AN INTEGRATED THEORY OF PROBLEMS AS AN ALGEBRAIC BASE FOR COMPLEXITY UNDERSTANDING AND AUTOMATIC PROBLEM SOLVING.

Giovanni Guida - Dino Mandrioli - Amedeo Paci - Marco Somalvico

Milan Polytechnic Artificial Intelligence Project - MP-AI Project
Milan, Italy.

1. INTRODUCTION

The studies about the theory of problem solving, to which belong the results that we shall present in this paper, are intended to achieve the following main goals :

- a rather precise understanding of the human behaviour in problem solving activity;
- a clear definition of what we mean by an Automatic Problem Solver (APS);
- the formulation of an Abstract Theory of Problem-Solving which can clearly point out the theoretical possibilities and the limits of an Automatic Problem Solver;
- a proposal of an efficient structure of an Automatic Problem Solver which can perform the three basic activities of selection, search and learning;
- the formulation of a Theory of Problems which can be helpful as a theoretical base in the design of an Automatic Problem Solver;
- further investigations about Automatic Problem Solvers as non-deterministic interpreters of an high-level representation language and as automatic programmers;
- implications of the developped theories on fields of actual interest; e.g., Industrial Robotics, Computer Aided Medical Diagnosis, Intelligent Data Base Management Systems, Question Answering Systems, etc.

These studies are being developped at the Milan Polytechnic Artificial Intelligence Project since three years.

The purpose of this paper is to present some new results achieved by the authors in the above mentioned fields of research. The paper is divided into two Parts.

In part A, which is constituted by the Sections II and III, an Integrated Theory of Problems in presented in a formal way.

In Part B, which is constituted by the Sections IV and V, some implications of the theory and important directions of future research are described.

More in detail the paper is devoted to present :

- in Section II, the basic definitions and results of the formalization of the State-Space Approach to Problem-Solving;
- in Section III, the basic definitions and results of the formalization of the Problem-Reduction Approach to Problem-Solving.
- in Section IV, a first cut Theory of Complexity, which constitutes an unformal base for the definition of a measure of complexity;
- in Section V, our basic point of view on Problem-Solving and a detailed unformal structure of an Automatic Problem Solver.

Part A - The Formal Theory

In this first part the formalization of the classical State-Space and Problem-Reduction Approaches to Problem-Solving [6], [9] is presented.

This formalization is achieved by means of an algebraic tool strictly connected with the Theory of Graphs [1], [3] and with the Theory of the AND/OR Graphs. Because of the unitary way in which the two different approaches have been considered during the formalization, the outcoming theory can in fact be called an Integrated Theory of Problems.

II. STATE-SPACE : FORMALIZATION

In this Section we present some basic definitions and results

of the formalization of the State Space Approach to Problem-Solving. A further investigation of these topics may be found in [4] . The Algebraic Theory of Automata [3] and the Theory of Graphs [1] were both taken into account in setting up the following theory.

The prooves of the theorems of this Section are omitted for the sake of brevity and can be found by the reader in [4].

We first present some basic definitions of the Theory of Graphs which shall be useful for the following investigations.

Definition 2.1

A (directed, labeled) graph is a triple G = (V,A,R) where :

- $V = \{v_o, \ldots, v_{n-1}, \ldots\}$ is a set of elements called the vertices of G;
- $A = \{a_o, \ldots, a_{m-1}, \ldots\}$ is a set of elements called the labels of G;
- $R = \{R_{a_o}, \ldots, R_{a_{m-1}}, \ldots\}$ is a set of functions from V into V.

□

Definition 2.2

An arc of a graph G = (V,A,R) is a couple of vertices $u=(v_o,v_1)$ such that :

$$(\exists a)\ ((a \in A) \wedge (v_o\ R_a = v_1)) \qquad (2.1)$$

the vertex v_o is called the initial vertex and the vertex v_1 is called the final vertex.

□

Definition 2.3

A loop is an arc $u = (v_o, v_1)$ such that $v_o = v_1$.

□

Definition 2.4

A path of a graph G = (V,A,R) from a vertex $v_i \in V$ to a vertex $v_f \in V$ is a finite sequence of vertice $\mu = (v_i \ldots, v_1, \ldots, v_{k-1}, v_f)$

such that :

$$(\exists x)\ ((x \in A^{+}) \wedge (x = a_{i_1} \ldots a_{ik}) \wedge (v_1 = v_1 R_{a_{i_1}}) \wedge \ldots \wedge (v_f = v_{k-1}\ R_{a_{i_k}}) \tag{2.2}$$

where:

$$A^{+} = A^{*} \{\varepsilon\}$$

The string x is called a generating string of μ. Moreover we say that the path μ has length k.

□

We can now present the basic definitions of the State-Space Approach.

Definition 2.5

A (deterministic) problem schema M is a triple $M = (S, \Sigma, \Gamma)$ where :

- $S = \{s_o, s_1, \ldots, s_{n-1}, \ldots\}$ is a set of elements called the states of M;
- $\Sigma = \{\sigma_o, \sigma_1, \ldots, \sigma_{m-1}, \ldots\}$ is a set of elements called the inputs of M;
- $\Gamma = \{\gamma\sigma_o, \gamma\sigma_1, \ldots \gamma\sigma_{m-1}, \ldots\}$ is a set of functions of S into S called the operators of M.

□

Definition 2.6

A (deterministic) problem $\tilde{P}$ is a quintuple $\tilde{P} = (S, \Sigma, \Gamma, i, f)$: where (S, Σ, Γ) is a (deterministic) problem schema, and :

- $i \in S$ is called the initial state;
- $f \in S$ is called the final state.

□

Definition 2.7

A (deterministic) extended problem P is a quintuple $P=(S, \Sigma, \Gamma, I, F)$ where (S, Σ, Γ) is a (deterministic) problem schema and :

- $I \subseteq S$ is called the set of the initial states;
- $F \subseteq S$ is called the set of the final states.

□

Definition 2.8

A solution of the problem $P = (S, \Sigma, \Gamma, i, f)$ is a string :

$$x = \sigma_{i_1} \sigma_{i_2} \ldots \sigma_{i_k} \in \Sigma^* \tag{2.3}$$

such that :

$$i \gamma_x = f \tag{2.4}$$

where :

$$\gamma_x = \gamma_{\sigma_{i_1}} \gamma_{\sigma_{i_2}} \cdots \gamma_{\sigma_{i_k}} \tag{2.5}$$

(i.e. γ_x is made up by the composition of operators), and γ_ε is the identify function on S, if ε is the null string.

□

Definition 2.9

A solution of the extended problem $\tilde{P} = (S, \Sigma, \Gamma, I, F)$ is a string $x \in \Sigma^*$ such that :

$$(\exists i)\ (\exists f)\ ((i \in I) \wedge (i \gamma_x = f) \wedge (f \in F)) \tag{2.6}$$

□

Definition 2.10

The solution set of a (extended) problem P $(\tilde{P})$ is the set $X_P \subseteq \Sigma^*$ $(X_{\tilde{P}} \subseteq \Sigma^*)$ which contains all the solutions of P $(\tilde{P})$.

