DECADE
A Hybrid Expert System For Catalyst Selection
Parti: Expert System Considerations

by

R. Banares-Alcantara, A. W. Westerberg, E. I. Ko, M. D. Rychener

EDRC-06-12-86^

September 1986

DECADE

A Hybrid Expert System for Catalyst Selection

Parti: Expert System Considerations

by

Ren6 Bafiares-Alcántara [*] [**] [***]

Arthur W. Westerberg*

Edmond I. Ko*

Michael D. Rychener**

[*]*Department of Chemical Engineering,*

***Design Research Center*

Carnegie-Mellon University

Pittsburgh, PA 15213

[***]*Currently at the Instituto de Investigaciones Electricas,*

Interior Internado Palmira; Apartado Postal 475

Cuernavaca, Morelos. MEXICO.

Table of Contents

Abstract 1
1. Introduction 1
1.1. Motivation 1
1.2. Overview 3
2. Background on catalyst selection 4
2.1. The catalyst selection problem 4
2.2. The Fischer-Tropsch reaction 6
3. Background on Hybrid Knowledge-Based systems 8
3.1. Knowledge Representation 9
3.1.1. Rule-based representation 11
3.1.2. Frame-based representation 11
3.1.3. Procedural representation 12
3.2. Problem-Solving Methods 13
3.2.1. Depth-first search 14
3.2.2. Means-Ends analysis 15
3.3. Knowledge Abstraction 17
3.4. Implementation 18
3.4.1. LISP 19
3.4.2. OPS5 20
3.4.3. SRL 21
3.5. Conclusions 22
4. DECADE as a hybrid knowledge-based system 22
4.1. Blackboard Model Structure 23
4.1.1. The Blackboard 23
4.1.2. The Knowledge Sources 24
4.1.3. The Context 26
4.2. Blackboard Model Control 26
4.2.1. Overall control 26
4.2.2. Internal control in the knowledge sources 28
4.3. Using DECADE 29
4.4. Conclusions 30
5. **Future** Work and Conclusions 31
5.1. Links between symbolic and numerical languages 32
5.2. Conclusions 33

List of Figures

Figure 1-1: Dichotomy of approaches in the design process 2
Figure 2-1: Some of the products derived from the direct processes of synthesis gas 7
Figure 3-1: Example of a production rule 11
Figure 3-2: Example of a frame 12
Figure 3-3: Example of a numerical procedure (mathematical version). 12
Figure 3-4: Depth-first search applied to the classification of the reaction producing ethane 15
Figure 3-5: Means-Ends analysis table for the prediction of mechanism for alkanes 16
Figure 3-6: Means-Ends analysis applied to the methanation reaction 17
Figure 3-7: Example of a numerical Lisp function 19
Figure 3-8: Example of an algorithmic Lisp function 20
Figure 3-9: Example of an OPS5 **rule** 21
Figure 3-10: Example of an SRL schema 22
Figure 4-1: General Structure of a Blackboard 23
Figure 4-2: Simplified version of the overall **control in DECADE** 27
Figure 4-3: Simplified version of the control inside **the knowledge sources** 29

List of Tables

Table 4-1: DECADE as an example of a Blackboard Model. 25
Table 4-2: Results for different reactions at the same level of abstraction. 30
Table 5-1: Four ways to link programs written in different languages. 32

Abstract

DECADE (Design Expert for CAtalyst DEvelopment) is a prototype expert system for catalyst selection. The objective of DECADE'S development has been to investigate and evaluate the potential of expert system's technology applied to the solution of chemical engineering problems. DECADE'S particular application problem consists of prescribing a set of catalytic materials that have an acceptable probability of being appropriate for a target reaction. The class of reactions for which DECADE has specific knowledge is carbon monoxide hydrogenation.

Given DECADE'S architecture and implementation, it can illustrate the integration of different paradigms along some of the several dimensions of expert systems applications: knowledge representation, problem-solving methods, and levels of knowledge abstraction. All these properties are achieved through the use of different languages (FranzLisp, OPS5, SRL1.5) brought together in a blackboard model architecture.

1. Introduction

1.1. Motivation

Many of the tasks encountered in the engineering practice cannot be completely articulated into a well-formed algorithmic procedure. They are a combination of a variety of special-purpose heuristic problem-solving methods and traditional algorithmic parts, scheduled in different sequences depending on the particular problem.

With the advent of computers those tasks that have been amenable to formalization into a mathematical language have been successfully automated and are currently solved by computers, making their solution faster and more accurate. However, those tasks, that because of their nature have not been successfully reduced to an algorithmic form, have only very recently been attempted to be solved by computers. The end result is that the current approach for the solution of engineering problems has been the combination of fully automated tasks, and a set of symbolic or heuristic tasks, that are being performed "by hand" by the engineer. The sequencing and conciliation of these parts is also performed outside the computer. As seen in Figure 1-1, the four traditional subparts of the Chemical Engineering design process are a combination of encoded routines (shown in upper case), and symbolic problem-solving (in lower case). We can offer two comments regarding Figure 1-1:

- although the knowledge encoded in the numerical algorithms is necessary, it is not sufficient, and

- integrating the different parts of the design process into **a** single procedure increases the quality of the results.

Figu re 1 - 1 : Dichotomy of approaches in the design process

Advances in the area of Computer Science, and in particular Artificial Intelligence, provide the means of automating the heuristic tasks. A number of successful expert systems or knowledge-based systems have been developed for areas other than engineering, areas that do

not contain an algorithmic part. Also, many of the initial knowledge-based systems dealt with simplified tasks, that allowed the use of an exclusive representation and problem-solving method to solve the problem. Unfortunately (for engineering), these initial programs lacked facilities to reflect the rich variety of knowledge necessary to solve engineering problems.

As the Expert System area has matured, it has been possible to start proposing architectures to solve engineering design problems. It is one of the goals of this paper to convince the reader that in order to do so, a hybrid knowledge-based system is convenient if not necessary. There is a budding set of hybrid architectures being developed and tested. This paper investigates one possible way to implement a hybrid system: the Blackboard model architecture.

1.2. Overview

Having introduced the nature of the areas that are touched by this study and stated the objectives of the work, the rest of the document is organized as follows:

The second section attempts to provide background on the catalysis area, especially on the process of catalyst selection. The treatment will be focused on the hydrogenation of carbon monoxide, which is the specific area of knowledge of DECADE [Bañares-Alcántara 86].

The third section also provides background to the reader, in this case about hybrid knowledge-based systems. Hybridity can be attained in different areas, and this section characterizes those areas and investigates their relevance to the design process.

In section four DECADE is presented as an example of a blackboard model. DECADE is a hybrid knowledge-based system for catalyst selection, and its structure and control are reviewed in light of the blackboard architecture characteristics.

The ending section will summarize the most important facts. From these facts, a set of conclusions is drawn, and the contributions of the work can be weighed. Also, some recommendations for future work in this and related areas are given.

2. Background on catalyst selection

The selection of a catalyst has a major impact on the economics of chemical processes because the catalyst affects the feasibility and the degree of conversion of raw materials to final products, and generally raw material and product costs dominate the total cash flow of a process. From the point of view of the design process, it is important to realize that not only the reactor, but the totality of the plant, are designed taking into account in a direct or indirect way the characteristics of the reaction (conditions of temperature and pressure in which it must **run,** side products, conversion).

Selecting a catalyst is not an easy task since there is little information on:

- **which are all the properties of a catalytic material that affect the characteristics of a reaction**
- **how each of these properties affect the characteristics**
- **the interactions that the components of combined catalysts (catalysts with more than one component) have on each other in relation to the reaction that they are catalyzing**

2.1. The catalyst selection problem

The catalyst selection problem is a very complex and inexact activity, based to a large extent on **experience. Furthermore, the underlying theory for catalyst selection is not complete enough** to **permit the prediction of a unique, complete and certain answer.**

The selection of a catalyst is a problem that is currently solved only by a relatively small number of **experts interacting in a consultation environment with the user. It is prone to decomposition into smaller and very different subproblems, some of them amenable to algorithmic solution,** but the **majority only solvable through the use of heuristic reasoning given their lack of formalization. Also, since the order of execution of the subproblems is not fixed, but varies greatly depending** on the **characteristics of the individual problem, a flexible solution strategy is needed. The solution** to the **overall problem should not only preserve the functionality (proper selection of catalysts), but** the form **(interactive environment with the user) as well.**

Some attempts have been made to formalize the process of catalyst selection, among which the one **described in the book** *"Design of Industrial Catalyst"* **([Trimm 80]) is a good example. The** *"scientific basis of design"* **of catalysts described in the book, and other methodologies** contained in some other **publications, although different, coincide in prescribing a number of subtasks that are** useful to **perform when selecting a catalyst (for the other methodologies see for example the section** *"Catalyst Selection"* **in [Klier 82]). An enumeration of the subtasks follows:**

• Stoichiometric analysis

Write down all the reactions that come from all possible combinations of the reactants and products of the target reaction, incorporating only chemically stable compounds, and without making reference to the reactions en the surface. This task is akin to the one solved in the Ph.D. thesis of R.B. Agnihotri ([Agnihotri 78]).

A simplification of this step consists of listing only the target reaction, the reactions **producing useful or** acceptable side products, and the reactions **that** need **to be inhibited because they produce** unacceptable side products.

• Thermodynamical analysis

Calculate the Gibbs free energy of the listed reactions in order to identify those which are (thermodynamically) feasible. Calculate the equilibrium conversions.

Calculation of the enthalpy of reaction is also useful in terms of the thermal stability required from the catalytic material, and for heat transfer calculations.

• Literature Search

One step that is consistently stressed is the literature search. As a matter of method, it is advised to search for all available information about the target reaction, analogous reactions, and data like activity patterns, heats of adsorption, proposed mechanisms, observed intermediates, etc. The search for general data should be done prior to the selection process (this practice can prune the search space considerably), while search for very specific information can be done whenever it is needed.

• Classification of reactions

It is convenient to group the reactions listed in the stoichiometric analysis in terms of their class. The list of possible reactions may be very large, but the list of classes of reactions is considerably smaller. This is important because many of the heuristics are given in terms of the classes of reactions rather than reactions themselves *(e.g.* the *activity patterns).*

• Identify types of chemical bond rearrangements occurring in each reaction.

