level, its action may execute at that or at another level. By acting at different levels of abstraction, solution islands are created and progressively linked into a solution to the problem. Problem-solving activity terminates when some, problem-specific, condition holds on the blackboard: this is detected either by a Knowledge Source or by a monitor process whose role is to control the activities of the system.

3.3. Control

The architecture operates a basic match-deliberate-act cycle. During the match phase, Knowledge Sources respond to blackboard events. During the deliberation phase, the system determines which KSs are to be executed: those KSs selected during the deliberation phase are executed in the last part of the cycle and cause blackboard events.

At any point in the problem-solving process, one or more Knowledge Sources may respond to

blackboard events. Unlike other architectures (most notably, the production rule architecture (Newell, 1973)), it is not necessarily the case that any given KS will be executed on the same cycle as it was triggered. Knowledge Sources can wait many cycles before execution, and are held in the meantime in a control database. The control database, which is typically organised as an agenda, is used by the system controller or *scheduler*. The scheduler is responsible for executing Knowledge Sources executions according to the prescriptions of one or more control strategies. It operates by examining the agenda and selecting the most appropriate Knowledge Source to execute, given the prevailing blackboard state. The criteria determining the most appropriate KS are provided by the control strategy in force at selection time.

The implementation of control as a strategy-based process operating on a control memory is the second main source of power to the blackboard architecture. The architecture permits the concurrent or sequential application of a mixture of strategies, the mixture being determined by the needs of the problem under attack. Combinations of bottom-up and top-down reasoning can easily be accommodated by blackboard systems and may be intergrated with other types of strategy. The HASP/SIAP sonar interpretation system (Feigenbaum, 1978; Nii, 1982) introduced a model-driven strategy into a basically bottom-up process; the HEARSAY-II system adopted a two-phase strategy combining bottom-up processing with an opportunistic second phase.

Opportunistic control is often cited as *the* differentiating property of blackboard systems (but, for a different perspective, see (Craig, 1987a)). Opportunistic control is basically the strategy which selects the best option available, and is sensitive to the state of the problem-solving process and the available knowledge. Most blackboard systems implemented to date have incorporated an opportunistic element into their control processes in an attempt to promote faster convergence on a solution: the OPM system (Hayes-Roth, 1979), indeed, was based on an opportunistic model of human planning behaviour. HEARSAY-II's second, opportunistic, phase was designed so that the best possible use was made of available speech signal hypotheses.

The scheduler in a blackboard system may be implemented in a variety of ways. In the HEARSAY-II system, it was implemented as a set of procedures in the system's implementation language. Since that time (1976), different approaches have been adopted, all with the aim of increasing the scheduler's flexibil-

ity. The AGE blackboard shell (Nii, 1979) employs generic scheduling components which are sensitive to various types of events. The schedulers which can be constructed using AGE combine pieces of LISP code with control rules. AGE scheduling is commonly referred to as *event* scheduling.

The second main approach to scheduling is *knowledge* scheduling which considers the control problem as the selection of the best item of knowledge to apply to the emerging solution; event scheduling views the problem as being how best to handle an event. Knowledge scheduling in the BB1 system (Hayes-Roth, 1984; Hayes-Roth, 1986) is used as the basis for a powerful theory of control called the Blackboard Control Model (Hayes-Roth, 1985). The Blackboard Control Model considers control to be another domain in which to solve problems and views the control process as a variety of planning. The Control Model explicitly represents control decisions on a second blackboard: this has the advantage that explicit reasoning processes can be employed to control a system's activities. Because there is an explicit representation of the decisions made in controlling a system, the Control Model is considerably more flexible than other proposals. The power of this model derives from the fact that the control blackboard is responsible for controlling its own activities: that is, control is reflexive. The NBB blackboard system shell (Craig, 1987c) used to construct the CONFER system uses the Control Model as a way of making explicit its control decisions for greater transparency of operation.