□

We outline that the solution set X_P of a problem P is not necessarily finite. We are now able to introduce some initial formal properties of these notions.

Theorem 2.1. Given an extended problem $\tilde{P} = (S, \Sigma, \Gamma, I, F)$ we have :

$$X_{\tilde{P}} = \bigcup_{P_i \in E_{\tilde{P}}} X_{P_i} \tag{2.7}$$

where :

$$E_{\tilde{P}} = \left\{ P_i \mid P_i=(S,\Sigma,\Gamma, i, f) \wedge (i \in I) \wedge (f \ F) \right\} \qquad (2.8)$$

□

Although this theorem states a close relation between the solution of an extended problem and the solution of a set of problem, there is no indication about the methods of how to "reduce" in a general case the solution of an extended problem $\tilde{P}$ to the solutions of the set of problems $E_{\tilde{P}}$.

In fact this "reduction" is closely related to the search strategy adopted in the problem solving process.

Conversely, we want to focus our interest in the following pages only on problems and their properties.

We now outline explicitely the close relation existing between the definition of problem and the definition of graph.

Definition 2.11

The underlying graph of the problem $P = (S,\Sigma,\Gamma, i, f)$ is the graph $G = (S,\Sigma,\Gamma)$.

□

We assume as well known the concept of length of a string x that we shall denote by $l(x)$.

We outline that if $x \in \Sigma^*$ is the solution of a problem $P = (S,\Sigma,\Gamma,i,f)$ $l(x)$ is exactly the number of expansions required to obtain the solution x, i.e. the number of times that an operator $\gamma_\sigma \in \Gamma$ has to be applied.

Definition 2.12

A (k-step) solution of a problem $P = (S,\Sigma,\Gamma, i, f)$ is a solution $x \in \Sigma^*$ of P such that $l(x) = K$.

□

Definition 2.13

The (k-step) solution sequence generated by the (k-step) solution $x = \sigma_{i_1} \sigma_{i_2} \dots \sigma_{i_k}$ of the problem $P (S,\Sigma,\Gamma, i, f)$ is the se-

quence of states:

$$G_x^{(k)} = (i, s_1, s_2, \ldots, s_{k-1}, f) \qquad (2.9)$$

such that :

$$s_1 = i \; \gamma\sigma_{i_1}$$

$$s_2 = s_1 \; \gamma\sigma_{i_2}$$

$$\vdots$$

$$s_{k-1} = s_{k-2} \; \gamma\sigma_{i_{k-1}} \qquad (2.10)$$

$$f = s_{k-1} \; \gamma\sigma_{i_k}$$

□

It is evident that a (k-step) solution sequence of a problem is in fact a path of length k from i to f.
Let us now introduce the basic definition of cost.

Definition 2.14

A cost function c on a set Σ is a measure function of Σ^* into R_+ (set of real non negative numbers) such that :

$$c(xy) = c(x) + c(y) \qquad (\forall x)\,(\forall y)\,((x \in \Sigma^*) \wedge (y \in \Sigma^*)) \qquad (2.11)$$

□

Theorem 2.2 . A cost function c on a set Σ is completely determined by its restriction to Σ.

□

Definition 2.15

Given a cost function c on Σ , the simple cost C of a solution $x \in X_p$ of a problem $P = (S, \Sigma, \Gamma, i, f)$ is defined as :

$$C(x) = c(x) \qquad (\forall x)\,(x \in X_p) \qquad (2.12)$$

□

Definition 2.16

Given a cost function K on S x Σ the composite cost K of a solution $x \in X_p$ of a problem $P = (S, \Sigma, \Gamma, i, f)$ is defined as :

$$K(x) = K(\tilde{x}) \tag{2.19}$$

where :

$$\tilde{x} = (i, \sigma_{i_1}) (S_1, \sigma_{i_2}) \dots (S_{q-1}, \sigma_{i_q}) \tag{2.14}$$

iff :

$$x = \sigma_{i_1} \sigma_{i_2} \dots \sigma_{i_q}$$

and

$$G_x^{(k)} = (i, S_1, S_2, S_2, \dots, S_{q-1}, f) \tag{2.16}$$

is the (q-step) solution sequence generated by the (q-step) solution x of P.

□

Definition 2.17

A solution $\bar{x} \in \Sigma^*$ of a problem $P = (S, \Sigma, \Gamma, i, f)$ is minimal iff :

$$l(\bar{x}) = \min_{x \in X_p} \{ l(x) \} \tag{2.17}$$

□

Definition 2.18

A solution $\bar{x} \in \Sigma^*$ of a problem $P = (S, \Sigma, \Gamma, i, f)$ is simply (compositely) optimal iff :

$$C(\bar{x}) = \min_{x \in X_p} \{ C(x) \} \; (K(\bar{x})) = \min_{x \in X_p} \{ K(x) \}) \tag{2.18}$$

□

We outline that in general neither the existence nor the uniqueness of a simply (compositely) optimal o r minimal solution of a problem can be proved.

III. PROBLEM-REDUCTION : FORMALIZATION

In this Section we present some basic definitions and results of the formalization of the Problem- Reduction Approach to Problem-Solving. The adopted formalism is congruent with that one used in the view setting up an unitary, integrated theory.

We first present some basic definitions of the Theory of Graphs which shall be useful for the following investigations.

Definition 3.1

A (directed, labeled) AND/OR graph is a quadruple $\Delta = (V,A,R,W)$ where :

- $V = \{ v_o, \ldots, v_{n-1}, \ldots \}$ is a set of elements called the vertices of ;
- $A = \{ a_o, \ldots, a_{m-1}, \ldots \}$ is a set of elements, called the labels of ;
- $R = \{ R_{a_o}, \ldots, R_{a_{m-1}}, \ldots \}$ is a set of functions from V into V;
- $W = \{ w_o, \ldots, w_{u-1}, \ldots \}$ is a set of elements called the AND/OR constraints of Δ and associated to the vertices.,

where for each $i = 0, \ldots, n-1, \ldots$ associated to the vertex v_i, we have:

$$w_i = \{ d_{i_1}, d_{i_2}, \ldots, d_{i_{K_i}} \} , \qquad K_i \geq 0 \tag{3.1}$$

and for each $j = i_1, i_2, \ldots i_{K_i}$:

$$d_j = a_{J_1} a_{J_2} \ldots a_{J_l} \in A^* \tag{3.2}$$

such that :

$$\bar{d}_j \subseteq \mathcal{P}_i \tag{3.3}$$

where :

$$\bar{d}_J = \{ x \mid x \text{ is an element of } d_J \} \subseteq A \tag{3.4}$$

$$\xi_i = \{ a_s \mid (a_s \in A) \wedge (v_i \text{ is an element of the domain of } R_{a_s}) \} \subseteq A \qquad (3.5)$$

We note that :

$$d_j = \xi \quad \text{iff} : \quad \bar{d}_J = \phi \qquad (3.6)$$

□

Definition 3.2

A (directed, labeled) canonical AND/OR graph Ω is an AND/OR graph $\Delta = (V, A, R, W)$ such that for each $i = 0, \ldots, n-1, \ldots$ we have :

$$- \mid w_i \mid = 1 , \qquad (3.7)$$

or

$$- \quad w_i = \{ d_1, d_2, \ldots, d_j, \ldots, d_{k_i} \}$$

and for each $j = 1, \ldots, k_i$:

$$l(d_j) = 1 \qquad (3.8)$$

If condition (3.7) is met we say that the corresponding vertex v_i is an AND-vertex, if condition (3.8) is met we say that the corresponding vertex v_i is an OR-vertex.