Although this step is not explicitly mentioned in some of the methodologies, it is consistently used as the basis for the proposal of surface steps whenever there is no experimental data available.

• Proposal of a surface mechanism

None of the methods have a formalized strategy for the proposal of the surface steps, all of them either extract them from the literature, or obscurely propose them using a *priori* **knowledge. This step many times could be considered as the basis of the design, in that the information conveyed from it supplies pointers to many alternative methods of enumerating and ranking catalytic materials.**

!t is worth mentioning that a mathematical method for the enumeration of all possible mechanisms has been proposed ([Happel&Sellers 83]). Its inputs are the possible intermediate species and elementary surface steps that the user wants to consider, even though it is often necessary to obtain such data from the literature. The availability of the surface data is assumed, when many limes such data are uncertain and difficult **to** find.

- Reaction path identification and preliminary catalyst material selection

From the data obtained in the mechanism prediction, those steps **that have to be favored,** and those steps that have to be inhibited are identified. This **will produce a list of** requirements to demand **from** the materials **that will be catalysts.**

- **Experimental testing**

Sometimes, during the process of selection, a set of experiments is proposed. Such experiments have the purpose of either obtaining missing data or studying the behavior of a partial solution *(e.g.* study the interactions between the different components of a catalyst).

Nevertheless there are subparts of the problem that not only are far from being formalized, but also are even lacking a consensus of which methodology to follow. Take for example the specification of the problem. There is no information on what is considered enough input information for the **prediction of catalysts. In the method proposed by Trimm, the input information is the** desired **product. A combination of other data like the available raw materials or the ranges of operating conditions can also form part of the input, but there is no clear idea on what is the minimum amount of data needed. As a rule of thumb, the more information that is available from the user, the easier it is to prescribe a catalyst. Data are not always available though.**

2.2. The Fischer-Tropsch reaction

The knowledge in DECADE has been focused to a single reaction: the Fischer-Tropsch reaction **(for more information about this reaction, consult the book written by Anderson [Anderson&KR 84], the monograph by Dry [Dry 81], or the short article by Haggin [Haggin 81]). While constraining the area of knowledge reduces the search space in size, we think that it maintains the important characteristics of the domain. The Fischer-Tropsch reaction is thought to be a representative reaction in terms of catalyst selection, given the fact that enough studies have been made about it, but no one can claim to understand it perfectly well, leaving room for the application of knowledge-based systems.**

The Fischer-Tropsch reaction is named after two German chemists: Franz Fischer and Hans **Tropsch, who, in 1926, described it for the first time. The Fischer-Tropsch reaction** can be

considered the main reaction of C_1 chemistry. It can be described as the production of hydrocarbon and oxygenated organic molecules via the reaction of carbon monoxide and hydrogen. The mixture of carbon monoxide and hydrogen is known as *synthesis gas* or *syngas*. The molecules so produced have predominantly straight carbon chains, at least in the C_4—C_7 range ([Haggin 81]).

The range of subreactions possible when using carbon monoxide and hydrogen as reactants falls into three divisions ([King&CG 81]):

1. Direct process starting with syngas (see Figure 2-1),
2. Indirect process starting with methanol[1] or methanol mixed with syngas, and
3. Indirect process by combining a third molecule with syngas or methanol.

Only the first division is of specific interest to this study.

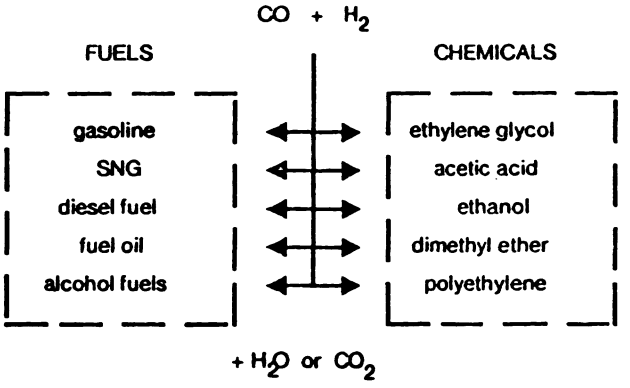


Figure 2-1: Some of the products derived from the direct processes of synthesis gas

All the Fischer-Tropsch reactions are exothermic, and produce water as a side product (at certain conditions, a reaction known as water-gas shift { $CO + H_2O \rightarrow H_2 + CO_2$ } may change the overall side product from water to carbon dioxide). One other side reaction is the decomposition of carbon monoxide ({ $2\,CO \rightarrow CO_2 + C$ }; also known as the *Boudouard reaction*).

Given a set of starting conditions for the Fischer-Tropsch process, the Schultz-Flory equation predicts the proportion of C_1 products to higher carbon-number products. The product distribution depends on such variables as the catalytic material, reaction temperature and pressure, and feed gas composition. However, only methane and methanol can be produced with a 100% selectivity.

---

[1] Note that in this instance methanol is a derived product of the hydrogenation of carbon monoxide.

Several mechanisms have been proposed over the years, but no single mechanism is applicable to all catalytic surfaces. A very complete review can be found in the work of Rofer-OePoorter [Rofer-DePoorter 81]. A mechanism worth mentioning for its simplicity and generality is the one proposed by Bell [Bell 81], for the formation of hydrocarbons over Group VIII metals.

3- Background on Hybrid Knowledge-Based systems

Among the problems that are well suited for hybrid implementations are design problems (in particular in engineering). The justification for the need of hybrid systems is that, since engineering knowledge is heterogeneous (in terms of the kinds of problems that it encompasses and the methods used to solve them), then the use of heterogeneous representations is natural. Another important factor to take into account is the fact that some programs that solve a part of a given problem may already exist, and it would not be feasible or convenient to rewrite them into another format only to make them compatible with the overall system.

A system may be hybrid in several ways. Attempts to characterize this hybridization have resulted in the following classification of the properties that may be of importance when constructing a knowledge-based system:

1. Knowledge Representation

   Deals with what is known in traditional numerical applications as the *database,* and in knowledge-based systems as the *knowledge base.* It can be considered as the description of the problem space, its properties and internal laws.

2. Problem-Solving Strategy

   Encompasses the set of methods that can be used when attacking a problem. The methods describe how to manage the available information and how to obtain the missing information in order to achieve a goal state. In terms of the problem space, problem-solving methods prescribe the way in which one should move from the initial state to the solution state passing through the partial solution states.

3. Knowledge Abstraction

   It is a common method of decomposing the problem into subproblems (in a more general way it could be considered as one particular problem-solving strategy). More abstract representations hold less information about the problem but are easier to work with. A solution in an abstract level can be used as a guide for the search of the solution of a less abstract one.

4. Implementation

There are a large number of languages, operating systems, nnri processors that can be used when programming. There are relations among these items though. Only a limited number of languages are supported in a given operating system. In turn, a particular processor can run only a single operating system at a given time.

Note that the factors enumerated above are not completely independent, but were separated only in order to ease the characterization of a system. For example, a choice of problem-solving strategy may influence the choice of knowledge representation, and, in turn, these two can determine the implementation used. In some other cases the implementation tools limit the choice of the rest of the factors.

3.1. Knowledge Representation

Knowledge has different forms. It can be certain or uncertain, formalized or unformalized, **structured or unrelated, etc. It can be found in formulas, tables, statements, traditional practices** or **embedded in methodologies, but, when it has to be translated in such a way that it can be** stored and **used by a computer, a knowledge representation mechanism has to be chosen. The issues that** affect **the selection of knowledge representation are the naturalness of representation, efficiency of** storage **and manipulation, and consistency and compatibility with the rest of the representations in** the **system.**

There is no consensus on how to classify the different classes of knowledge representation. The **following is one of the ways in which it can be classified:**

1. **Production rules**

   Production rules or IF-THEN statements consist of a conditional part and an action part. The conditions of a rule have to be satisfied in order for the rule to act. The actions of a rule execute a series of operations that will modify the state of the problem. *Demons* **and** *active values* (**[Kunz&KW 84]) are special cases of productions rules.**

   Production rules are prescribed for the representation of control mechanisms, problem-solving strategies, heuristics, and in general any kind of knowledge that is applicable only when a given context is present.

   A reference to the usage and advantages of production systems is [Brownston&FKM 85], while [Hayes-Roth 85] is a good introductory document.

2. **Frames**

   A frame is a structure that represents a concept. It can have any number of **attributes** or **properties attached to it, some of the properties can be relationships. An attribute** may **have any number of values (i.e. no value, one value, several values). The importance** of **being able to represent relations is that a given frame can inherit properties** (attributes **and/or values) from the frames to which it is related. In reality the frame concept is**

considered to be a special case of another representation structure: the semantic networks. Semantic networks are used mostly in Natural Language research, and, in general, frames are appropriate for knowledge-based applications.

Frames are a convenient and natural way to represent descriptive information, that is, objects, their properties and their relations. They also represent very naturally the information carried in hierarchically structured domains.

For an introduction to frames consult [Fikes&Kehler 85].

3. Procedures

Procedures are probably the best known representation structure to engineers. Traditional numerical formulas map in a straightforward way into procedures. The concept is more general though, since one can think of procedures that deal with symbolic data rather than with numerical data. Sequential execution of statements, iteration and recursion are the three control schemes available to procedures.

A procedure can be used as an action of a production rule, or as the mechanism that manipulates the information contained in a frame.

4. Logic

Theorems, axioms, and any knowledge that can be formalized into a first-order logic notation can naturally be mapped into a logic representation (for an introduction see [Genesereth&Ginsberg 85]).

5. Graphics

Sometimes it is convenient to represent objects by icons in the user interface display. The icons can be manipulated. Although the icons are eventually translated into internal structures of a different nature, those structures are transparent to the user, and therefore the icons can be thought of as primitive representation objects.

6. Object oriented

Object oriented programming is actually a mixture of knowledge representation, and a style of programming. The structures used in representing knowledge are a constrained case of the more general frames in that they can be modified only by procedures within themselves known as *methods* (as defined in the object-oriented paradigm). The objects are allowed to communicate and interact only through messages.

Object-oriented programming has been most successful for simulation problems, especially where graphic displays of the dynamics are desirable.