The blackboard architecture has been successfully employed in a number of systems. Many of the systems have solved problems in signal processing (HEARSAY-II and HASP/SIAP, for example) and protein chemistry (PROTEAN (Hayes-Roth, 1984; Hayes-Roth, 1986) and CHRYSALIS (Terry, 1983), for example). The architecture is being applied in other domains also, for example, in the construction site layout domain (Tommelein, 1986). Its power and flexibility single it out at present as the main candidate for application in complex problem domains.

4. EXPLANATION

Within AI there are many systems which claim to incorporate explanation facilities. This section briefly presents an alternative approach, based upon a different definition of what constitutes an explanation.

Instead of explaining, most systems engage only in justification of their reasoning or action (e.g., Davis, 1982). This is achieved by presenting the reasoning steps the system has performed in terms of the KSs that have been applied in deriving a solution or sub-goal. Mere statement of the steps taken by a system in solving a problem does not constitute an explanation of *why* those steps were taken, only that those particular steps *were* taken, and that each was justified in the circumstances (in that it legally followed from the previous state). It is left to the user to decide why a particular step was selected in preference to others, and to decide the relative importance of each step in achieving the goal being examined.

4.1. The Goal of Explanation

The goal of explanation is the arrival at a state of mutual understanding between agents engaged in a dialogue. In the case of system explanation, this translates into a goal of describing a reasoning sequence or process in such a way that the user attains *understanding* of that reasoning Such a task encompasses more than just a statement of the problem solving steps. It requires that general knowledge be incorporated, and that the problem be viewed at different levels of abstraction, so that users can interrogate the system until satisfied that they understand why a particular decision was made. Systems that simply replay the order of rule application cannot achieve the primary goal of elaboration to transfer understanding.

One reason for the limited form of "explanation" in currently implemented systems is that "explanation" tends to be implemented by an extra module added to the functioning problem solving framework. In this situation, development of explanation facilities is constrained by the inference engine and rule structure that has been adopted. For adequate explanation to be produced by a system, it should be considered as a primary aspect from the first stages of prototype design. A system that intends to incorporate explanation should be *designed* to support explanation (Clancey, 1983; Swartout, 1983; Neches, 1985), not treat it as an additional feature that can only be implemented in a restricted form because of the imposed system structure. Explanation is a fundamental aspect of AI systems, particularly those that are aimed at applications involving human interaction. Good explanation incorporates both justification so that an expert can verify that the correct decision has been taken, and elaboration so that others can understand why particular decisions were taken and how those steps relate to the problem as a whole. Only a system that has been specifically designed for explanation can hope to satisfy the goal of promoting user understanding.

4.2. Explanation in CONFER

In order to perform explanation, a variety of data must be accessed, including data stored on the blackboard. The data stored on the blackboard during the problem solving process results from KS execution. The knowledge in the KSs is in a form which cannot be explicitly accessed. In order to achieve explanation, it is necessary to incorporate additional knowledge to relax this constraint. CONFER therefore incorporates two additional representations of knowledge required by the explanation process: a declarative knowledge base and a set of process models (these will be described below).

Given the three kinds of knowledge available to CONFER (knowledge applied during problem-solving, general knowledge and process models), explanation facilities can be provided which can account for the decisions made by the system in terms of the knowledge directly applied, together with other relevant information and knowledge of the process as a whole. The form of explanation provided will depend on the needs of the user. Initially the explanation form will be of a "traditional", rule-oriented form that allows for justification of system action. With the additional kinds of knowledge available, users will have the facility to investigate aspects of this explanation until they achieve a level of description they can understand. Alternatively, users may enquire as to the role played by a particular decision or item of knowledge in the problem. It is proposed that these facilities will enable CONFER to:

- focus on a particular concept for explanation, and
- abstract away from a concept and view it in terms of the whole process.

The incorporation of diverse knowledge structures will assist CONFER in approaching the goal of providing explanations that promote understanding in a manner not so far implemented in other systems.

