Invited Paper

Towards Integration of Heterogeneous Knowledge for Highly Autonomous Analysis of Dynamical Systems— Preliminary Report from the PSX Project

TOYOAKI NISHIDA*

Analysis of dynamical systems is one of the core issues in engineering problem solving. The purpose of the PSX project is to build a highly autonomous system for dynamical systems analysis, called PSX, which can investigate the characteristics of given dynamical systems without external assistance. The major issue in the PSX project is the integration of heterogeneous knowledge for dynamical systems analysis.

In this paper, I discuss a couple of issues involved in the integration of heterogeneous knowledge, namely, the vertical and horizontal coordination problems. The vertical coordination problem is concerned with the coordination of knowledge units at different levels of abstraction. The horizontal coordination problem is concerned with the organization of widespread knowledge into a coherent body.

I will first discuss the vertical coordination problem on the basis of our experiences with PSX2NL, which analyzes the qualitative behavior of ordinary differential equations in the two-dimensional phase space. I demonstrate the complementary roles played by qualitative and quantitative analysis in PSX2NL. I will then present the notion of the *knowledge community*, a society of interacting agents, as a framework for horizontal coordination, and will discuss how the horizontal coordination problem can be handled within this framework.

1. Introduction

Analysis of dynamical systems is one of the core issues in engineering problem solving. Almost every kind of engineering problem solving requires understanding of the characteristics of certain kinds of mathematical model, such as differential equations, so that artifacts may be properly designed, monitored, controlled, or diagnosed.

Although computer technology has made great advances in scientific and engineering computation, its most successful applications are still limited to low-level information processing, such as numerical calculation, visualization, or at best symbolic computation. High-level decision making still depends on human experts. Thus, nobody can take advantage of the computational power of conventional tools unless she or he has ample knowledge about dynamical systems analysis, as well as the tools, or can receive the assistance of domain experts.

The purpose of the PSX project is to build a highly autonomous system for dynamical systems analysis, called PSX,[1] which can investigate the characteristics of given dynamical systems without external assistance. The major issues addressed in the PSX project are twofold: (1) construction of a knowledge base that covers a wide range of mathematical concepts and techniques in dynamical systems analysis, and (2) integration of knowledge-based methods with existing numerical and symbolic computation techniques.

In this paper, I discuss a couple of issues involved in the integration of heterogeneous knowledge, namely, the vertical and horizontal coordination problems.

The vertical coordination problem is concerned with the coordination of knowledge units at different levels of abstraction. To achieve high autonomy, the following questions need to be addressed:

- How can abstract knowledge about dynamical systems analysis be used to plan, control and monitor numerical analysis, interpret its results, and modify its plans?
- What is an adequate representation for interfacing knowledge units at different levels of abstraction?

The horizontal coordination problem is concerned with the organization of widespread knowledge into a coherent body. The major issues here are as follows:

*Department of Information Science, Kyoto University, Sakyo-ku, Kyoto 606-01, Japan.

[1] PSX stands for "Phase Space eXplorer."

- A common protocol that allows knowledge units to exchange information
- The syntax and semantics of a knowledge representation language for describing objects and concepts referred to in dynamical systems analysis
- A system of common terminology written in the above knowledge representation language.

In the rest of this paper, I will first discuss the vertical coordination problem on the basis of our experiences with PSX2NL, which analyzes the qualitative behavior of ordinary differential equations in the two-dimensional phase space. I demonstrate the complementary roles played by qualitative and quantitative analysis in PSX2NL. I will then present the notion of the *knowledge community*, a society of interacting agents, as a framework for horizontal coordination, and will discuss how the horizontal coordination problem can be handled within this framework.