3.1.1. Rule-based representation

As an example of a production rule we present the rule in Figure 3-1. Its purpose is to determine if the precondition *(i.e.* whether the reaction has been classified or not) for the surface mechanism prediction is satisfied. The motivation for this test is that the prediction of surface steps is restricted to certain classes of reactions.

The production rule has three conditions. The first two conditions determine whether the goal of predicting a set of surface steps is being executed by the surface steps knowledge source or **not. The third condition determines if the reaction has been classified. The symbol <reaction-name> is a variable bound to the name of the reaction for which the mechanism is going to be predicted (and therefore this rule can be applied to any reaction and not only to one alone).**

```
{{ rule: [goal: surface mechanism] argument present, precondition unsatisfied

   IF
        (1) current goal is to propose a reaction mechanism
            for a reaction with name <reaction-name>
   AND
        (2) there is a knowledge source attempting to execute the goal
   AND
        (3) the reaction with name <reaction-name> has no classification

   THEN
        (a) propose as a subgoal of the current goal to classify
            the reaction with <reaction-name>

        (b) put the current goal on hold

        (c) put the current knowledge source on hold

        (d) inform the user of this decision
```

Figu re 3-1: Example of a production rule

The production rule above has four actions. The first one is to post a subgoal of the mechanism goal: **classify the current reaction. The second and third actions put the parent goal and the knowledge source on hold (until there is information about the success or failure of the subgoal). The last** action **informs the user of the situation and the previous actions taken.**

3.1.2. Frame-based representation

Figure 3-2 contains an example of a frame. The frame in Figure 3-2 describes the object **"hydrogenolysis" as being a "class of reaction" and being of class "multimolecular". The rest of the attributes are used during the process of classification of a reaction. Any reaction that** satisfies the

{{ **frame:** [hydrogenolysisj
is a: class of reaction
is class: inul timolecul ar
constraints: { hydrogen must be a reactant
AND EITHER
there must be tv/o different kinds of products
OR
if there is only one type of product
then there must be two molecules of it }

Figure 3-2: Example of a frame

constraints contained in them (and also all the constraints of the nodes which are its **parents), is of class "hydrogenolysis". Note that the constraints, rather than properties, are statements to be satisfied.**

3.1.3. Procedural representation

Figure 3-3 shows a formula for the estimation of enthalpy, used for the calculation of AG during the **prediction of the thermodynamic feasibility of a reaction. The estimation is represented as a mathematical expression.**

$$\Delta H = \Delta H_0 + \int_{T_0}^{T} CpJT + \Delta H_{vap}$$

Figu re 3-3: Example of a numerical procedure (mathematical version).

The formula displayed in Rgure 3-3 calculates the enthalpy of a system at a temperature T as the sum **of the enthalpy at a reference temperature T_Q, the contribution of the heat capacity over** the **temperature range, and the contribution of the latent heat. If *Cp* is approximated to be:**

$$Cp = \mathbf{a} + \mathbf{b}T + \mathbf{c}\mathbf{r}^2 + \mathbf{d}\mathbf{r}^3 \qquad (1)$$

then the integral f_r CpdTbecomes

$$\int_{T_0}^{T} CpdT = \mathbf{a}(\mathbf{T} - TJ + \tfrac{1}{2}(7''^2 - 7;^2) + \tfrac{1}{3}(\mathbf{r}^3 - 7;^3) + (J^* - T_0^*) \qquad (2)$$

3.2. Problem-Solving Methods

Problem-solving is **the** process of developing a sequence of actions to achieve a goal. There are several classifications of problem-solving methods, each one springing from a different viewpoint of the area. The following list describes.the most useful:

- Newell (in [Newel! 69]) identifies two kinds of problems: the *well-structured* problems and the *ill-structured* ones. Accordingly, he reasons **that there** are **two** kinds of problem-**solving** techniques. **He** terms the **one that** deals **with ill-structured problems** *Heuristic programming.*

- **Cohen and** Feigenbaum propose a **classification for Heuristic programming. They focus the problem-solving in terms of planning, and arrive at four types of planning strategies: nonhierarchical, hierarchical, script-based and opportunistic ([Cohen&Feigenbaum 83]).**

- **Nilsson recognizes three types of problem-solving methods: state-space search methods, problem reduction search methods and predicate calculus ([Nilsson 71]).**

- **Stefik proposes a division in his prescriptive guide to building expert systems ([Stefik 82]). He starts proposing simple methods for simple problems, and as the problems grow more difficult (in terms of size, uncertainty, abstraction, etc.), he describes how given refinements on the methods can cope with the increased difficulty of the problem and give rise to a new method.**

The following classification orders appropriately the issues found in the area of engineering design. Notice that it is nothing more than a combination of the ideas exposed above.

1. Mathematical programming

Used in the solution of problems that can be represented in a mathematical format (well-structured problems). It encompasses the traditional numerical methods used in engineering.

2. Heuristic programming

Used in the solution of ill-structured problems, it can be subdivided into:

a. Nonhierarchical

The problem-solving strategy has only one level of abstraction. *Weak methods* are examples of nonhierarchical problem-solving:

i. Generate & Test
ii. Search (depth-first, breadth-first, best-first, etc.)
iii. Hill-climbing
iv. Means-ends analysis (specific case of the General Problem Solver strategy)
v. General heuristic search

b. Hierarchical

Generates a hierarchy of representations of a plan. The highest level is the simplest or most abstract, the rest are refinements of the previous one. One example of this method is the General Problem Solver proposed by Newell and Simon.

c. Script-based

It uses a skeleton plan from a library of plans (as opposed to generating it, like in the case of the hierarchical methods). Any agenda-based method is an example of this approach.

d. Opportunistic

This method has a more flexible control. The blackboard model implements this idea (more about this subject ahead).

3. Formal Methods

These methods correspond to the *logic knowledge representation* mentioned in section 3.1.

If the reader is interested in further information about the subject, we recommend [Cohen&Feigenbaum 83] as a reference. Now we present two specific examples of the use of these methods in DECADE.

3.2.1. Depth-first search

DECADE uses the depth-first search method during the process of classifying a reaction. Figure 3-4 shows the process of classification for the reaction of producing ethane. Each of the nodes in the tree is a frame of the type described in Figure 3-2. The nodes in the upper levels represent classes of reaction, those leaf nodes in the fourth level (*i.e.* ethane synthesis and propane synthesis) are instances of a reaction. Because of space limitations some of the nodes are grouped into rectangular boxes. Also note that two syngas reactions: "formaldehyde synthesis" and "methanol synthesis" are not grouped with the rest of the "fischer-tropsch" reactions. This arrangement was chosen because it reflects the fact that for these two cases the carbon monoxide bond is not broken, as opposed to the case for the rest of the syngas reactions. Indeed, this difference manifests later in the type of catalysts that these reactions require.

In order to classify a reaction, the problem reaction is tested to check if it satisfies the constraints contained in a node. If it does, the constraint satisfaction is tested recursively for each of the child nodes. In the particular case of the reaction to produce ethane, the evaluated nodes are shown in solid lines, and the satisfied nodes in bold lines. The result of the classification presents the problem reaction as of class "Fischer-Tropsch", and recognizes that it is identical with the "ethane synthesis" reaction already stored in the knowledge base.

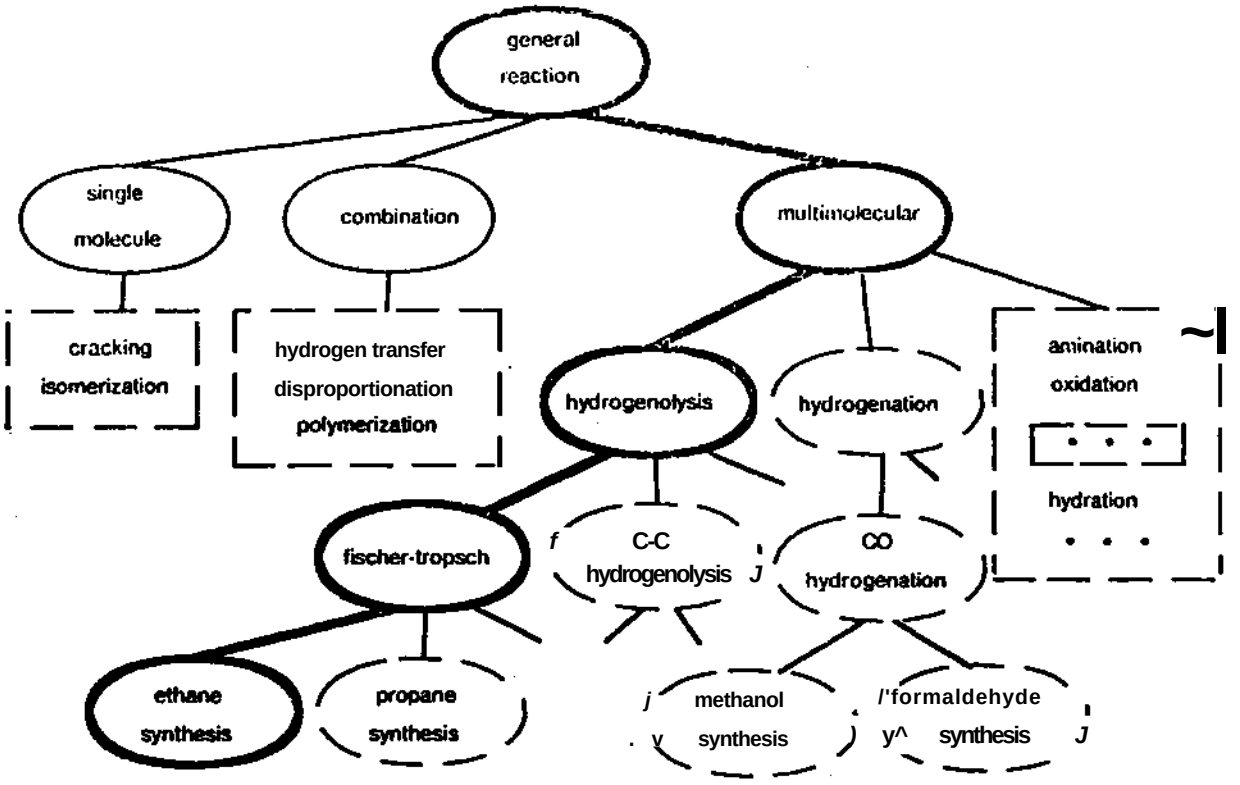
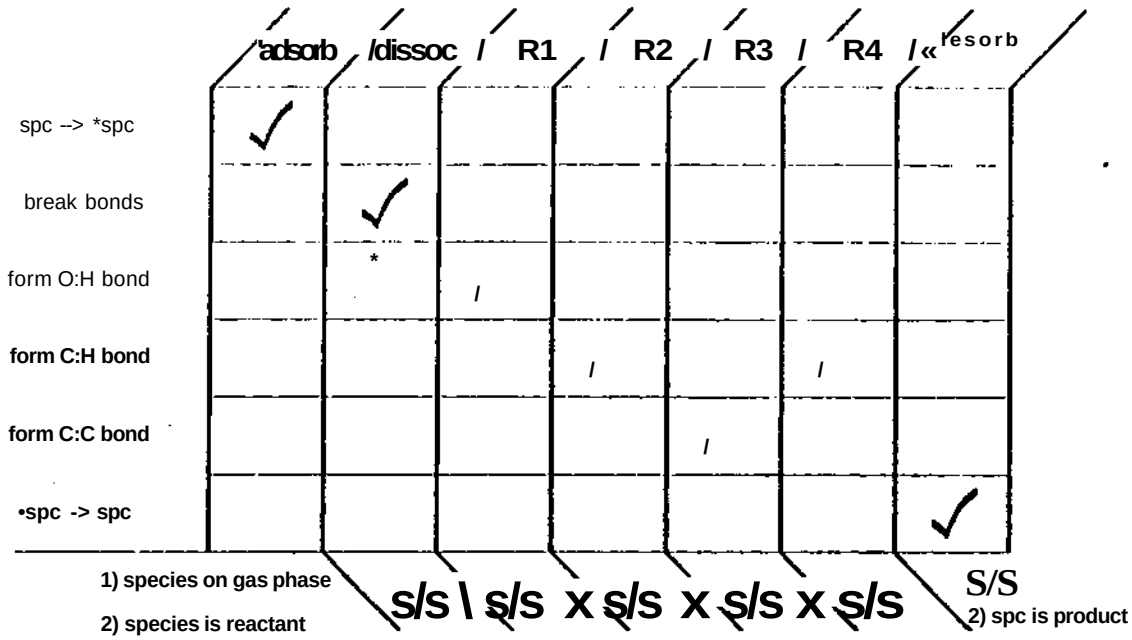