5. CONFER SYSTEM STRUCTURE

The CONFER system is designed to be an accountable process control system based on the blackboard architecture. The system has four main components, two of which are designed to improve the performance of the problem-solving system as well as to support explanation. The components are:

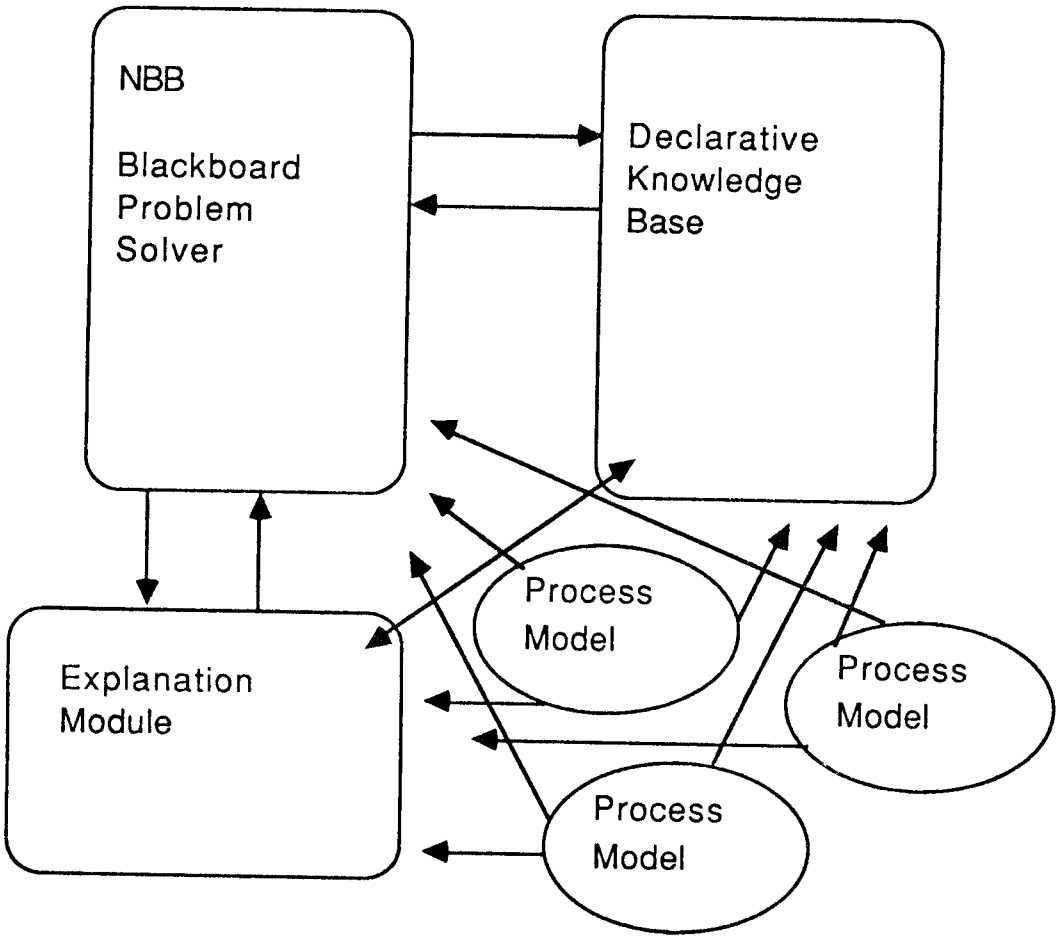

Figure 5. The Architecture of CONFER

- the blackboard problem-solving system;
- a declarative knowledge base;
- process models, and
- an explanation sub-system.

The components and their relationships are shown in figure 5.

We view CONFER as being explanation-driven. That is, the ability of the system to justify and explain its actions in a meaningful way is the most crucial factor in the design. In order to make the system accountable in this way, its components must be given access to explicit representations of problem domain knowledge -- both procedurally encoded as Knowledge Sources in the blackboard problem-solver, and declaratively represented in the knowledge base -- as well as to control strategies. As a consequence of this design decision, the system includes a declarative knowledge base and a set of process models.

