# SOUS CONSIDERATIONS CONCERNING THE PROBLEM BASE OF PURPOSEFUL SYSTEMS

Vitaly S. Losovsky
Institute of Uathematics
Academy of Sciences, USSR. Siberian Branch
Novosibirsk, 6J0090, USSR

## Abstract

The problem base structure for purposeful systems is considered. It cones out that predicate calculus is rather unwieldy for this purpose; semantic nets should be preferred. The main attention is paid to the convenience in representation of descriptive and operational information in the form of universal item (concept) in order to obtain natural and straightforward means for simulation of discrete and continuous processes, to provide proper interaction between them. The wide use of concept hierarchy is proposed. A single universal operator is introduced: the instantiation of concepts. The material is presented in informal way.

## Key Words

Problem solving, purposeful systems, data base, representation of knowledge, models of the world, semantic net, concept formation.

## Introduction

The notion of purposeful system seems to be a fundamental one in the field of Machine Intelligence. Consider a certain integrated model structure reflecting some pertinent properties of the physical world. One or more substructures are singled out representing active subjects. Some fixed configuration of the integrated structure is called a state. The structure can change along with time, so we must deal with a set of states. An initial state is fixed. In some or other way the goal state is characterized. Certain states are prohibited. The problem is to plan the behaviour of the active subjects, and then to execute the plan bringing the whole system in one of the goal states avoiding the prohibited ones and satisfying the given restrictions on time and resourses. The situation is really more complicated: we postulate the incomplete knowledge about the system. It results in so to say spontaneous changes of its state which cannot be predicted by the planning system. Then, not all active subjects axe liable to our control, Some of them may represent natural processes (seasonal changes during the year, eruption of volcanoes, snow-slips, etc.) or represent subjects being under control of some other system (or systems). At last, it is pragmatically more useful to investigate the behaviour of the large system. In this case we must abandon the hopes to obtain the true optimal solution.

We are going to study an artificial system represented in the form of computer software which is able to act purposefully in the foregoing sense, and external operational mode of which resembles the behaviour patterns of natural intelligence. The artificial system will not copy its neurophysiologies! processes; though some undoubted facts or empirically useful hypotheses obtained by the students of the living brain can be profitably taken into account during simulation. The creator of computer model has a possibility to organize its data base and operational sections in accordance with peculiarities of hard- and software at his disposal. It seems more appropriate to use the term "Machine Intelligence" instead of "Artificial Intelligence" in this case.

Apparently, the everyday reasoning problems will suit us all right at the beginning stage of the investigation.

There is no doubt that success in development of purposeful systems rather generally defined above will strongly stimulate the growth of effectiveness in many applications: natural language interaction with computer, CAI, control systems for integrated autonomous robots, models of large enterprises, etc.

## Known approaches

The development of purposeful systems (PS) attracts attention of researchers for a long time. The term "General Problem Solver" was coined by Newell, Shaw, and Simon in 1957 [2] , and since then ideas layed in its foundation were refind, improved during the decade and used in many implementations. J.McCarthy (1959) proposed developing of universal programs with common sense having the flexible strategy which can be effectively improved during the interaction with human creator (Advice TaJeer). A number of important considerations cun be found in [3j. A unique approach to the Droblem under discussion was evolved and deployed during efforts headed by U.Klykov, D.Pospelov and their colleagues; the results obtained can be found m [4] $_m$

Starting from 19&9 the progress in the art of purposeful systems design had been tied with the development of special programming languages for AI research

(C.Hewitt, [5J ). The power of this approach had been convincingly demonstrated by T.Winograd [6J • The use of pure mechanical theorem proving technique in PS-area is handicapped with serious practical limitations. The successful combination of theorem proving with the philosophy of GPS f1] along with many important mechanisms such as generalization of plans was evolved in SRI [7-9J . Among the most interesting recently issued papers is that of E.Sacerdoti [10l. A series of important results was obtained by the research group under R.F.Simmons in the University of Texas. It includes the investigation carried out by G.Hendrix which resulted in recommendations concerning the modelling of simultaneous interactive processes including the processes with continuous changes of model parameters.

The works mentioned above appreciably influenced the shaping of our approach.

## The Problem Description Language

It seems reasonable to express all useful findings in PS-area in the form of special PS-oriented language. Apparently it is the most effective way to generality. Next follows the discussion of the main premises for problem base construction of such a language.

Information about the physical world naturally falls into two categoriest descriptive and operational. The former declares names, properties and relations between objects specifying the model with names of properties and relations being predicates and that of objects—constants and variables. Operational part of the system changes a model. It results in deletion of some clauses and addition of some others. Situation can become even more complicated: the system can be acted upon by several operators concurrently; some can effect in gradual changes of values for various numeric objects. A series of operators can be under the control of planning system, others can be provoked by uncontrollable factors or be initiated by systems-competitors. Besides operators can have complex causative bonds.