Figure 3-4: Depth-first search applied to the classification of the reaction producing ethane

3.2.2. Means-Ends analysis

The Means-Ends analysis method is used in DECADE for the proposal of reaction steps on the surface. Figure 3-5 contains the means-ends analysis table with the necessary entries to propose steps for alkane forming Fischer-Tropsch reactions (a different kind of product requires an additional set of entries). The horizontal entries represent the differences. There are two kinds of differences: the phase where the species are present (gas or solid), and the difference in bonds from one species to another. The input data consists of the set of reactants and products of the target reaction. Aided with a knowledge base containing the bonds present in different chemicals, two lists of bonds are compiled: one containing the bonds present in the reactant side of the reaction, and the other one containing the equivalent list for the product side. The objective of the process is to find the necessary steps that transform the first list into the second.

The upper vertical entries stand for the operators, for which the preconditions are given in the lower vertical part of the table. There are two kinds of operators corresponding to the two kinds of differences described above. The first type represents the physical steps that transport a species from the gas phase to the surface and *vice versa.* The second type stands for the surface steps that



Figu re 3-5: Means-Ends analysis table for the prediction of mechanism for alkanes

can break, modify or form a bond. Because of space reasons some of the entries are represented by large fonts symbols, the key to these symbols is:

- **R1:** $*OH_x + *H \rightarrow *OH_{x+1}$ **(x = 0, 1)**
- **R2:** $*CH_x + *H \rightarrow *CH_{x+1}$ **(x = 0, 1)**
- **R3:** $*CH_x + *C_yH_z \rightarrow *C_{y+1}H_{x+z}$ **(x,y>0;z>2)**
- **R4:** $*C_xH_y + *H \rightarrow *C_xH_{y+1}$ **(x > 0; y > 1)**
- **S/S:** **species are in the surface**

The first four are operators that have preconditions, which can be seen as constraints to be satisfied before the operator can execute its action. The last item is one of the preconditions.

The schematic of the action of the means-ends analysis process on the *methanation* reaction is presented in Figure 3-6. The symbols represent the species that are likely to exist in the reaction process. Symbols preceded by a ••' are adsorbed species, the others are species in the gas phase. The lines joining the symbols represent the operators needed to transform the species, and the overall flow of information goes from left to right. The list of proposed steps produced by the means-ends analysis cannot be considered as a series of mechanistic events (in the sense that there is no

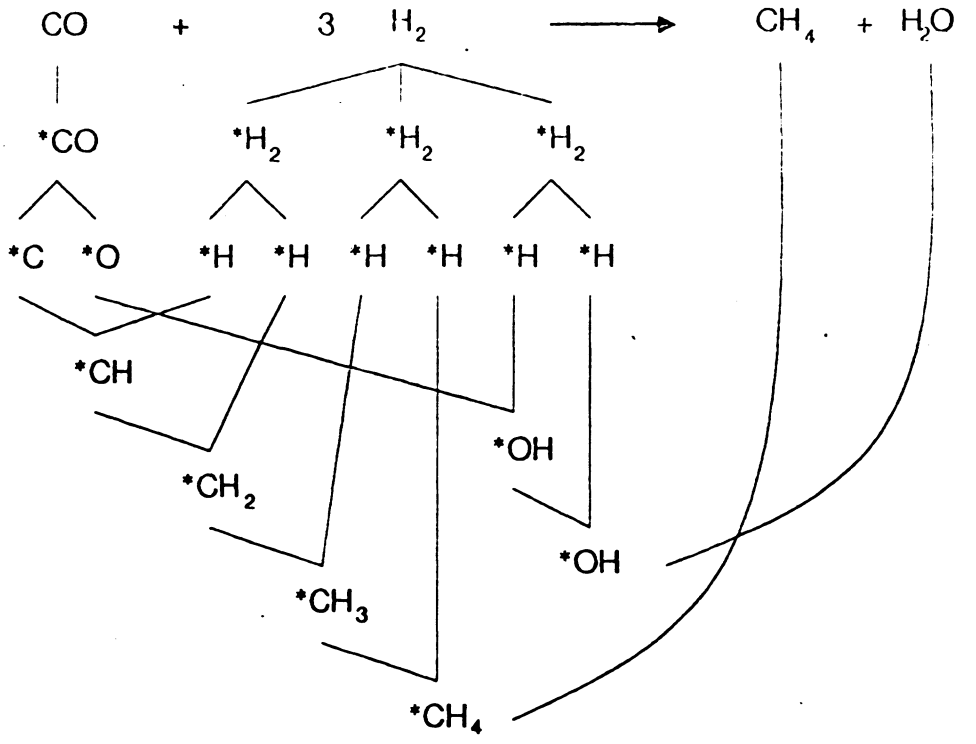


Figure 3-6: Means-Ends analysis applied to the methanation reaction

claim that the steps will occur in the exact order, or that some steps could be concerted rather than sequential). It is nevertheless useful in that it provides pointers to some of the necessary surface events that have to take place in order for the reaction to occur.

3.3. Knowledge Abstraction

The abstraction of knowledge is a useful technique for the decomposition of complex problems. It divides the problem space into levels, each level holding a different representation or view of the problem. The more abstract a level is, the simpler it is to find a solution since less information has to be managed.

There are two useful general classifications in terms of abstraction. The first one was proposed by Michie in [Michie 82], and according to it there are two kinds of knowledge:

1. The "low road" or heuristic type, represented by **[pattern → advice]**, and

2. the "high road" or causal type, represented as **[situation x actions → situation]**

Chandrasekaran and Mittal [Chandrasekaran&Mittal 83] expanded this concept, proposing the following division:

1. Table look-up

2. Partial Pattern Matching

3. Compiled structures

4. Deep structures

For DECADE, three levels of knowledge abstraction are used. As mentioned earlier, the more abstract the level is, the easier it is to solve a problem, but with less understanding of the causality.

1. Reaction level

   The objects managed at this level are reactions. Reactions have pointers to materials that can catalyze them. If the problem reaction is identical to a reaction contained in DECADE'S knowledge base, then the properties attached to the known reaction are associated with the problem reaction (in particular the catalytic material).

2. Molecular level

   In this level the objects are molecules. Molecules are parts of a reaction if they are contained in the reactant side or the product side of that reaction. It is possible to deduce properties of a reaction by observing the molecules that form it (in particular, it is possible to classify it). Once a reaction is recognized as a member of a reaction class, something additional may be said about the materials that can catalyze the problem reaction.

3. Species/Metal Surface level

   The objects in this level are species that can exist on the surface of a metal while a reaction is taking place. With these species (which are deduced from the molecular level and a set of heuristics), and a collection of rules, it is possible to propose a mechanism or series of steps that have to take place on the surface in order for the reaction to take place; These steps are affected by the reaction conditions and the nature of the surface *(i.e.* the catalytic material).

The above classification goes from decreasing level of abstraction, increasing level of difficulty in terms of problem solving, and decreasing level of accuracy in the prediction.

3.4. Implementation

There are many possible implementation procedures available inside the knowledge-based systems domain. Their choice depends on the basic approach taken [Hayes-Roth&WL 83]:

1. General-Purpose Programming Languages.
   Program the knowledge-based system from scratch. Although in principle any language can be used, only a few are convenient
   - LISP and dialects (e.g. MACLISP, FRANZLISP, INTERLISP, ZLISP, COMMONLISP, etc.) (see for example [Winston&Horn 81]).
   - PASCAL [Jensen&Wirth 74].

» FORTRAN, etc.

The advantage of this approach is that the knowledge-based system can be built as flexible, and appropriate ao dosired. The dinnclvanlago is that every capability has to be created by the programmer, where many of these tools and routines may already be programmed elsewhere.