□

It is obvious that the set V of the vertices of a canonical AND/OR graph Ω is partitioned into the following two subsets :

$$- \bar{V} = \{ v \mid (v \in V) \wedge (v \text{ is an AND-vertex of } \Omega) \}$$

$$- \tilde{V} = \{ v \mid (v \in V) \wedge (v \text{ is an OR-vertex of } \Omega) \}$$

Moreover an algorithm can be easyly defined which allows to construct for each (directed, labeled) AND/OR graph Δ a (directed, labeled) canonical AND/OR graph Ω which is "equivalent" to Δ . (the meaning of the word equivalent is now left to the intuition of the reader). We shall denote by the symbol $\sqcup$ the concatenation of sequence; i.e., the associative, non commutative, operation which associates to each ordered pair of sequences A_1, A_2 the sequence $A = A_1 \sqcup A_2$ containing exactly the elements of A_1 followed by the elements of A_2.

Definition 3.3

An AND/OR path froma vertex v_i to a set of vertices $V_f = \{ v_{f_1}, v_{f_2}, \ldots, v_{f_h} \}$ of and AND/OR graph $\Delta = (V,A,R,W)$ is a finite sequence $\rho = (b_o, b_1, \ldots, b_{v_1})$ of finite sequences of vertices of Δ such that:

$$b_o = (v_i)$$

$$b_1 = (v_1^1, v_2^1, \ldots, v_{p_1}^1)$$

$$\vdots$$

$$b_q = (v_1^q, v_2^q, \ldots, v_{r_1}^q, v_{r_1+1}^q, \ldots, v_{r_1+r_2}^q, \ldots, v_{r_1+r_2+\ldots+r_{p_{q-1}}=p_q}^q) \quad (3.9)$$

$$\vdots$$

$$b_k = (\hat{v}_1^k, \ldots, \hat{v}_h^k = p_k)$$

where :

- $b_1 = (v_1^1, v_2^1, \ldots, v_{p_1}^1)$ iff : (3.10)

$$(\exists d_j)\ ((d_j \in w_o) \wedge (d_j = a_{j_1} a_{j_2} \ldots a_{j_{p_1}}) \wedge (v_o R a_{j_1} = v_1^1)\ (v_o R a_{j_2} = v_2^1) \wedge \ldots \wedge (v_o R a_{j_{p_1}} = v_{p_1}^1)) \quad (3.11)$$

- for $q = 2,3,\ldots,k$

$$b_q = \bigsqcup_1^{p_{q-1}} \bar{b}_i \quad (3.12)$$

where :

$$\bar{b}_i = (\hat{v}_i^{q-1}) \quad \text{only if :} \quad (3.13)$$

$$v_i^{q-1} \in V_f \quad (3.14)$$

$$\bar{b}_i = (v_{r_1+\ldots+r_{i-1}+1}^q, v_{r_1+\ldots+r_{i-1}+2}^q, \ldots, v_{r_1+\ldots+r_{i-1}+r_i}^q) \quad (3.15)$$

only if :

$$(\exists d_j)((d_j \in w_i^{q-1}) \wedge (d_j = a_{j_1}\, a_{j_2} \ldots a_{j_{r_i}}) \wedge$$

$$(v_i^{q-1}\, R\, a_{j_1} = v^q_{r_1+\ldots+r_{i-1}+1}) \wedge \ldots \wedge \qquad (3.16)$$

$$(v_i^{q-1} R a_{j_{r_i}} = v^q_{r_i+\ldots+r_{i-1}+r_i}))$$

(where w_i^{q-1} denotes the $w \in W$ associated to the vertex v_i^{q-1})

- b_{k-1} contains at least one vertex which has not the superscript $\wedge$

- b_k is a permutation of the elements of V_f.

Moreover we say that the path ρ has <u>length</u> K-1.

□

We can now present the basic definitions of the Problem-Reduction Approach. We outline that in our formalization we have defined separately the syntactic aspects and the semantic ones ,in order to obtain a better evidence of the theory. In fact the semantics is presented in Definitions 3.4 and 3.5, and the syntax in Definitions 3.6, 3.7, 3.9 and 3.10.

We first introduce in a formal and general way, the concept of relations between problems, on which our formalization of the Problems-Reduction Approach is based. All the possible relations existing between problems which one of any interest for us can be based on the "comparison" of their solution sets as in the following definitions :

<u>Definition 3.4</u>

An <u>implicant</u> of a problem P is a couple $L = (\pi, \Psi)$ where :
- $\pi = \{P_1, P_2, \ldots, P_k\}$ is a finite set of problems;
- Ψ is a mapping of $X_{P_1} \times X_{P_2} \times \ldots \times X_{P_k}$ into X_P.

We will write :

$$\pi \xrightarrow{\Psi} P \qquad (3.17)$$

If Ψ is a mapping onto X_p, then L is said a full implicant of P. In this case, we will write :

$$\pi \overset{\Psi}{\Longrightarrow} p \qquad (3.18)$$

□

Some important and usual particular cases of this general definition can be found by the reader in [4].

Definition 3.5

An implicance schema is a couple $\Xi = (\mathcal{P}, \Theta)$ where :

- $\mathcal{P} = \{ P_o, P_1, \ldots, P_{n-1}, \ldots \}$ is a set of problems;
- $\Theta = \{ \Lambda_o, \Lambda_1, \ldots, \Lambda_{m-1}, \ldots \}$ is a set such that $|\Theta| \leq |\mathcal{P}|$

and for each $j = 0,1,\ldots, m-1, \ldots$:

$$\Lambda_j = \{ L_o, L_1, \ldots, L_{p_j} \} \qquad (3.19)$$

is a finite set of implicants of the problem $P_j \in \mathcal{P}$ such that for each $i = 0,1,\ldots, p_j$;

$$\pi_i \subseteq \mathcal{P} \qquad (3.20)$$

□

Definition 3.6

A reduction schema N is a triple $N = (\mathcal{P}, \Sigma, \Gamma)$ where

- $\mathcal{P} = \{ P_o, P_1, \ldots, P_{n-1}, \ldots \}$ is a set of problems;
- $\Sigma = \{ \sigma_o, \sigma_1, \ldots, \sigma_{m-1}, \ldots \}$ is a set of elements, called the inputs of N;
- $\Gamma = \{ \gamma_{\sigma_o}, \gamma_{\sigma_1}, \ldots, \gamma_{\sigma_{m-1}}, \ldots \}$ is a set of function of P into P, called the operators of N.