5.1. The Problem-solver

The role of the blackboard problem-solver is to perform monitoring and diagnosis tasks in addition to controlling the fermentation equipment. The blackboard recieves input from sensors attached to the fermentation vessel and from a mass spectrometer. This information is used to infer the state of the process in real time. The blackboard contains four abstraction levels in its domain panel: the levels are concerned with the input data, the general state of the fermentation process in terms of abstracted data and the behaviour of the culture at the cellular and molecular levels. The inclusion of cellular and molecular levels permits the system to make detailed hypotheses about the state of the process: in particular, the system can infer the kinds of microbiological processes responsible for observed behaviours. Furthermore, it allows the system to determine the potential effects of hypothesised control decisions which change the state of the fermentation process.

In addition to reasoning about the fermentation process, the problem-solver reasons about its own behaviour. The problem-solver is controlled by a version of the Blackboard Control Model (Hayes-Roth, 1985) which explicitly represents all control decisions at a variety of abstraction levels. The explicit representation of control makes it the subject of a reasoning process and also contributes to the construction

of a system with greatly increased flexibility. The explicit representation of control assists in explanation and justification. Since the problem-solver is able to reason about and record each control decision it has made, it is able to determine, at a later time, precisely *why* it made that decision. When justifying a particular process control decision, the blackboard problem-solver is able to examine the circumstances under which that decision was made. The task of explaining why a particular decision was made and of explaining the background to it is also eased because the blackboard records the state of the control problem as well as the process problem: this information is available for inspection at all times.

5.2. Declarative Knowledge Base

In addition to the explicit representation of control information, a declarative representation is also maintained. The declarative knowledge base contains static information about microbiological objects (e.g., DNA) and concepts (e.g., transcription): this is the kind of background knowledge possessed by human practitioners and forms a general context within which they operate.*

The declarative knowledge base also contains a dynamic component derived from the operations of the problem-solver. As part of the knowledge base, there is a representation of the kinds of actions which may be taken by Knowledge Sources and the relationships they have to blackboard events and states. This information can be used by the blackboard to extend its problem-solving capabilities and to reason about its own actions at a deeper level. However, the dynamic component of the knowledge base also records information which can be of use in explanation -- explanation for humans *as well as* for the problem-solver.**

By anchoring problem-solving decisions in background knowledge, a rich context is provided within which to do explanation. We have conjectured that explanation depends, in part, upon the ability to elaborate concepts and action descriptions. This ability requires *a priori* that relevant background information be available: the declarative knowledge base provides this information.

---

* The knowledge base is being developed by Catherine R. Betts.

** The latter is an issue we have not yet adequately explored.

5.3. Process Models

The system contains a number of process models. Process models are used in a variety of ways by CONFER -- they are used to assist explanations of micro- and macro-biological processes and to assist problem-solving Knowledge Sources in the process control and monitoring activities. Although we have not yet worked out the full details of the models and their structure, we believe that they will have dual declarative/procedural representations. The procedural representation corresponds to a fairly conventional Qualitative Reasoning approach to processes (Bobrow, 1984).

Using a procedural representation, a biological process can be simulated to determine its behaviour under different conditions -- this is useful in detailed reasoning about the possible effects of a decision to alter one or more of the fermentation process' parameters. The procedural representation is also useful in attempting to determine what could be the cause of some observed effect. When queried by the user, the procedural representation can be run to provide information to the explanation sub-system, in effect animating the process for the purposes of explicating it to the user. The procedural models used by CONFER are incremental in the sense that they can be started from any state and resumed after results have been extracted. The need for incremental models is dictated by the requirements of the blackboard problem-solver.