2. Skeletal Systems.
   Borrow from previously built systems. One common practice is to generalize a successful knowledge-based system and try to make it domain-independent, using this generalization in the construction of a new system. This has the advantage of allowing one to create a new knowledge-based system in a very short time with relatively small effort. The main problems with this approach are that many times the skeletons are very inflexible and difficult to extend to deal with tasks not originally planned; and that the generalization step is not completely successful, thus keeping some domain-specific characteristics that could interfere with the new task. In general, their convenience is directly proportional to the similarity of the original task and the proposed new application.
   - **MYCIN** [Davis&BS *77]* -• **EMYCIN** [vanMelle&SB81]-* PUFF [Feigenbaum *77]*
   - CASNET[Weiss&KS 77] -• EXPERT [Weiss&Kulikowski 79]
   - PROSPECTOR [Reboh81] -» KAS [Reboh&Duda80]-> **HYDRO** [Gaschnig&RR 81] and CONPHYDE [Bañares&WR 85]

3. General-Purpose Representation Languages.
   Developed specifically for knowledge engineering. A knowledge-based system can be tailor-made using high-level tools in the development. Although they are supposedly domain independent, their choice should be affected by the proposed application.
   - ROSIE, RLL[Greiner&Lenat80], HEARSAY-III [Erman&LF81]
   - OPS5[Forgy81], OPS83
   - **SRL1**.5 [Wright&Fox 83], **PSRL** [Rychener 84]
   - LOOPS by XEROX, KEE by IntelliCorp, **Knowledge Craft** by Carnegie Group

Aside from the three approaches just described, other factors affect the choice of implementation, and they could prove to be even more important: the type of problem to be solved, the desired capabilities of the knowledge-based system, and the availability of the selected tools.

3.4.1. LISP

Figure 3-7 is the Lisp implementation of the formula shown in Figure 3-3.

```
(defun deih ()
    (plus delhO
          (times a dt)
          (times 0.5 b dt2")
          (times (quotient 1.0 3.0) c dt3**)
          (times 0.25 d dt4*«)
          Hvap
    ))
```

Figure 3*7: Example of a numerical Lisp function

For brevity we are assuming that the values for the variables (e.g. a, dt2* *, Hvap) are global and have been bound previously.

The next figure is somewhat more typical to Lisp. It is a recursive function that works with symbolic arguments rather than with numerical ones. Its purpose is to compare if the objects contained in the list *listi* are all also contained in the list *list?..* The function is used when determining if **two** reactions are the same (by testing if they have the same set of reactants and **products).**

```
(defun compare-mats (listl list2)
    (cond ((null listl))
          (t (cond ((member (first listl) list2)
                    (compare-mats (tail listl)
                                  (delete (first listl) list2)))
                   (t nil)
             ))
    ))
```

Figu re 3-8: Example of an algorithmic Lisp function

The first statement in the conditional (cond) is the termination condition, a test to see if *listl* is empty. If *listl* is **not empty, then the function tests if the first element of *listl* is a member of** *list2%* **if** so, **the** *compare-mats* **function is applied recursively to the tail of *listl* as first argument and the result of extracting the coincident element from** *list2.*

3.4.2. OPS5

One of the languages used in DECADE is OPS5. OPS5 is a programming language used extensively in knowledge-based system applications and in other Artificial Intelligence areas. OPS5 primitives are production rules that "fire* *{i.e.* execute its actions when its preconditions are matched) according to the content of working memory, and whose actions modify that working memory, create other production rules, or perform information input/output. For a better description of the language, consult its manual [Forgy 81], and the recent book on OPS5 programming [Brownston&FKM 85].

OPS5 is a very appropriate language to encode the production rules contained in DECADE, *i.e* the **control rules and the heuristics. Figure 3-9 is the codification of the production rule shown in Figure 3-t. It uses predefined objects such as *goal, EKS* (knowledge source) and** *reaction,* **in order to** match **its conditions. Its actions can modify the matched objects or can create new ones as in the** case **of** *messg.* **Boldface strings are OPS5 operators.**

```
(p sm::+arg-precond
    { (goal
          ^name          propose-mechanism
          ^assigned-to   mechanism-eks
          ^status        in-process
          ^argument      { <> nil <reaction-name> }
          ^id            <id-number>                )     <goal> }
    { (EKS
          ^name          mechanism-eks
          ^status        active                     )     <EKS> }
    - (reaction
          ^name          <reaction-name>
          ^classification <> nil                    )
    -->
    (make goal
          ^name          classify-reaction
          ^status        proposed
          ^argument      <reaction-name>
          ^subgoal-of    <id-number>
          ^id            (lisp-function:generate-unique-id)  )
    (modify <goal>
          ^status        waiting                    )
    (modify <EKS>
          ^status        waiting                    )
    (make messg
          ^type          entering-subgoal
          ^source        mechanism-eks
          ^status        sent
          ^default       classify-reaction          )
)
```

Figure 3-9: Example of an OPS5 rule

3.4.3. SRL

SRL1.5 [Wright&Fox 83] (or SRL for short) is a language interpreted into FranzLISP that runs on a VAX computer using the UNIX operating system. It has been developed by the Intelligent Systems Laboratory at Carnegie-Mellon University.

It is appropriate for declarative knowledge, and it is therefore used for descriptive purposes. SRL supports a very sophisticated representation of concepts and their relations, and the support is very flexible. It is possible to inherit values from related **schemata**, specify the inheritance path, modify the inheritance mechanism, etc. Figure 3-10 is the codification of the frame shown in Figure 3-2. Double quoted symbols stand for other frames (therefore the slots containing them are relations). *Lambda* is a generic name of a function, in this example both *Lambda*s have the same argument: *schema*. The function *mat@rside* with the parameters (*material reaction side*) returns **true** if the *material* is present in the *side* of the *reaction*, and **false** otherwise. (*e.g.*

```
{{ hydrogenolysis
        is-A: "class of reaction"
        ISCLASS: "mul timolecular"
        HASCLASSES: "OC hydrogenolys is" "f ischer- tropsch"
                    "hydrodesu1furization"
        CONSTRAINTS: 2
        CONSTRAINT TYPE: and
        CONSTRAINTS (lambda (schema)
                            (matSrside H2 schema reactants))
              explanation: "H2" is a reactant
        CONSTRAINT2:
           (lambda (schema)
                   (or (= (length (valueg schema products)) 2)
                       (and (= (length (valueg schema products)) 1)
                            (> (nth-coeff schema products 1) 1))))
              explanation: "C-C" bond breaking}}
```

Figu re 3-10: Example of an SRL schema

(mat@rside ^{H}CO^{H} ^{M}methanation^{H} "reactants") -+ true). The function *length* with the parameter *(list)* returns the number of elements that are contained in the *list,* it is applied to the result of (valueg *reaction* "products"), a list containing the products of *reaction.* Lastly, the result of executing (nth-coeff *reaction* "products" *n)* is the coefficient of the n^{m} product in the *reaction.*

3.5. Conclusions

In summary, it would always be important to consider the use of the most appropriate language for the representation and solution of a subproblem. This factor has to be weighed against the advantages of uniformity. One disadvantage of using hybrid systems is that a diversity of representations may hide some of the sequencing of a task. Another is that since data types are in general not the same, it is necessary to check for data type consistency and compatibility.

4. DECADE as a hybrid knowledge-based system

As we have seen so far, there are advantages to using different representation and control structures within the same system. The blackboard model is a general and simple architecture that allows the integration of dissimilar pieces of code ([Hayes-RothB 83] is a good introduction to the blackboard model architecture). DECADE is an example of the implementation of a blackboard model hybrid system.

4.1. Blackboard Model Structure

The blackboard model is a paradigm that allows for the flexible integration of modular pieces of code into a single problem-solving environment, it is a general and simple model that allows for the representation of a variety of behaviors. Given its nature, it is prescribed for problem-solving in knowledge intensive domains that use large amounts of diverse, errorful and incomplete knowledge, therefore requiring multiple cooperation of knowledge sources in the search of a large problem space. In terms of the type of problems that it can solve there is only one major assumption: that the problem-solving activity generates a set of intermediate results.

It was originally proposed in the development of Hearsay-II, a speech understanding system that interprets spoken requests for information from a database [Hayes-Roth&Lesser *77],* Since then it has been used in a number of application programs, for example for signal processing ([Nii&FAR 82]), design of alloys([Hulthage&RFF 85]), VLSI design ([Bushnell&Director 85]), and several more.

The blackboard model consists of a data structure (the blackboard) containing information (the context) that permits a set of modules (knowledge Sources or KSs) to interact (as illustrated in Figure 4-1).

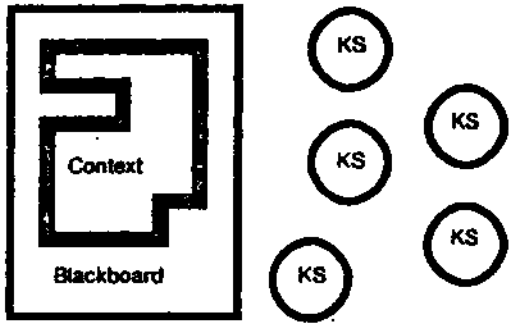


Figure 4-1: General Structure of a Blackboard

In the following subsection the structure of a typical blackboard model is described in more detail.

4.1.1. The Blackboard

The blackboard can be seen as a global database or working memory in which distinct representations of knowledge and intermediate results are integrated uniformly. It can also be seen as a means of communication among knowledge sources, mediating all of their interactions. Finally, it can be seen as a common display, debugging and performance evaluation area.

It may be structured so as to represent different levels of abstraction and also distinct and possibly

overlapping intervals in the solution. The division of the blackboard into levels parallels the process of abstraction of the knowledge, allowing elements at each level to be described approximately as abstractions of elements at the next lower level. This partition of the knowledge may be not only natural, but useful, in that a partial solution (i.e. group of hypotheses) at one level can be used to constrain the search at adjacent levels.

4.1.2. The Knowledge Sources

The Knowledge Sources in DECADE are kept separate, independent and anonymous (*i.e.* they do not have to know of the existence of the rest). This partitioning into "modules" is useful in that it makes the problem more tractable by creating a set of subproblems, each of which is easier to pose and solve. Of course, at a higher level their activities must be coordinated. Equally important, this separation eases the modification and evaluation of the system.