□

Definition 3.7

A constraint on a reduction schema $N = (\mathcal{P}, \Sigma, \Gamma)$ is a set

$$Y = \{ y_o, y_1, \ldots, y_{n-1}, \ldots \}$$

where for each i = 0, ..., n-1,..., associated to the Problem P_i, w e have :

$$y_i = \{ e_1, e_2, \ldots, e_{k_i} \} \qquad k_i \geqslant 0 \tag{3.21}$$

where for each $J = 1,2,\ldots, k_i$:

$$e_J = \sigma_{j_1} \sigma_{j_2} \ldots \sigma_{j_{l_j}} \in \Sigma^* \tag{3.22}$$

such that :

$$\bar{e}_j \subseteq T_i \tag{3.23}$$

where :

$$\bar{e}_j = \{ x^{\chi} \mid x^{\gamma} \text{ is an element of } e_J \} \subseteq \Sigma \tag{3.24}$$

$$T_i = \{ \sigma_s \mid (\sigma_s \in \Sigma) \wedge (P_i \text{ is an element of the domain of } \gamma_{\sigma_s}) \tag{3.25}$$

we assume that :

$$e_j = \varepsilon \qquad \text{iff :} \qquad \bar{e}_j = \emptyset \tag{3.26}$$

□

Definition 3.8

A constraint Y on a reduction schema $N = (\mathfrak{P}, \Sigma, \Gamma)$ is natural iff there exists an implicance schema $= (\mathfrak{P}, \mathfrak{M})$ such that, if we assume :

$$\begin{aligned} \mathfrak{P} &= \{ P_o, P_1, \ldots, P_{n-1}, \ldots \}, \\ \mathfrak{M} &= \{ \wedge_o, \wedge_1, \ldots, \wedge_{m-1}, \ldots \}, \\ Y &= \{ y_o, \ldots, y_{n-1}, \ldots \} \end{aligned} \tag{3.27}$$

we have :

$$- \ |\mathfrak{M}| = |Y| \tag{3.28}$$

- for each i = 0,1,..., m-1,..., n-1,... :

$$y_i = \{ e_1, e_2, \ldots, e_{k_i} \} \tag{3.29}$$

and for each $j = 1,2,\ldots, k_i$

$$e_j = \sigma_{j_1} \sigma_{j_2} \ldots \sigma_{j_{l_j}} \tag{3.30}$$

iff :

$$\Lambda_i = \{ L_1, L_2, \ldots, L_{k_i} \} \tag{3.31}$$

and

$$\Pi_j = \bigcup_{1}^{l_j} s \{ P_i \gamma \sigma_{j_s} \} \tag{3.32}$$

□

Definition 3.9

A (deterministic) reduction problem schema is a quadruple $M = (\mathfrak{P}, \Sigma, \Gamma, Y)$ where $(\mathfrak{P}, \Sigma, \Gamma)$ is a reduction schema and Y is a constraint on $(\mathfrak{P}, \Sigma, \Gamma)$.

□

Definition 3.10

A (deterministic) reduction problem is a sextuple $Z = (\mathfrak{P}, \Sigma, \Gamma, Y, P_i, \mathfrak{P}_f)$ where $(\mathfrak{P}, \Sigma, \Gamma, Y)$ is a reduction problem schema and :

- $P_i \in \mathfrak{P}$ is called the initial problem;
- $\mathfrak{P}_f \subseteq \mathfrak{P}$ is called the set of the final problems.

□

We now outline explicitely the close relation existing between the definition of reduction problem and the definitions of graph and AND/OR graph.

Definition 3.11

The underlying graph of the reduction problem $Z = (\mathfrak{P}, \Sigma, \Gamma, Y, P_i, \mathfrak{P}_f)$ is the graph $G = (\mathfrak{P}, \Sigma, \Gamma)$

□

Definition 3.12

The underlying AND/OR graph of the reduction problem $Z = (\mathfrak{P}, \Sigma,$

Γ, Y, P_i, $\mathcal{P}_f$) is the AND/OR graph $\Delta = (\mathcal{P}, \Sigma, \Gamma, Y)$. □

It is now clear that, as we have already pointed out, the formalism of the reduction problem schemata (or equivalently of the AND/OR graphs) is just the syntax of the implicance schemata, which on the other side are the semantics of the reduction problems schemata. We outline that the same reduction problem schema could have several "interpretations", i.e. associated implicance schemata, and viceversa.

Differently from the method followed in Section II for introducing the concept of solution, we are now going to define first the solution sequence and then the solution of a reduction problem.

Definition 3.13

A(k-step) solution sequence of a reduction problem $Z = (\mathcal{P}, \Sigma, \Gamma, Y, P_i, \mathcal{P}_f)$ is an AND/OR path ρ of the underlying AND/OR graph $(\mathcal{P}, \Sigma, \Gamma, Y)$ from the vertex P_i to the set of vertices $V_f = P_{f_1}, P_{f_2}, \ldots, P_{f_h}$ such that :

$$V_f \subseteq \mathcal{P}_f \tag{3.33}$$

and

$$l(\rho) = k \tag{3.34}$$

□

Definition 3.14

The (k-step) solution generated by the (k-step) solution sequence $\rho = (b_o, b_1, \ldots, b_{k+1})$ of the problem $Z = (\mathcal{P}, \Sigma, \Gamma, Y, P_i, \mathcal{P}_f)$ is the string $x = \mathcal{L}_1 \mathcal{L}_2 \ldots \mathcal{L}_k$ of finite sequences $\mathcal{L}_j$ of strings $\xi_i \in \Sigma^*$ such that i_f :

$$\begin{aligned} b_o &= (P_i) \\ b_1 &= (P^1_1, P^1_2, \ldots, P^1_{p_1}) \\ &\vdots \end{aligned} \tag{3.35}$$

$$b_q = (p^q_1, p^q_2, \ldots, p^q_{r_1}, p^q_{r_1+1}, \ldots, p^q_{r_1+r_2}, \ldots p^q_{r_1+r_2+\ldots+r_{p_{q-1}}} = p_q)$$

.

.

.

$$b_{k+1} = (p^{k+1}_1, \ldots, p^{k+1}_{p_{k+1}})$$

we have :

$$\mathcal{L}_1 = (\xi^1_1)$$

$$\mathcal{L}_2 = (\xi^2_1, \xi^2_2, \ldots, \xi^2_{p_1}) \qquad (3.36)$$

.

.

.

$$\mathcal{L}_q = (\xi^q_1, \xi^q_2, \ldots, \xi^q_{p_{q-1}})$$

.

.