The declarative representation of biological processes is expected to serve a role very similar to the declarative knowledge base. That is, the declarative process representation permits inferences to be made at any time without the need to execute a process. This entails that the causal links between steps in each process can be examined and the consequences seen at any time. The declarative process model also permits examination of the process structure. This appears to be particularly valuable when establishing the states through which a process must pass in order to reach some goal state. The goal state might be suggested, for example, by a problem-solving Knowledge Source involved in setting fermentation process parameters; alternatively, the user might cause a goal state to be set during an explanation session by asking a question about what happens to the process when the fermenter is in a given state.

Currently, we are examining a process model for the *lac* operon. This model is fundamentally important to the system because it is the basic mechanism driving the fermentation process under examination.

We anticipate the need for other models, e.g., a cell growth model and the two compartment model of cellular metabolism. The interactions between the various models remain an interesting open question at this time.

5.4. Explanation

CONFER's final component is the explanation sub-system. In the above description, the role of explanation has been stressed as a major design factor. It is the explanation sub-system which is finally the consumer of the information held in or derived by the system and is the least understood of CONFER's four modules. The explanation sub-system has two basic roles: the first is to provide justifications, the second is to provide explanations. The justification task can be performed in a conventional manner by tracing the decisions made by the blackboard problem-solver and augmenting the trace with information about Knowledge Sources. The explanation component is concerned with using the background information provided by the knowledge base and process models to augment the user's understanding of the process and its state.

The blackboard problem-solver maintains all problem solving steps it has taken in its global database, enabling the explanation module later to examine the state of the problem when a particular decision was made. In addition to the domain knowledge required to solve a problem, the blackboard also maintains all relevant control knowledge concerning how the problem was solved. The levels of abstraction within the blackboard architecture facilitate explanations of varying granularities. In addition to information held on the blackboard, the explanation sub-system makes use of knowledge stored in the declarative knowledge base and the results of executing the process models.

6. CONCLUSIONS

This paper has described the CONFER system. CONFER is a real-time problem-solving system for the monitoring and control of fermentation processes. The process CONFER is designed to control is the production of β-galactosidase from a mixed medium. The system has been designed with accountability as a primary goal. As a consequence of this decision, explanation is seen as a crucial feature and is supported by a wide variety of procedural and declarative knowledge. The extensive knowledge possessed by the sys-

tem is required to support it in the wide variety of tasks it must perform. The system is, we believe, the first of its kind insofar as it embodies considerably more knowledge, in more varied forms, than in most previous AI systems.

REFERENCES

(Bobrow, 1984) Bobrow, D.G. Qualitative Reasoning about Physical Systems: An Introduction, Artificial Intelligence Journal, Vol. 24, pp. 1 - 5, 1984

(Clancey, 1983) Clancey, W.J. The Epistemology of a Rule-Based Expert System -- A Framework for Explanation, Artificial Intelligence, Vol. 20, pp. 215 - 251, 1983

(Craig, 1987a) Craig, I.D. The Blackboard Architecture: A Definition and Its Implications, Research Report RR98, Department of Computer Science, University of Warwick, March, 1987

(Craig, 1987c) Craig, I.D. The BB-SR System, Research Report RR94, Department of Computer Science, University of Warwick, February, 1987

(Craig, 1987b) Craig, I.D. The Blackboard Architecture: Example Systems, Research Report RR101, Department of Computer Science, University of Warwick, May, 1987

(Darnell, 1986) Darnell J., Lodish H. and Baltimore D., Molecular Cell Biology, Scientific American Books, New York, 1986

(Davis, 1982) Davis,R. Teiresias: Applications of Meta-Level Knowledge, in Davis, R. and Lenat, D.B., Knowledge-Based Systems in Artificial Intelligence, McGraw-Hill, New York, 1982

(Erman, 1975) Erman, L.D. and Lesser, V.R., A Multi-level Organisation for Problem Solving Using Many, Diverse, Cooperating Sources of Knowledge. Proc. IJCAI 4, Vol. 2, pp. 483 - 490, 1975