Knowledge Sources in DECADE are divided into two components:

1. Condition, Precondition or Front End

   Monitors the blackboard for elements matching its precondition. The precondition has the double purpose of finding a subset of hypotheses that are appropriate for an action and of invoking the knowledge source in that subset. The subset has been called the *Stimulus Frame* of the knowledge source instantiation ([Lesser&Erman 77]). Each knowledge source is data-directed in that it monitors the blackboard for data matching its precondition.

2. Action or knowledge-specific component

   When the precondition component is matched, a copy of the knowledge source is instantiated (invoked) and finally executed. In the case that more than one knowledge source fulfilled its precondition part, the execution is subject to the result of a **conflict resolution process** (more on this in the blackboard model control section).

The knowledge sources may be divided in a number of different ways, depending on the characteristic that is used to discriminate them.

1. Generic vs. specific

   The knowledge source may be useful in a whole set of knowledge-based systems (*e.g.* the Focus of Attention), or specific to one application (*e.g.* the mechanism prediction knowledge source).

2. Unique vs. redundant

   Several knowledge sources performing the same task but with different capabilities may be present in the same system. The difference in capabilities can be in terms of accuracy, consumed resources, certainty of the result, required preconditions, etc.

3. Local vs. distributed

Knowledge sources may reside in the same processor or in different ones.

4. Homogeneous vs. hybrid

Knowledge sources may have the same structure and/or control, **but** they may be completely different in either.

Table 4-1 lists the current components of the blackboard environment present in DECADE. The **headers of the second to fifth columns of the table reflect the dimensions in which DECADE is a hybrid system.**

Name	Knowledae ReDresentation	Problem Solvina Method	Lanauaae	Levels	[rules.functionsl
User Interface	rules	—	OPS5	—	[12,16]
Focus of Attention	rules	agenda	OPS5	—	[23,1]
Scheduler	rules	—	OPS5		[9,0]
Specify Reaction	rules frames	search numerical	OPS5 SRL	1.2	[50,25]
Thermo Checking	functions rules	generate & test numerical	Lisp OPS5	1	[25,4]
Classify Reaction	functions frames	search	SRL OPS5	1.2	[14,15]
Surface Steps	rules frames	means-ends	OPS5 SRL	2,3	[42,36]
Select Catalyst	rules frames functions	search generate & test	SRL OPS5	1,2,3	[143,59]

Table 4-1: DECADE as an example of a Blackboard Model.

The last column contains two numbers representing the amount of code (and therefore the size) of **each of the knowledge sources. The first quantity is the number of production rules, and the** second **the number of Franz Lisp functions. Not all the code is represented in the table though. There** are

assorted Franz Lisp functions used in the communication of OPr>!5 and SHI . Also, the frames or schemas constituting most cf the database are not present (they are not counted in the table because there is no clear-cut assignment of a given schema to a knowledge source). I"ho total amount of code is:

- OPS5 production rules: 318
- Franz Lisp functions: 203
- SRL schemas: 328

4.1.3. The Context

The context is the set of entries or context elements contained in the blackboard that contain the information representing the state of the solution process. Those entries may include perceptions, observations, beliefs, hypotheses, decisions, goals, interpretations, judgements, or expectations. Also, they may have relationships to one another. In particular, one such organization may combine a set of entries as the representation of a single object viewed from different levels of abstraction.

In DECADE there are objects that represent goals *{goal),* questions and information messages *(messg),* knowledge sources *(EKS),* and other general concepts in the blackboard. There are also domain specific objects: those which represent reactions *(reaction),* catalysts *{catalyst),* surface steps *(ss),* etc. Figure 3-9 shows some of these objects in action.

4.2. Blackboard Model Control

The blackboard model can accommodate a range of control mechanisms and problem-solving strategies. This flexibility in range applies at all levels: from each of its components (knowledge sources), to the system as a whole.

4.2.1. Overall control

In DECADE, the overall control is determined by one of the knowledge sources: the Focus of Attention. A simplified description of the behavior of the Focus of Attention is schematized in Figure 4-2. A lower case string can be interpreted to be a production rule that is part of the Focus of Attention knowledge source. The rectangles represent parts of the process where the control passes to knowledge sources other than the Focus of Attention. According to the figure, after the user selects the kind of problem he wants to solve, the rule *post goal* will post in the blackboard a description of the goal that needs to be solved. Any knowledge source that has access to the blackboard and is able to solve such kind of problem can post an estimate for the solution of the

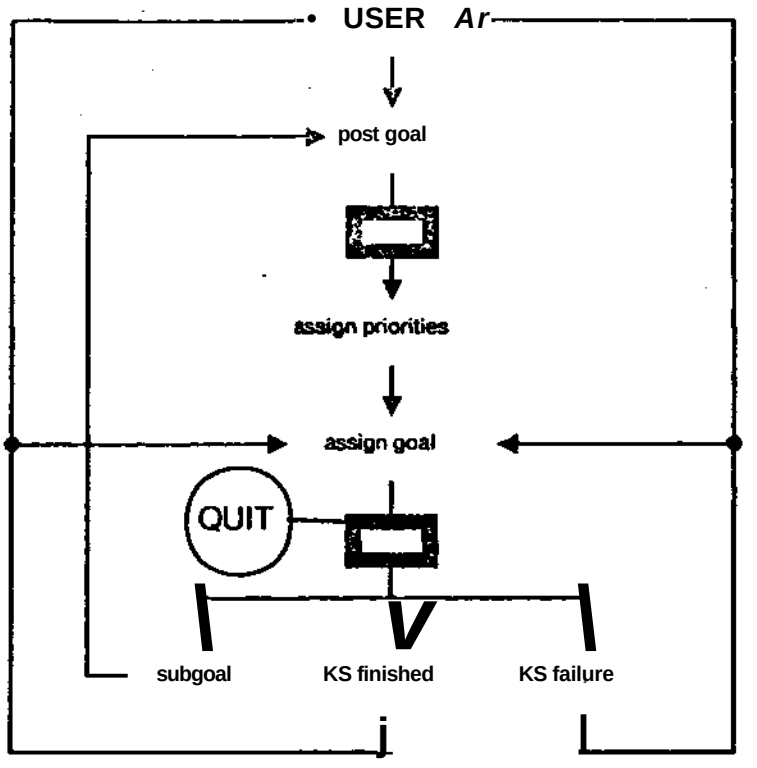


Figure 4-2: Simplified version of the overall control in DECADE

previously posted goal. Since this last step is performed by modules other than the Focus of Attention, it is depicted as a rectangle. The Focus of Attention waits until all the estimates have been submitted, then it evaluates them, assigning priorities to each of the knowledge sources that submitted an estimate. Once each knowledge source is rated, the best one is assigned the original goal, and that module will start the solution of it.

There are three possible outcomes after a goal has been assigned to a knowledge source.

1. The module solves the problem, it posts its solution in the blackboard and returns the control to the user (if the goal was originally requested by him), or to the part of the Focus of Attention that assigns the next goal (when the goal was requested by another knowledge source as a subgoal — see next item —)

2. The module cannot solve the requested problem because it needs some other partial results. In this case a subgoal is posted, or, more accurately, a goal with a pointer to the parent goal is posted in the next level of recursion. The subgoal is to be treated exactly as any other goal, with the only difference that any outcome from it is not considered a final solution, but is passed to the knowledge source that requested it. There is no limit to the levels of recursion that can be used.

3. The module cannot solve the requested problem; it failed. In this case the control is handed back to the part of the Focus of Attention that assigns the execution of goals. If more estimates are present the goal is assigned to the next best estimate, if not the control is handed back to the user.

4.2.2. Internal control in the knowledge sources

We have described the behavior of DECADE as a whole. Also, each of the knowledge sources has a set of rules representing a common internal control mechanism for the effects of common operation inside the blackboard environment. It reflects the actions that the knowledge source must take with respect to factors such as the presence or absence of an argument, the success or failure of the preconditions, the success or failure of the knowledge source itself, etc. This control mechanism can be subdivided into three parts:

1. The first part is a rule that responds to requests for identification (see the left side of Figure 4-3). This is used in what is called the *census*. Having this internal process permits the use of a set of initially anonymous knowledge sources, while at the same time it permits the system to know which modules are present during run time, and what are their main characteristics. This knowledge is only used as information to the user, and in no way diminishes the generality of the architecture.

2. The second part is another rule that responds to requests for estimates as described in Figure 4-2 (see right side of Figure 4-3).

3. The third part is a set of rules that determines when a goal can be solved straightforwardly, how to request a subgoal and receive its result, and finally what to do if a goal has been requested but it is not clear which argument should be used. Explanation of this last item follows in the next paragraphs.

The central part of Figure 4-3 shows the internal manipulation of goals in a knowledge source, it follows the same notation as the one from Figure 4-2. Once a goal has been assigned to a knowledge source, the rule *begin action* activates the knowledge source and starts processing the goal. There are several possibilities at this point, but the most important ones are:

1. Rule *+ arg + precond* fires.

   The goal has an specified argument and its preconditions are satisfied. In this case the knowledge source executes its actions and whenever it ends, it passes the control back to the blackboard (in effect the Focus of Attention).