.

$$\mathcal{L}_k = (\xi^k_1, \ldots, \xi^k_{p_k})$$

where :

- ξ^1_1 is exactly the d_j referred to in (3.11)

- for q = 2,3,..., k :

$$\mathcal{L}_q = \bigsqcup_1^{p_{q-1}} i \; \mathcal{D}_i \qquad (3.37)$$

where :

$$\mathcal{D}_i = (\varepsilon) \qquad (3.38)$$

iff in (3.12) $\bar{b}_i$ is obtained as in (3.13)

$$\mathcal{D}_i = (d_j) \qquad (3.39)$$

where d_j is exactly the string referred to in (3.16) iff in (3.12) b_i is obtained as in (3.15).

Moreover we will call the set $F_x = \{ \hat{P}_1^{k+1}, \ldots, \hat{P}^{k+1}_{p_{k+1}} \} \subseteq \mathfrak{P}_f$ the covering set determined by the (k-step) solution x of Z.

□

Definition 3.15

A solution x of a reduction problem $2 = (\mathfrak{P}, \Sigma, \Gamma, Y, P_i, \mathfrak{P}_f)$ is a string of finite sequences of strings of Σ^* such that x is a (k-step) solution of Z geenerated by a (k-step) solution sequence ρ of Z for some nonnegative integer k.

□

Definition 3.16

The solution set of a reduction problem $Z = (\mathfrak{P}, \Sigma, \Gamma, Y, P_i, \mathfrak{P}_f)$ is a set X_Z such that :

$$X_Z = \{ x \mid x \text{ is a solution of } Z \} \tag{3.40}$$

□

Definition 3.17

The total length λ of x solution $x = \mathcal{L}_1 \ldots \mathcal{L}_k$ of a reduction problem $Z = (\mathfrak{P}, \Sigma, \Gamma, Y, P_i, \mathfrak{P}_f)$ is defined as :

$$\lambda(x) = \sum_{1}^{k} \sum_{1}^{p_{i-1}} 1 \, (\mathcal{L}^i_j) \tag{3.41}$$

□

We outline that $\lambda(y)$ is exactly the number of expansions required to obtain the solution x, i.e. the number of times that an operator $\gamma_\sigma \in \Gamma$ had to be applied.

We introduce now the important concept of cost of a solution of a reduction problem.

Definition 3.18

Given a cost function c on Σ the simple cost C of a solution $x = \mathcal{L}_1, \ldots, \mathcal{L}_k$ of a reduction problem $Z = (\mathfrak{P}, \Sigma, \Gamma, Y, P_1, \mathfrak{P}_f)$ is defined as :

$$C(x) = \sum_{1}^{k} \sum_{0}^{p_{j-1}} c(\xi^j_i) \tag{3.42}$$

Definition 3.19

A solution $\bar{x}$ of a reduction problem $Z = (\mathfrak{P},\Sigma,\Gamma, Y, P_i, \mathfrak{P}_f)$ is minimal iff :

$$\lambda(\bar{x}) = \min_{x \in X_Z} \{ \lambda(x) \} \tag{3.43}$$

□

Definition 3.20

A solution x of a reduction problem $Z = (\mathfrak{P},\Sigma,\Gamma, Y, P_i, \mathfrak{P}_f)$ is simply optimal iff :

$$C(x) = \min_{x \in X_Z} \{ C(x) \} \tag{3.44}$$

□

It is then possibly to define the composite cost of a solution x and a compositely optimal solution x in a fully analogous way as that one followed in Section II. We omit these definitions for the sake of brevity.

Definition 3.21

We say that a problem $P = (S,\Sigma,\Gamma, i, f)$ is solved iff a couple $T = (P, X)$ is given where :

$$X \subseteq X_p \tag{3.45}$$

and

$$X \neq \emptyset \tag{3.46}$$

If $X = X_p$ we say that P is fully solved.

□

We conclude this Section by pointing out the close relation existing between the two different approaches presented in Section II and III.

Theorem 3.1. Given a reduction problem $Z = (\mathfrak{P},\Sigma,\Gamma, Y, P_i, \mathfrak{P}_f)$ and a solution $x \in X_Z$ of Z if all the final problems of the covering set F_s determined by x are solved and if Y is a natural constraint on $(\mathfrak{P},\Sigma,\Gamma)$ it is possible to construct a solution $y \in X_{p_i}$ of P_i.

Proof

The proof of this Theorem is directly obtained from Definition 3.8, 3.13, 3.18, and 3.23.
The algorithm for building up y is also easily defined.

□

Part B - Topics for Future Research

In this second part some important directions for further investigation are presented. Informal bases for the definition of a measure of complexity inside the above presented Integrated Theory of Problems and for the design of an Automatic Problem Solver are described in detail.

IV - TOWARD A MEASURE OF COMPLEXITY

In this Section we present some preliminary concepts with the aim of setting up an unformal base for future definition of a measure of complexity inside the above presented Integrated theory of Problems.

The intuitive concept of complexity can be better specified by splitting it into the following three components :
- complexity of a solution;
- complexity of the algorithm used for finding out a solution;
- complexity of a representation of a problem.

Let us now investigate these three different concepts in more detail. The complexity of a solution is determined foundamentally by its simple (composite) cost and by its length. In fact the cost takes into account the complexity of the application of each operator necessary to build up the solution, and the length takes into account the number of an operators which must be applied, i.e. it gives an estimate of the time spent for the construction of the solution. This first type of complexity, which we shall call solution complexity

is of course related to a solution of a problem in a given representation, i.e. it can be useful for compairing different solutions of a some representation of a problem.

The complexity of the algorithm used for finding out the solution must take into account the time spent for the research, the memory occupation, and the number of unsuccessful attempts done in the such activity.

This type of complexity, which we shall call algorithm complexity it therefore related to an algorithm followed in the search of a given solution of a problem in a given representation.

We can then define for each couple (solution, algorithm) of a given representation of a problem a global complexity which takes into account both the solution complexity and the algorithm complexity and gives a quite precise estimate of the complexity of the global solution process of a problem.

Before proceeding further, we note that, as we have already outlined in Section III, the solution of a reduction problem $Z = (\mathcal{P}, \Sigma, \Gamma, Y, P_i, \mathcal{P}_f)$ is conceptually different from the solution of P_i, which is in fact the problem that we wish to solve. However the solution of P_i can be constructed from the solution of Z if the conditions of Theorem 3.1 hold. In this case, which we call the Ibrid Approach, it is clear that the global complexity of the solution of P_i must take into account the global complexity of the Problem-Reduction part, the solution complexity of the State-Space part, and the complexity of the construction of the solution of P_i, once the solution of Z is found.

We conclude this Section by briefly investigating the third type of complexity we mentioned above, i.e. the complexity of a representation of a problem, which we shall call the representation complexity. If we refer to a given intuitive problem and to a set of different representations of it, inside the same approach or inside different approaches (i.e., State-Space, Problem-Reduction, or Ibrid), we want to define e measure of complexity which clearly points out which representation is the most suitable (natural) for the given intuitive problem.

The representation complexity must therefore take into account an "average value" of the global complexities of all the possible couples (solution, algorithm) of a given representation of a problem.