2. Vertical Coordination—Interplay between Different Levels of Abstraction

2.1 Analysis of Ordinary Differential Equations

In scientific and engineering problem solving, many important dynamical systems are specified as a system of first-order ordinary differential equations (ODEs) of the form:

$$\frac{dx}{dt}=f(x), \quad f: R^n \rightarrow R^n \tag{1}$$

on a finite number of time-dependent variables $x(t)=\{x_1(t), \ldots, x_n(t)\}(t \in R)$. Equation (1) is said to be linear if f is linear, and nonlinear otherwise. Systems of linear ODEs can be solved analytically in the sense that it is possible to compute the representation of x as a function of t provided that the eigenvalues and eigenvectors of an $n \times n$ coefficient matrix can be obtained. In addition, the behavior of systems of linear ODEs is simple and can be classified into a small number of categories according to the types of eigenvalues of their coefficient matrix. Systems of nonlinear ODEs are not that simple. They cannot always be solved analytically, and there are infinitely many varieties of behavior. The properties of nonlinear ODEs are studied in applied mathematics, from qualitative points of view. Dynamical systems theories [Hirsch and Smale, 1974; Guckenheimer and Holmes, 1983] provide a mathematical basis for answering such questions as:

- How does a given system of ODEs behave after a long run?
- How stable is a given system of ODEs under perturbations?
- How does the behavior pattern of a parameterized system of ODEs change as some parameters are changed?

According to an applied mathematician,[1] qualitative analysis investigates the following issues in turn:

- How many attractors there are
- Their approximate locations in the phase space
- The topological structure of the attractors
- The approximate locations of basins
- The structure of the basins.

Mathematicians seek the answers to these questions by combining numerical calculation, formula manipulation, and mathematical reasoning. To mimic such intellectual activities by computers, it is necessary to develop a framework for the integration of heterogeneous knowledge, at least vertical coordination between knowledge at different levels of abstraction.

In what follows, I would like to survey how the vertical coordination problem has been handled in PSX2NL [Nishida, *et al.*, 1991; Nishida and Doshita, 1991], a program that analyzes the qualitative behavior of systems of ODEs defined in the two-dimensional phase space.[2]

2.2 Preliminaries of Qualitative Analysis of ODEs

Let us first introduce the basic concepts and methods of dynamical systems analysis employed in PSX2NL. In this subsection, I will illustrate the basic ideas by using an example. Consider the following system of nonlinear ODEs:

$$\begin{cases} \dfrac{dx}{dt}=-2x^3+2x+2y \\ \dfrac{dy}{dt}=-x. \end{cases} \tag{2}$$

This system of ODEs is called Van der Pol's equation, and is concerned with two independent *state variables* x and y which vary as a function of time t. Given a particular pair of values for these variables, the vector of state evolution $(\frac{dx}{dt}, \frac{dy}{dt})$ is determined from the right-hand side. In this sense, the system of ODEs (2) introduces the *vector field* in the phase space, as shown in Fig. 1(a).

In ordinary situations, the vector field implicitly specifies the *phase portrait*, a set of non-intersecting directed curves such that each directed curve is tangent to the vector field. Part of the phase portrait for Van der Pol's equation is shown in Fig. 1(b). Each curve, called an *orbit*, represents a solution to the given system of ODEs under some initial condition. One thing to note is that each directed curve maps points in the phase space as a function of time. In this sense a system of ODEs is said to introduce a *flow* in the phase space.

At the origin, there is a special orbit consisting of a single point. At this point, the right-hand side of the

[1] H. B. Stewart in a lecture at the Department of Electrical Engineering, Kyoto University on July 11, 1989.

[2] I should note, though, that we have recently started an effort to implement PSX3, an extension of PSX2NL that can handle systems of ODEs defined in the three-dimensional phase space. Preliminary results are reported in another paper [Mizutani, *et al.*, 1992].