All this aroses serious doubts in the practicability of the first order predicate calculus for description of such systems. Moreover, when the transition is performed from the semantic net representation to the uniform set of clauses being the problem base for resolution-type theorem provers, one can notice that some important information fades away. It is the non-homogeneous structure of semantic net which is associated with its objects. This organisation of data naturally provides straightforward acquisition of relevant sentences tied in contexts, discovering links between various contexts and the goal directed transformation of their content. The greater practicability of semantic nets was also emphasized earlier by R.F.Simmons [12J.

It seems reasonable to abandon the uniformity and formal elegance of logic in behalf of flexible universal programming language. This language must allow the manipulation of complex data structures and provide the convenient facilities for procedural embedding of semantics inherent in the processes being simulated. Trying to draw the expressive means of the language to those (hypothetically) used by humans one ought to work out uniform, or at least similar, format for descriptive and operational Information. At last the fundamental relations, such as sub-, superconcepts, cause-effect and time must simply and naturally be interpreted by the system*

Experimental heuristics oriented language for purposeful systems simulation (HEOPS) was decided to embed in LISP 1.5 paying the tribute to its universality, laconism and complex data structures allowed. The concept classes hierarchy and time synchronization for processes in HEOPS is borrowed substantially from SIMULA [13].

## The Systems Problem Base

The problem base of the HEOPS includes two basic parts: declarative section and operational space. Declarative section includes the definitions of all concepts and relations currently existing in the system. In the operational space is "builded" the initial situation for a given problem. Its solution coprises several transformations through the application of the sole operation: "instantiate" one or other definition from the declarative section. The boundary between the two sections is rather conventional: it is allowed to transfer non-instantiated definitions from declarative to operational parti one can imagine the inverse too: synthesis of some new definition in the operational space which is followed by its transfer to the declarative section for further use.

Two types of objects are liable to declaration: relations and concepts.

### Declaration of Relations

It is instructive to think about relations as arcs of semantic net. However the number of their arguments not necessary equals to two. Extra arguments play the role of relation modifiers. Thus, let A and B be two nodes. The relation (DISTANCE A B I) means: "The distance is defined between objects designated by the names A and B, and this distance equals to X". It is convenient in computer to gather all relations of the semantic net having identical first arguments and place them on the property list of the corresponding node, deleting its name from the explicit forms of the relations.

When the declaration of some relation is interpreted the relevant information defining the semantics of the given

relation is put on the property list of the choosen relation name in the form of traditional property-value pairs:

- The number and also the necessary properties for arguments of the relation are specified. In the foregoing example A and B should be physical bodies, and X—a number.

- The pointer to the "superrelation" is inserted. Thus, the predicate SAT can have as its superrelation the predicate COMMUNICATE.

- The pointer may be inserted to the list of "subrelations" currently existing in the system. The predicate COMMUNICATE thus can contain in its subrelations[1] list: SAY, WRITE, SUGGEST, HINT, etc.

Declaration of Concepts

Concepts are the main object type in the HEOPS. Declared are classes of concepts. Each class gets its name, for example, ROOM, MONKEY, GET, APPROACH, etc.

In class declaration generally are included:

1. Semantic net of applicability (application pattern), probably, with pointers to super- and subconcepta.
2. Activation pattern.
3. Operational section.
   - 3-1 Delete-list.
   - 3.2 Add-list.
   - 3.3 Recomputetion formulae.
   - 3.4 Notice-list.
4. Passivation pattern.
5. Suppression pattern.

Such classes usually represent a process. The class with sections 2—4 absent is called assertional and represents simple or compound entity—the part of semantic net with some set of nodes and relations between them.

The nodes used in declarations fall into one of the following categories:

P: - free node of simple type: during instantiation can be replaced with own, terminal, or classified node.

G: - free node of generic type; not disappear when instantiated, but produces moreover the finite set of nodes (F:-type) each of which can be further instantiated.

T: - terminal node; its semantics comprises solely its name, it cannot have any other properties and cannot be further instantiated.

C: - classified node, i.e. the node having its class declaration in the system, when instantiated requires pattern matching upon existing net.

Instantiation of Concepts

The first argument of the aforementioned operation of instantiation is the name of concept declared (dynamically) earlier. The second argument is the list of instantiation parameters. Thus, the concept MONKEY can obtain after instantiation the fixed name, specific location in some room; it can have some object in its hands. It can even start moving at once if in its class declaration the appropriate operational section is present.

During instantiation the application pattern of the given concept is matched upon the current configuration of the operational space: explicitly—through instantiation parameters and implicitly—upon the nodes and arcs of the currently existing semantic net. The relevant information is choosen in this case by the concept body itself. Instantiation results in the new state of the operational space with new concept "implanted" there. Otherwise inapplicability is indicated. The more interesting mode of operation results in the sequence of recursive calls for instantiation of some other concepts in order to eliminate the discrepancies found. This process is guided mainly by special pointers extant in declarations and relating some relations with concepts responsible for their origination, support, or removal.

Operational information is reflected in the proposed problem base in two ways: in ready to use procedural form as operational section (3) of concept class declaration, and as a so called possibility hint. In the later case in the applicability pattern (1), add-list (3.2) of class declaration, or somewhere in the semantic net of the operational space are inserted nodes—reflections of certain concepts-processes. Thus the node MONKEY can be linked with possibility node MOVE, probably with some restrictions on the extent of mobility (say, remaining on some solid surface). This means that the MONKEY can move. In order for it really to move the concept corresponding to this node of the net must be instantiated using the declaration of MOVE.