V - TOWARD THE DESIGN OF AN AUTOMATIC PROBLEM SOLVER

In this Section we present our basic point of view on Problem - Solving and a first-cut, unformal, design of an Automatic Problem Solver. The standpoint of our considerations is the following one: computer science is an experimental discipline which is centered around an unitary and global goal : man-computer interaction in solving problems. This interaction can be syntetically represented as a path connecting the following basic concepts:

- intuitive problem,
- represented problem,
- solved problem.

The path itself is constituted by the three activities of formalization, automatic resolution, and matching.
These concepts are graphically illustrated in Figure 1 and they will be widely investigated in the whole Section.

The intuitive problem is an entity which independently faces the man and can be viewed as an undefined and unlimited source of information.

From it, through the activity of formalization, the man operates an extraction of a finite and precisely described amount of information, namely the represented problem. This information is chosen as valuable and sufficient in order to provide, through mechanical, or interactive, computation, the construction of the solved problem (i.e., the solution of the problem).

It is clear that in principle the solution construction can be performed in an artificial and completely automatic way by means of a general purpose computational tool that we shall call Automatic Problem Solver.

On the other hand, the activity of formalization belongs exclusi-

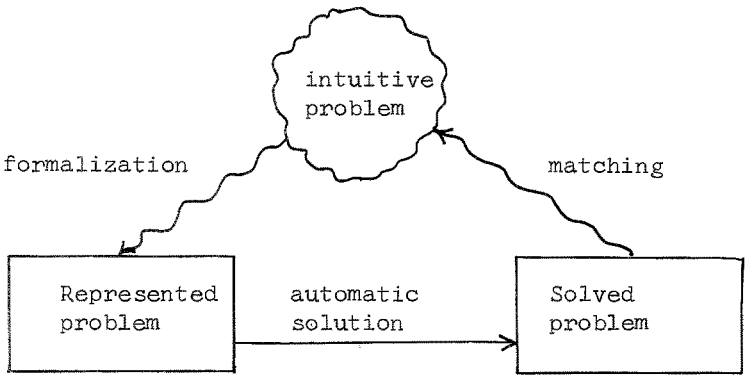

Figure 1 - Problem-Solving activity.

vely to the man.

Let us now investigate in more detail these outlined concepts.

The formalization activity, performed by the man, provides, as it has been previously exposed, the represented problem, as an artificial object which is obtained from the intuitive problem.

The invention of the represented problem consists in the precise description of a finite quantity of information which the man formulates by means of the observation of two distinct entities, namely: the intuitive problem, and the Automatic Problem Solver.
The invention of the represented problem requires that the man performs two basically different activities, in its formalization process.

The first activity is devoted to the specification of the methods and ways which "tune" the Automatic Problem Solver, considered as an originally general purpose tool, into a well precised special purpose tool which is oriented by the semantic domain from which the intuitive problem is originated.

The information described by consequence of this first activity, is called control information and it is the first part of the information contained in the represented problem.

The second activity is dedicated to the selection from the intuitive problem of a finite quantity of information which is considered by the man as useful, and, hopenfully, efficient and sufficient,

in order to allow the special Problem Solver to achieve its goal of providing an automatic solution of the problem.

The information described by consequence of this second activity, is called problem information and it is the second part of the information contained in the represented problem.

It is conceptually important to observe that both the two previously described activities are done by the man with the conscience of being faced by ignorances of two different types, namely :

- what part of the problem information is actually relevant to the computer and shall be utilized in order to solve the problem;
- what is the actual way in which this relevant information shall be processed in order to construct the solved problem.

These two types of ignorances are useful to point out two functions performed by the automatic problem solver, which are intended to give artificial answers to these ignorances.

The first function, which is devoted to produce an automatic answer to the first type of ignorance, consists in an appropriate selection of one part of the information contained in the represented problem, and considered, by the Automatic Problem Solver, as useful and relevant for its activity of solving the problem.

This activity is performed by a first part of the Automatic Problem Solver, called Selector, as it is shown in Figure 2, where all the block-structure of an Automatic Problem Solver is illustrated.

Therefore we will call global represented problem the input of the Selector and selected represented problem the output of the Selector.

The second function, which is devoted to produce an automatic answer to the record type of ginorance, consists in a skillful search of the cooperation process, embracing the already selected information, which essentially makes up the solution algorithm and, thus, yields the solution of the problem.

This activity is performed by a second part of the automatic problem solver, called Searcher, as it is shown in Figure 2.

Therefore, while the input of the Searcher will be the selected represented problem, the output of the Searcher will be the solved