2. Rule *+ arg-precond* fires.

   The goal has a specified argument but its preconditions have not been satisfied. New subgoals are proposed corresponding to the preconditions that have not been satisfied in

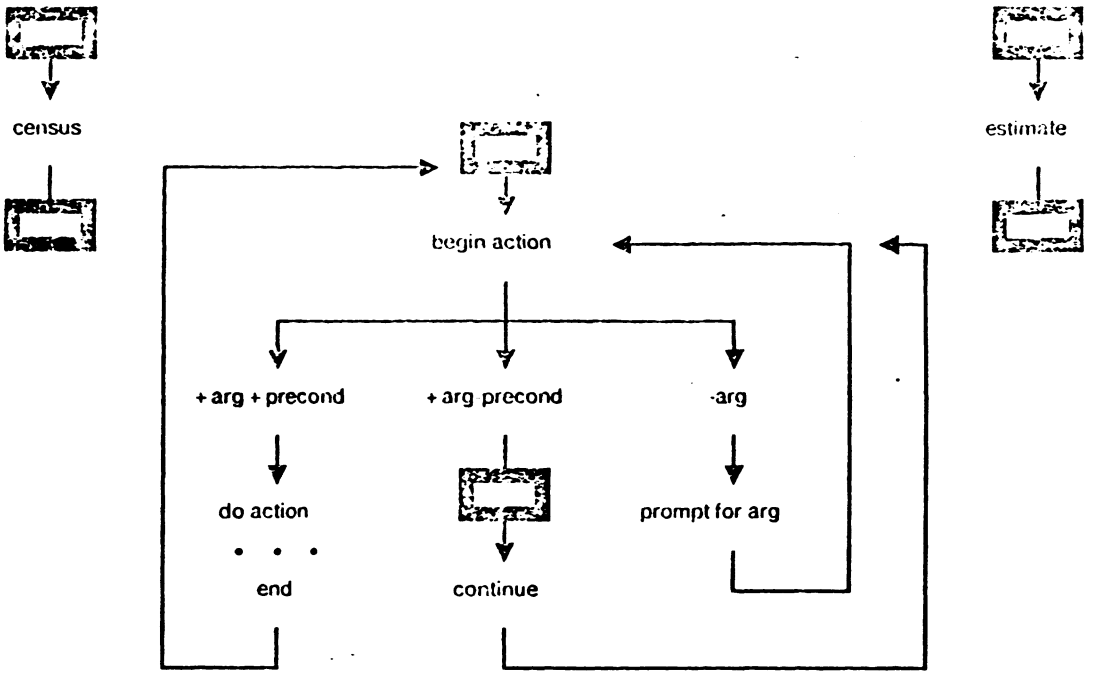


Figure 4-3: Simplified version of the control inside the knowledge sources

an attempt to do so. Whenever these subgoals are solved (assuming no failure, in which case the Focus of Attention would take charge of the situation), the original knowledge source continues with its execution by having reduced the situation to *+ arg + precond*.

3. Rule *-arg* fires.

   This is a particular case of the *+ arg-precond* situation, if we consider that obtaining an argument (by asking the user to give one, or prompting for the necessary data to form one) can be thought of as a subgoal. In this case the subgoal is much simpler though.

4.3. Using DECADE

While this paper has concentrated on the organization and control aspects of DECADE from the expert system's point of view, a follow up paper will present in detail the selection strategy from the "chemistry" point of view. Specifically there are three levels of responding to a request for catalyst selection:

1. responding with catalysts that are known to work from published experimental results and that are in the DECADE knowledge base,

2. classifying the reaction from generic to specific classes and recommending catalysts known to catalyze reactions in all of these levels of classification, and

3. determining both required and not wanted surface *r,u*\lvr,* (adsorption, desorption, dissociation, addition) for the reaction, and selecting catalysts known to enhance the needed steps and to suppress the unwanted ones for related reactions.

Table 4-2 shews catalysts and operating conditions recommended by DECADE at the third level for the synthesis of methane, ethane, methanol and ethanol.

Product	Temperatures (K)	Pressures (atm)	Prescribed materials
methane	**600-700**	**≤ 10.**	Fe, **Ru, Co,** Rh, **Ni, Pd**
methanol	**500-600**	**≥ 100**	**groups IB & IIB, Rh, Ir, Pd, Pt, Cu**
ethane	**500-600**	**-10**	**Fe, Ru,Co, Rh, Ni, Pd**
ethanol	**500-600**	**~ 10**	**Rh, Ir, Pd, Pt**

Table 4-2: Results for different reactions at the same level of abstraction.

These species include no chain formation (methane, methanol), chain formation (ethane, ethanol), complete CO dissociation (methane, ethane), partial dissociation (ethanol) and no dissociation **(methanol). These answers compare very favorably with known literature results.**

The following paper will also illustrate the explanation facilities of DECADE for an example request. Specifically, DECADE can explain why certain catalysts are (or are not) selected, why the operating conditions ranges are selected and which surface steps are required to be present or absent.

4.4. Conclusions

This section has presented the basic components of the blackboard model architecture: its **structure and operation. In particular, it has showed how these concepts are used in DECADE,** The **blackboard architecture is, nevertheless, useful in the general case where the integration of** various **representation and problem-solving methods is desired.**

5. Future Work and Conclusions

All the features described in this document are implemented and tested, but as of the middle of September of 1985 DECADE is not yet fully operational. Results from the different abstraction levels have yet to be conciliated, knowledge about some reactions completed, the explanation capabilities finished, etc.

Several other issues will have to be investigated in the future in order to make the application of knowledge-based systems to engineering more practical. A few of these items are:

- Characterization of the control

  In DECADE the control was centralized within the Focus of Attention knowledge source. Alternative control schemes with decentralized control could prove to be more powerful.

- Parallel processing

  DECADE runs in a single computer. Although no mention has been made of this issue, it is conceivable that the efficient integration of several processors could be the single most important factor in improving the performance of knowledge-based systems. This area remains largely unexplored.

- Alternative implementations

  This document has only investigated one way of implementing hybrid knowledge-based systems. There are other systems that make use of different representations and problem-solving methods at the same time (see [Kunz&KW 84] for example). There are hybrid environments (*e.g.* LOOPS, KEE, Knowledge Craft) and hybrid languages (*e.g.* OPS83, S.1) that would be worth evaluating as an alternative to the blackboard architecture.

  Although it is not the purpose of this document to compare the ideas presented here with other systems, one comment is in order: systems like KEE, which were designed as hybrids, have a larger degree of integration than the one achieved in a blackboard model; but in order to attain such integration, the subparts tend to be less powerful than the subparts that DECADE has. In other words: there is a compromise between degree of integration and the expressive power of the subparts.

- Integration of symbolic and numerical computing

  The main purpose of the document has been to introduce the reader to the symbolic parts of engineering design, yet the issue of coupling numerical and symbolic computations still exist. It is addressed in the next subsection.

5.1. Links between symbolic and numerical languages

The question of integrating traditional numerical algorithms with knowledge based systems has been the goal of several research efforts in Chemical Engineering (see [Bariares&SVWR 85] for a review of knowledge-based systems in Chemical Engineering). The approach has been in general to start from a typical Chemical Engineering system and then adapt pieces of nontraditional code/programming techniques to it. The present research has taken a different stand, in that it began in the Expert System end of the problem and has been extending to the traditional algorithmic end, therefore DECADE currently uses very limited amounts of numerical computation (and the little it has is encoded in FranzLisp). Parallel research has been conducted in the interfacing of existing Fortran programs to symbolic code, and the results have been successful.

As a result of the effort of a team in the Design Research Center at Carnegie-Mellon University, the following results can be reported on four different methods that have been used to connect symbolic to numerical languages ([Talukdar&CLBJ 85]). The results are summarized in Table 5-1.

	Disk Files	Foreign Functions	Pipes or Sockets	IPC
Operating System	Any	Any	UNIX 4.1/4.2	UNIX 4.1
Concurrency	Yes	No	Yes	Yes
Number of Machines	One[a]	One	Multiple[6]	Multiple
Implementation languages	Any	Lisp, C, Fortran, Pascal	Lisp, C	Lisp, C Pascal
Transferred information	I/O	Parameters	I/O	I/O
Data type	Any	Limited by language	Strings	Structured data
Additional code	None	None	Minimal[6]	Minimal[0]
Relative Speed[d]	2	1	3	4

(a) Multiple if the installation has over-the-network file-transfer facilities
(b) One machine in the case of pipes, Multiple machines for sockets
(c) About 250 lines of code to use pipes and IPC from Franz Lisp
(d) 1 is the fastest. 4 the slowest

UNIX is a trademark of Bell Laboratories

Table 5-1: Four ways to link programs written in different languages.

1. Disk Files: The programs to be linked read and write from common files. This is a simple mechanism but tends to be slow.

?. Foreign Functions: LISP, C :\nc\ PASCAL allow compiled versions of programs written in other languages to be treated as callabie functions. I he parameters that can be passed through this mechanism vary from LISP to C to PASCAL. The interpreted version of OPS5 runs in a LISP environment. Therefore, OPS5 programs can use foreign function calls to exchange data with other programs.

3. Pipes **and** Sockets: These are interprocess communication facilities provided by UNIX.

4. IPC: This is **an Inter** Process Communication facility written at CMU **for** machines running UNIX 4.1 and connected by a local area network. IPC places no restrictions on the data types to be communicated. Connections between C and PASCAL programs are **made directly. FORTRAN and LISP require small C** interfaces.

5.2. Conclusions

Experience in the application of knowledge-based systems to engineering design has shown us that the use of hybrid systems is necessary. Hybrid knowledge-based systems make it possible to take full advantage of the existing programs, and also allow the automatization of a larger portion of the **design process.**

Hybridity is a concept that can be achieved in several dimensions: knowledge representation and **abstraction, implementation languages, problem solving methods, etc. In order to construct hybrid systems it is necessary to be able to mix different components (programming languages, modules, programs, levels of abstraction, etc.) into a common working space. Although no complete solution has been developed to accomplish this goal, the blackboard model seems to be the ideal paradigm for this purpose.**

In particular we have presented the specific problem of catalyst selection, described how it consists **of a variety of subtasks, described how each one of these subtasks is subject to representation and solution using existing computer methods, and finally presenting DECADE, a knowledge-based system that illustrates the above ideas.**

References

[Agniholri 78] Agnihotri, Rajani Bhushan.
Computer-Aided Investigation of Reaction Path Sytithesis.
PhD thesis, Chemical Engineering Department, University **of** Houston, August, 1978.

[Anderson&KR **84]**
Anderson Robert Bernard, Kolbel H. and Ralek M.
The Fischer-Tropsch Synthesis.
Academic Press, Inc., New York, 1984.

[Bañares&SVWR 85]
Bañares-Alcántara R., Sriram D., Venkatasubramanian V., Westerberg A. and RychenerM.
Knowledge-Based Expert Systems for CAD.
Chemical Engineering Progress **81 (9):25-30, September, 1985.**

[Bañares&WR 85] Bañares-Alcántara René, Westerberg W. Arthur and Rychener Michael D.
Development of an Expert System for Physical Property Predictions.
Computers & Chemical Engineering 9(2): **127-142,1985.**

[Bañares- Alcántara 86]
Bañares-Alcántara René.
*DECADE. **A** hybrid knowledge-based system **for** catalyst selection.*
PhD thesis, Chemical Engineering Department, Carnegie-Mellon University, January, 1986.