Instantiation can be done partially: i.e. implanted concept can retain some of its free nodes. As for example, during the process of getting bananas in the well known AI problem, it may turn out that the distance between the monkey and bananas can be diminished if the monkey will be standing on something rising above the floor. The node designating this "something" remains free until it eventually becomes instantiated to, say, a box if the later is available.

Time dependency

Time and cause-effect bonds deserve special discussion. In HEOPS it is planned to use the sequence list akin to that introduced in SIMULA [131. Recomputation formulae (3-3) are used in class declarations in order to fix the crucial moments of time. The notice-list (3.4) contains the pointers to the instantiated concepts

in the operational space which must be activated after fulfilment of certain conditions. The provision of special machinery for process interception at arbitrary moment with return to the arbitrary continuation point is not planned. Instead the processing of concepts (also quasiparallel processes') is arranged so that exit and reentrance are done in precpecified points; special provisions must be used to retain the contexts' value.

## Super- and Subconcept Relations

During the instantiation process the need in certain more general relations can arise. Each instantiated node in the semantic net has a pointer to its class name. For example, the specific monkey named Cheetah remains to be a monkey having all its properties. A MONKEY is a subconcept of ANIMAL, and the fact that any animal needs food need not be repeated for all subconcepts of the concept ANIMAL. An appeal to more general concepts proved useful in some situation can be fixed in instantiated subconcept as explicit addition to the semantic net.

## Conclusion

Organizational principles of the HEOFS language problem base for simulation of purposeful systems, informally discussed in this paper, will be specified more accurately in future. Under current consideration are also the principles of its computer implementation. In particular, we are faced with working out the efficient methods for contexts[1] bookkeeping to hold the nodes' values after instantiation and preserving the convenient access to the hierarchy of concepts. One of the crucial points in future investigations will be the realization of effective bootstrapping procedure for the system both in descriptive and operational aspects. This process can be regarded as system's learning in one of the most important mode: algorithmic evolution.

The simulation of the system is being done on HESM-6 computer with DIAPACK OS. The host language is LISP 1.5 [14,15] which is run under TELELISP monitor system implemented in the Institute of Mathematics, Siberian Br. Acad. Sci. USSR, provided with archives and text editor which make it possible to interact with the machine from distant terminal.

## References


1. George W.Ernst. Allen Newell,GPS: A Case Study in Generality and Problem Solving, Academic Press, NY, 1969.

2. A.Newell, J.C.Shaw, H.A.Simon, Preliminary Description of General Problem Solving Program-1, CIP Working Paper No,7.Carnegie Institute or Tecnnoiogy, Pittsburg, Pennsylvania, 1967.

3. J.McCarthy, P.J.Hayes, Some Philosophical Problems from the Standpoint of Artificial Intelligence, Machine lutelli-fence4, B.Meltzer, D.Michie Cede) Edinburgh TI.P., 1969.

4. U.I.Klykov, Situational Control in Large Systems, "Energia", Moscow,1974 (in Russian)•

5. C.Hewitt, PLANNER: A Language for Proving Theorems in Robots, Proceedings of the Int. Joint Conf. on Ai Bedford. Mass, Mitre Corp., I9t>9.

6. T.Winograd, Understanding Natural Language, Edinburgh University Press,1972.

7. R.E.Pikes, N.J.Nilsson, STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving, Artificial Intelligence. 2. 1971, pp.189-208.

8. R.E.Pikes, P.E.Hart, N.J.Nilsson, Learning and Executing Generalized Robot Plans, Artificial Intelligence. 3.1972.

9. R.E.Fikes, P.E.Hart. N.J.Nilsson, Some New Directions in Robot Problem Solving. Machine Intelligence 7 , B.Meltzer and D.Michie leas), wliey, New York,1972.

10. Earl D.Sacerdoti, Planning in a Hierarchy of Abstraction Spaces, Artificial Intelligence. 5, 1974, pp.115-135.

11. Gary G.Hendrix, Modelling Simultaneous Actions and Continuous Processes, Artificial Intelligence. 4, 1973,pp. 145-ittO.

12. Robert F.Simmons, Bertram C.Bruce, Some Relations Between Predicate Calculus and Semantic Net Representation of Discourse, Sec.Int.JT.conf.AI,London, 1971, pp. 524-530.

13. O.Dahl, B.Myhrhaug, K.Nygaard, SIMULA 67, Common Base Language, Oslo,1968.

14. S.S.Lavrov, G.S.Silagadze, The Input Language and Interpreter of LISP-Based Programming System for HESM-6, Comp. Center, Acad. Sci. USSR, Moscow, 1969 (in Russian).

15. V.M.Ufa, Elaboration of the Programming System LISP—BESM-6, in Symbol Information Processing, issue 1. Comp. center, Acad. Sci. USSR, Moscow, 1973 (in Russian).