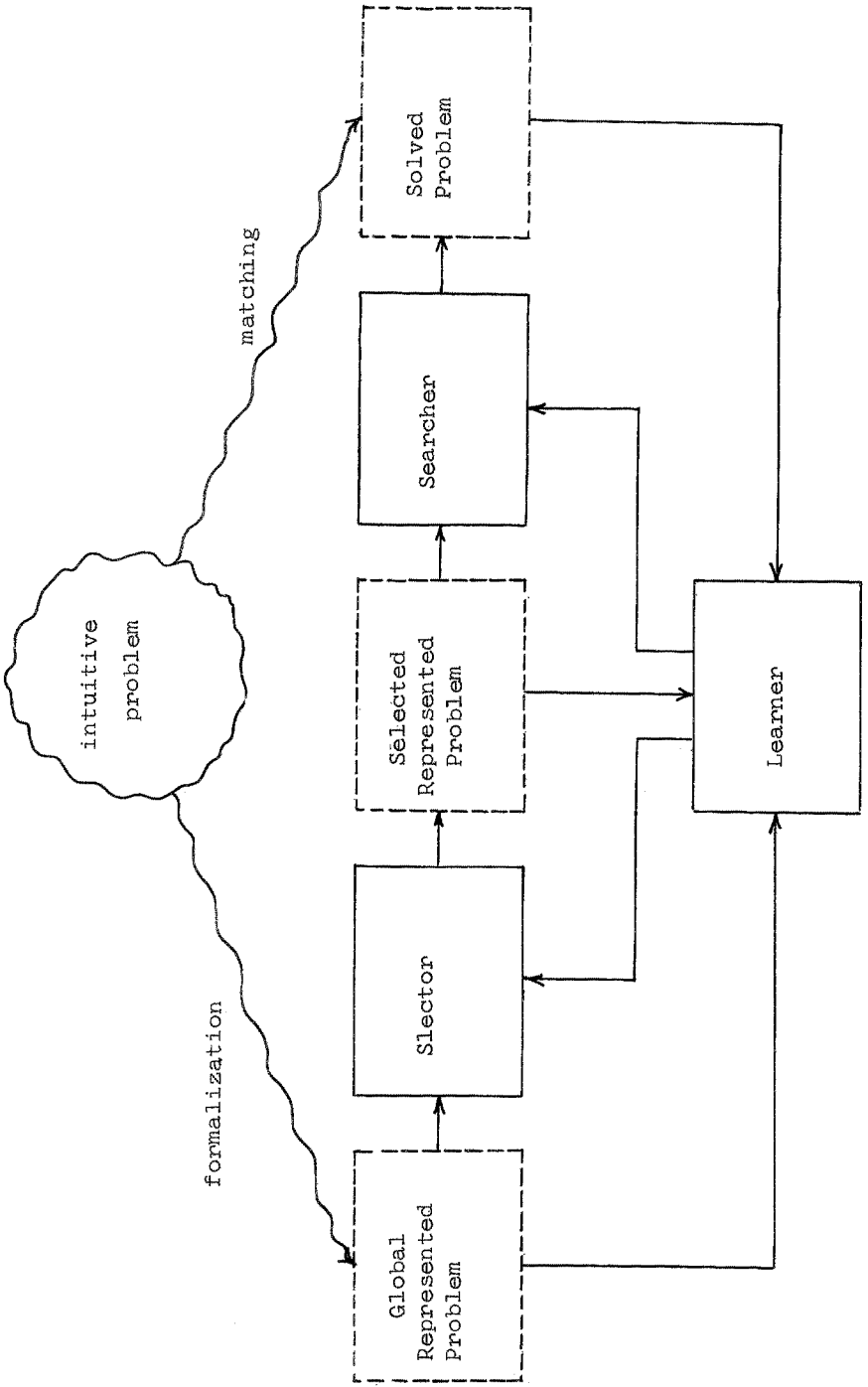

Figure 2 - Structure of an Automatic Problem Solver.

problem.

As it has been previously illustrated, the control information is the information which enables the man to specify the special configuration of the problem solver oriented towards a particular semantic domain.

Therefore, by means of the control information, the structures of the Selector and Searcher are completely specified.

This definition of the structure of the Selector and of the Searcher by the man can be considered just as an initial specification which, during the ongoing solution process, can possibly be changed and improved.

This modification and enhancing activity is the typical activity of learning which is able to provide a dynamic evolution of the structure of the Selector and of the Searcher.

This selfchanging activity is performed by a third part of the automatic problem solver, called Learner, as it is shown in Figure 2.

Therefore, the inputs of the Learner are constituted by the global represented problem, by the selected represented problem, and by the solved problem.

The outputs of the Learner are the automatically constructed and modifiable specifications of the Selector and of the Searcher.

Thus, the kernel of an automatic problem solver appears to be an artificial metasystem which is initialized by the man as an initial system, and, afterwards, can evolve itself, in a way appropriate to enhance its artificial performances in solving problems.

Therefore learning can be viewed as the ability of selfawereness of the whole automatic problem solver.

Whichever has been the method followed by the man in performing its formalization task for the construction of the represented problem it is necessary for him to choose an appropriate formalism apted both to provide a "good" represented problem and to catalyze a "valid" artificial activity for the automatic problem solver.

We can rightfully call such a formalism the Representation Language, which the man needs for cooperating with computer.

While the classic programming languages have been conceived to communicate to the computer algorithms, the representation languages can be conceived to channel to the computer represented problems.

Therefore we can also look at an Automatic Problem Solvers as at the Interpreter of the Representation Language in which the represented problem have been communicated to the computer, and, moreover, as at an Automatic Programmer.

Let us now present a first-cut design of an Automatic Problem Solver and a detailed description ofits mode of operation.

These concepts are graphically illustrated in Figure 3.

The control information and the problem information will constitute, expressed in an appropriate Representation Language, che Control-base and the Problem-base.

The automatic problem solver acts on these two bases of information as an interpreter and can perform the three basic activities of selection, search, and learning.

Its activities are controlled and organized by a monitor systhem. We now examine in detail the above outlined concepts, which are graphically illustrated in Figure 3. Let us begin with the Problem-Base.

The problem-base contains all informations on the problem to be solved (P) and its environment, which the man thinks as sufficient for the solution of P. In fact the problem-base can be built up as a set which contains: P (possibly many different representations), implicants of P, solved problems, other problems and reduction problems (having possibly P as their initial problem), simple and/or composite costs functions for the problems and the reduction problems of the problem-base, etc. The problem-base is first submitted to an ordering process which gives to the information contained in it an hierarchic ordering (e.g., a discrimination net or a tree) with the purpose of allowing an easier and more efficient exploration of it, whenever it is required by the monitor system.

The basic carachteristic of the problem-base is that it constitutes a dynamic set. In fact whenever the automatic problem solver achieves some useful results during its search activity, this is inserted