[Bell 81] **Bell T. Alexis.**
Catalytic Synthesis of Hydrocarbons over Group VIII Metals. A Discussion of the Reaction Mechanism.
*Catal. **Rev. - Sci. Eng.** **23(1 & 2):203-232,1981.***

[Brownston&FKM 85]
Brownston L, Farrel R., Kant E. and Martin N.
*Programming Expert Systems in OPS5. **An** Introduction **to Rule-Based Programming.***
Addison-Wesley Publishing Company, Inc., Reading, Mass., 1985.

[Bushnell&Director 85]
Bushnell Michael L. and Director S. W.
*ULYSSES - **An Expert-System Based** VLSI **Environment.***
Technical Report, Department of Electrical and Computer Engineering, Carnegie-Mellon University, Pittsburgh, PA 15213,1985.

[Chandrasekaran&Mittal 83]
Chandrasekaran, B. and Mittal Charles.
Deep versus compiled knowledge approaches to diagnostic problem-solving.
International Journal of Man-Machine Studies **19:425-436,1983.**

(Cohere Feigenbaum 83]
Cohen Paul R. and Feigenbaurn tEdward A.
The Handbook of Artilicial Intelligence.
HeurisTech Press, Stanford, California, 1983, pages 515-522, Chapter XV: Planning and Problem Solving.

[Davis&BS *77]* Davis R., Buchanan B.G. and Shortliffe E.
Production rules as a representation for a knowledge-based consultation program.
Artificial Intelligence 8:15-45, 1977.

[Dry 81J Dry M.E.
The Fischer-Tropsch Synthesis.
In Anderson John R. and Boudart Michael (editors), *CATALYSIS. Science and Technology,* **chapter 4, pages 160-255. Springer-Verlag, New York, 1981.**

[Erman&LF 81] Erman L.D., London P.E. and Fickas S.F.
The design and an example use of HEARSAY-III.
In *Proceedings of the Seventh International Joint Conference on Artificial* ***Intelligence,*** **pages 409-415. International Joint Conferences on Artificial Intelligence, 1981.**

[Feigenbaum *77]* Feigenbaum E.A.
The art of artificial intelligence: Themes and case studies of knowledge engineering.
In *Proceedings of* ***the*** *Fifth International Joint Conference on Artificial Intelligence,* ***Volume I,*** **pages 1014-1029. International Joint Conferences on Artificial Intelligence, Cambridge, Massachusetts USA, August, 1977.**

[Fikes&Kehler 85] Fikes Richard and Kehler Tom.
The Role of Frame-Based Representation in Reasoning.
Communications of the ***ACM*** **28(9):904-920, September, 1985.**

[Forgy 81] Forgy Charles L.
OPS5 User's ***Manual***
Department of Computer Science, Carnegie-Mellon University. Pittsburgh, PA 15213,1981.
CMU-CS-81-135.

[Gaschnig&RR81]
Gaschnig J., Reboh R. and Reiter J.
Development *of a* ***knowledge-based expert*** *system* ***for water resource*** *problems.*
Final Report SRI Project 1619, SRI International, August, 1981.

[Genesereth&Ginsberg 85]
Genesereth Michael R. and Ginsberg Matthew L.
Logic **Programming.**
Communications of the ACM **28(9):933-941**, **September, 1985.**

[Greiner&Lenat80]
Greiner R. and Lenat D.
A representation language **language.**
*AAA1***1:165-169,1980.**

[Haggin 81] Haggin Joseph.
Fischer-Tropsch: new life lor old technology.
Chemical & Lnyincctinj N<y.vs 139:20-02, October 2G, 1901.

[Happel&Sellers 83]
Happel John and Sellers Peter H.
Analysis of the Possible Mechanisms for a catalystic reaction system.
In *Advances in Catalysis.* Academic Press Inc., 1983.

[Hayes-Roth 85] Hayes-Roth Frederick.
Rule-Based Systems.
Communications of the ACM 28(9):921 -932, September, **1985.**

[Hayes-Roth&Lesser *77]*
Hayes-Roth, Frederick and Lesser R. Victor.
Focus of Attention in the Hearsay-II Speech Understanding System.
In *Proceedings of the Fifth International Joint Conference on Artificial Intelligence,* ***Volume I*, pages 27-35. International Joint Conferences on Artificial Intelligence, Cambridge, Massachusetts USA, August, 1977.**

[Hayes-Roth&WL83]
Hayes-Roth, Frederick, Waterman A. Donald and Lenat B. Douglas editors.
Teknowledge Series in Knowledge Engineering. **Volume 1: *Building Expert Systems.***
Addison-Wesley Publishing Company, Reading, Massachusetts USA, 1983.

[Hayes-RothB 83] Hayes-Roth Barbara.
The Blackboard** Architecture: **A** General **Framework** for **Problem Solving?.
Heuristic Programming Project Report No. HPP-83-30, Computer Science Department, Stanford University, May, 1983.

[Hulthage&RFF85]
Hulthage Ingemar, Rychener Michael D., Fox Mark S., and Farinacci Martha L.
The Use of Quantitative Databases in Aladin, an Alloy Design System.
In *Proceedings of the Workshop on Coupling Symbolic and Numerical Computing in **Expert Systems.*** **American Association of Artificial Intelligence, Bellevue, Washington, August, 1985.**

[Jensen&Wirth 74]
Jensen, Kathleen and Wirth, Niklaus.
PASCAL User Manual and Report. Second Edition.
Springer-Verlag, 1974.

[King&CG 81] King D.L., Cusumano J.A. and Garten R.L.
A Technological Perspective for Catalytic Processes Based on Synthesis Gas.
***Catal. Rev. -** Sci. Eng.* **23(1 & 2):233-263, 1981.**

[Klier 82] Klier K.
Methanol Synthesis.
In *Advances in Catalysis,* pages **243-313. Academic Press, Inc., 1982.**

[Kunz&KW 84] Kunz J. C, Kehler T. P. and Williams M. D.
Applications Development Using **a Hybrid Al Development System.**
***TheAl Magazine* :41-54, Fall, 1984.**

[Lesser&Erman 77]
Lesser Victor R. and Erman Lee D.
A Retrospective view of the Hearsay-II Architecture.
In *Proceedings of the Fifth International Joint Conference on Artificial Intelligence, Volume I*, pages 790-800. International Joint Conferences on Artificial Intelligence, Cambridge, Massachusetts USA, August, 1977.

[Michie82] Michie, D.
High-road and low-road programs.
AI Magazine **3(1):21**-22,**1982.**

[Newell 69] Newell Allen.
Heuristic Programming: Ill-Structured problems.
In Aronofsky Julius S. (editors), *Progress in Operations Research. Relationship Between Operations Research and* ***the Computer,*** **chapter 10, pages 361 -414. John Wiley & Sons, Inc., USA, 1969.**

[Nii&FAR 82] Nii Penny H.f Feigenbaum Edward A., Anton John J. and Rockmore A.J.
Signal-to-Symbol Transformation: HASP/SIAP Case Study.
The ***AI Magazine*** **:23-35, Spring, 1982.**

[Nilsson 71] Nilsson Nils J.
Problem-Solving Methods in Artificial Intelligence.
McGraw-Hill Book Company, USA, 1971.

[Reboh 81 j Reboh, Rene.
Knowledge Engineering Techniques and Tools in the PROSPECTOR Environment.
Technical Note 243 SRI Project 5821,6415 and 8172, Artificial Intelligence Center. SRI International, June, 1981.

[Reboh&Duda 80] Reboh R. and Duda R.O.
KAS - The Knowledge acquisition system for PROSPECTOR.
Technical Report, Expert System Workshop, 1980.
pages 38-41.

[Rofer-DePoorter 81]
Rofer- DePoorter C.K.
A Comprehensive Mechanism for the Fischer-Tropsch Synthesis.
Chemical Reviews **81:447-474,1981.**

[Rychener 84] Rychener Michael D.
PSRL: An SRL-Based Production System
DRAFT edition, Intelligent Systems Laboratory. The Robotics Institute., Carnegie-Mellon University. Pittsburgh, PA 15213,1984.

[Stefik 82] Stefik Mark, Aikins Janice, Balzer Robert, Benoit John, Birnbaum Lawrence, Hayes-**Roth Frederick and Sacerdoti Earl.**
The Organization of Expert Systems: A Prescriptive Tutorial.
Technical Report VLSI-82-1, XEROX. Palo Alto Research Centers, 1982.

[Talukdar&CLBJ 85]
Talukdar S. N., Cardozo E., Leão L., Bañares-Alcántara R. and Joobani R.
A System for Distributed Problem Solving.
In *Proceedings of the Workshop on Coupling Symbolic and Numerical Computing in Expert Systems*. American Association of Artificial Intelligence, Bellevue, Washington, August, 1985.

[Trimm 80] Trimm L. David.
Chemical Engineering Monographs. Volume 11: *Design of Industrial Catalysts*.
Elsevier Scientific Publishing Company, Amsterdam, The Netherlands, 1980.

[vanMelle&SB 81] van Melle W., Shortliffe E.H. and Buchanan B.G.
EMYCIN: A domain-independent system that aids in constructing knoledge-based consultation programs.
MachineIntelligence 9(3), 1981.

[Weiss&KS 77] Weiss S.M., Kulikowski C.A. and Safir A.
A model-based consultation system for the long-term management of glaucoma.
In *Proceedings of the Fifth International Joint Conference on Artificial Intelligence, Volume I*, pages 826-832. International Joint Conferences on Artificial Intelligence, Cambridge, Massachusetts USA, August, 1977.

[Weiss&Kulikowski 79]
Weiss S.M. and Kulikowski C.A.
EXPERT: A system for developing consultation models.
In *Proceedings of the Sixth International Joint Conference on Artificial Intelligence*, pages 942-947. American Association of Artificial Intelligence, Tokio, Japan, 1979.

[Winston&Horn 81]
Winston P.H. and Horn B.K.P.
LISP.
Addison-Wesley, Reading, Mass., 1981.

[Wright&Fox 83] Wright J.M., Fox Mark S.
SRL1.5 User Manual
Intelligent Systems Laboratory. The Robotics Institute., Carnegie-Mellon University. Pittsburgh, PA 15213, 1983.