(Erman, 1980) L.D. Erman, F. Hayes-Roth, V.R. Lesser and D.R. Reddy. The HEARSAY-II speech-understanding system: Integrating knowledge to resolve uncertainty. Computing Surveys, Vol. 12, pp. 213-253, 1980

(Feigenbaum, 1978) Feigenbaum, E. and Nii, H.P., Rule-based Understanding of Signals in Pattern -Directed Inference Systems, D.A. Waterman and F. Hayes-Roth (Eds.), Academic Press, 1978

(Hayes-Roth, 1977) Hayes-Roth, F. and Lesser, V.R., Focus of Attention in the Hearsay-II speech understanding system. Proc. IJCAI 5, pp. 27 - 35, 1977

(Hayes-Roth, 1979) F. Hayes-Roth and B. Hayes-Roth. A Cognitive Model of Planning. Cognitive Science, Vol. 3, pp. 275-310, 1979.

(Hayes-Roth, 1983) Hayes-Roth, B. The Blackboard Architecture: A General Framework for Problem Solving? Report No. HPP-83-30, Heuristic Programming Project, Computer Science Dept., Stanford University, Palo Alto, CA, May 1983

(Hayes-Roth, 1984) B. Hayes-Roth. BB1: An Architecture for Blackboard Systems that Control, Explain and Learn about their own Behavior. Technincal Report HPP-84-16, Computer Science Department, Stanford University, 1984.

(Hayes-Roth, 1985) B. Hayes-Roth. A Blackboard Architecture for Control. Artificial Intelligence Journal, Vol. 26, pp. 251-321, 1985.

(Hayes-Roth, 1986) B. Hayes-Roth, A. Garvey, M.V. Johnson Jr., and M. Hewett. A Layered Environment for Reasoning about Action. Report KSL 86-38, Knowledge Systems Laboratory, Computer Science Department, Stanford University, 1986.

(Jacob, 1961) Jacob F. and Monod J. Genetic Regulatory Mechanisms in the Synthesis of Proteins, Journal of Molecular Biology, Vol. 3, 1961

(Neches, 1985) Neches, R., Swartout, W.R. and Moore, J. Explainable (and Maintainable) Expert Systems, Proc. IJCAI-85, pp. 382 - 389, 1985

(Newell, 1973) Newell, A. Production systems: models of control structures. In W.G. Chase (ed.), Visual Information Processing, Academic Press, New York, 1973

(Nii, 1979) H.P. Nii and N. Aiello. AGE: A knowledge-based program for building knowledge-based programs. Proc. IJCAI 6, pp. 645-655.

(Nii, 1982) H.P. Nii, E.A. Feigenbaum, J.J. Anton and A.J. Rockmore. Signal-to-symbol Transformation: HASP/SIAP Case Study. AI Magazine, Vol 3, pp. 23-35, 1982.

(Nii, 1986a) H.P. Nii. The Blackboard Model of Problem Solving. AI Magazine, Vol. 7, pp. 38-53, 1986.

(Nii, 1986b) H.P. Nii. Blackboard Systems Part Two: Blackboard Application Systems. AI Magazine, Vol. 7, pp. 82-106, 1986.

(Stent, 1971) Stent G.S., Molecular Genetics: An Introductory Narrative, W.H.Freeman and Co., San Francisco, 1971

(Swartout, 1983) Swartout, W.R. EXPLAIN: A System for Creating and Explaining Expert Consulting Programs, Artificial Intelligence, Vol. 21, pp. 285 - 325, 1983

(Terry, 1983) A. Terry. The CHRYSALIS Project: Hierarchical Control of Production Systems. Memo HPP-83-19, Computer Science Department, Stanford University, 1983.

(Tommelein, 1986) I. Tommelein, V. Johnson, R. Levitt and B. Hayes-Roth. A prototype of the SIGHTPLAN system for the design of construction site layouts. Technical Report, Computer Science Department, Stanford University, 1986.