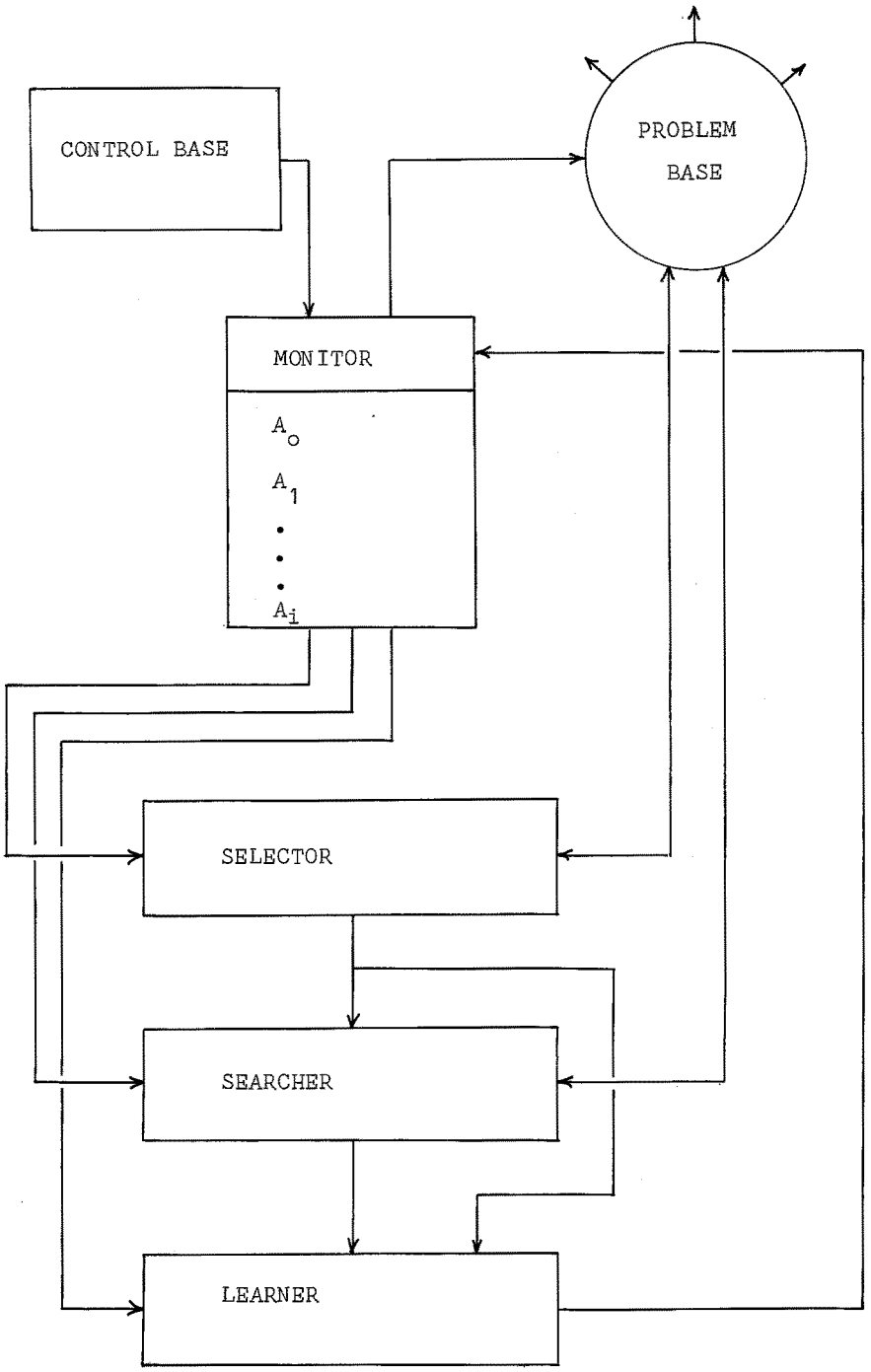

Figure 3. Schema of Automatic Problem Solver.

in the problem-base, at its right place, and can be used afterwards as a datum of the problem.

All problems which are contained in the problem-base may be either active problems or passive problems.

The active problems are those problems which can be considered as reduction operators when a reduction strategy is used in order to construct the solution of a problem.

The passive problems are the solved problems which constitute the "terminal nodes" of a reduction strategy.

We want now to emphasize that the man doesn't know exactly if the informations of the problem-base are enough, short or redundant for the solution of the problem; which of them are to be used; and, moreover, in which way must be organized the cooperation process between them in order to solve the problem.

We describe now the basic activities of the Monitor.

The Monitor can be considered as a system which can :

- give to the problem-base its appropriate hierarchic ordering;
- menage the generated attempts, by means of "interrupt", "activate" and "call garbage collector" signals.

The global activity of the Monitor system is fixed by the user of the automatic problem solver by means of the control information.

The ordering activity of the Monitor was already shown; we will only outline that it also controls the insertion in the problem-base of the new information arising during the solution of the problem.

The basic activities of the Monitor system are then the generation and the management of attempts. An attempt A is the set of information which contains the specifications of the selector and of the Searcher.

The attempts are generated by G in a temporal sequence, as a consequence of the initial control information and of the preceding learning activity. In particular it is clear that the first attempt A_o must be entirely specified in the control-base.

The generated attempts, A_o, A_1, A_2,..., are organized by the monitor in an appropriate hierarchic structure which must allow an easier management.

The attempts must be managed in such a way that they can be considered as coroutines: old attempts can be activated and new attempts cen be interrupted whenever it is useful. The attempt management is done by means of the control information. In Particular the "activate" and "interrupt" signals are generated on the basis of the computational effort done up to a certain point estimated by means of appropriate complexity measures defined in the control-base.
The attempt management must also provide a garbage-collector which distroies the old unuseful attempts, whenever it is necessary.

We can now examine in detail the content of the control-base. The control-base contains all informations that the man can draw from the intuitive problem and from his knowledge of the operating way of the artificial automatic problem solver in order to initialize the metasystem and to control its dynamic development.

In other words, the problem-solver, considered as a metasystem, is an artificial entity, existing outside the user, which has been provided by the artificial intelligence scientist as a general purpose tool.

However, the ingenuity and creativity of the user, can be exploited in order to "tune" such general purpose tool in the direction of the semantic domain from which the problem arises.

Hence, such initializstion (or specification) of the general purpose metasystem constitutes a way of defining a special purpose system. In particular the control-base must contain :
- the complete definition of the Monitor system and of the Learner;
- the first attempt A_o.

Now, we can precisely define the basic activities of the Selector, of the Searcher, and of the Learner.

The Selector can be considered as a system which can act on the problem-base and select: either a set of states to be expanded and a set of inputs for executing the expansions, if an expansion attempt is active at that time; or a set of passive problems which must take part to a reduction operation, and an set of active problems for performing the reduction, if a reduction attempt is active at that time. Of course a bidirectional exchange of informations between the Selec-

tor and the problem-base is provided.

The Searcher can be considered as a system which can act on that part of the problem base selected by the Selector and perform the expansion or reduction operation.

Of course a bidirectional exchange of information between the Searcher and the problem-base is also provided.

The Learner can be considered as a system which can provide the Monitor with all the requested information for a correct and efficient generation and menagement of attempts taking into account all the "past experience" done by the Selector, the Searcher and the Monitor itself.

The mode of operation of the whole system is then now quite easy to explain. By means of the activity of the Selector and of the Searcher the problem-base is incrementally expanded, i.e. new passive (solved) problems are constructed by means of the already existing ones and of the active problems. This process continues untill the problem P , that we wish to solve ,becomes also a solved one.

The efficiency of the system is provided by means of an adeguate generation and menagement of attempts, i.e. by the activity of the Monitor.

REFERENCES

1. Berge, C. "Théorie des Graphes et Ses Applications. Dunod, Paris, 1958.

2. Coray G. "Additive Features in Positional Games". ORME-IP-IRIA-NATO, Nato Advanced Study Institute on "Computer oriented learning, processes, Procédures informatique d'apprentissage". Bonas (Gers), France, August-September, 1974. Imprimé en France, IRIA, Domaine de Voluceau.

3. Ginzburg, A. "Algebraic Theory of Automata". Academic Press. Inc. New York. 1968.

4. Guida G., Mandrioli D., Paci A., and Somalvico M. "A Formal Framework for an Unitary Approachto the Theory of Problem-Solving". Proceedings IIASA Workshop on Artificial Intelligence - Question Answering Systems. Laxenburg Schloss, Austria. June, 1975.

5. Hartmanis, J. and Stearns, R.E. "Algebraic Structure Theory of Sequential Machines". Prentice-Hall, "Inc. Englewood Cliffs, N.Y. 1966.

6. Nilsson, N.J. "Problem-Solving Methods in Artificial Intelligence" Mc Graw-Hill Book Company, New York. 1971

7. Pohl,I. "Bi-Directional and Heuristic Search in Path Problems". SLAC Report N. 104. Stanford Linear Accelerator Center. Stanford University. Stanford, California, May, 1969.

8. Sangiovanni Vincentelli, A. and Somalvico M. "State-Space Approach in Problem-Solving Optimization". Politecnico di Milano. Istituto di Elettrotecnica ed Elettronica. Laboratorio di Calcolatori. Relazione Interna n. 73-15. MEMO-MP-AIM-12. May, 1973.

9. Minker J., and Vanderburg G.J. "State-Space, Problem-Reduction, and Theorem Proving-Some Relationships", Communication of the ACM, February 1975, Volume 18, Number 2, pp.107-115.