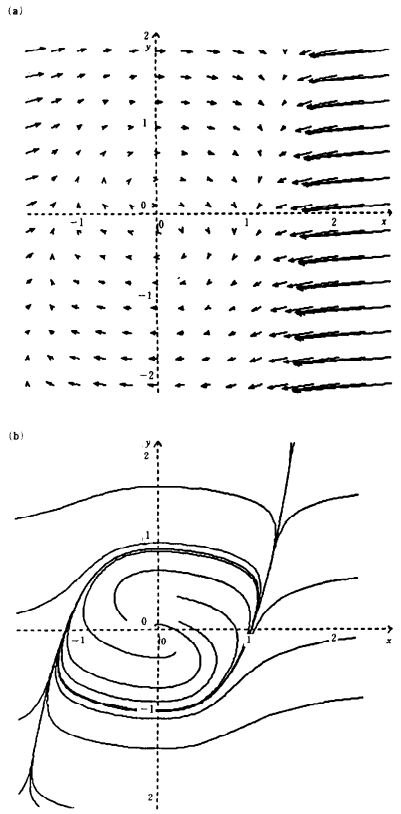


Fig. 1 Vector Field and the Phase Portrait of Van der Pol's Equation (2).

given system of ODEs becomes zero, and hence the state never evolves over time. Such an orbit is called a *fixed point*.

It is known that every orbit on two-dimensional phase planes either diverges to infinity, or approaches a fixed point or a cyclic orbit called a *limit cycle* as $t \to \pm\infty$.

2.3 Overview of PSX2NL

Given a system of ODEs, we would like PSX2NL to classify orbits according to their asymptotic properties, by making use of available computational resources. The central concern here is the vertical coordination problem: coordinating processes that handle information at different levels of abstraction.[1] In particular, we have to address the following computational issues:

1. Representation of flow: we have to seek an adequate representation of flow, one that is derivable by numerical or symbolic computation and that allows qualitative behavior to be derived;
2. Control structure: we have to seek a control structure that allows low-level computation (numerical and symbolic computation) and high-level qualitative reasoning to interact.

For the first issue, we represent flow as a collection of *flow mappings*, each of which represents a bundle of nearby orbits. For the second issue, we have employed a blackboard model and have implemented top-down and bottom-up analysis procedures on this basis.

Given a system of ODEs and a region of analysis,[1] PSX2NL goes through the following stages:

(step 1) Partition the given region of analysis into smaller regions called *cells*.[2]

(step 2) Analyze the flow in each cell and characterize it as a collection of flow mappings.

(step 3) Put together the flow mappings for each cell and derive long-term behavior.

In the following two subsections, I will describe our representation and control structure in more detail.

2.4 Reasoning about Qualitative Behavior with Flow Mappings

Roughly, we characterize the flow in a bounded region in terms of how each point on the boundary is mapped by the flow in the region. A point on the boundary where an orbit is coming into the region is mapped either to another point on the boundary, or to a fixed point or a limit cycle in the region. Similar proposition holds for points where orbits come out of the region. We aggregate continuous points on the boundary provided they are mapped to or from an open continuous segment of the boundary, the same fixed point, or the same limit cycle, and represent the flow as a collection of flow mappings between these continuous segments or fixed points or limit cycles. This operation corresponds to aggregating nearby orbits in the cell, according to their qualitative properties in the cell.

Suppose we are to represent the local flow in region *ABEF* in Fig. 2(a). As shown in Fig. 2(b), some points on segment $\overline{E, F}$ are mapped from the fixed point X in the region, while other come from other points on the boundary. Although all points on the open segment $\Phi_1(Q)$, F come from other points on the boundary, the source is not one continuous segment, but a sum of four open segments: $\overline{\Phi_1^{-1}(P), Q}$, $\overline{\Phi_1^{-1}(S), P}$, $\overline{\Phi_1^{-1}(R), S}$, and $\overline{F, R}$, and three landmarks delimiting them: $\Phi_1(P)$, $\Phi_1(S)$, and $\Phi_1(R)$, so we represent the flow related to $\overline{\Phi_1(Q), F}$ as a collection of seven flow mappings:

$$\overline{\Phi_1^{-1}(P), Q} \to \overline{\Phi_1(Q), \Phi_1(P)}$$
$$\oplus\ \overline{\Phi_1^{-1}(S), P} \to \overline{\Phi_1(P), \Phi_1(S)}$$
$$\oplus\ \overline{\Phi_1^{-1}(R), S} \to \overline{\Phi_1(S), \Phi_1(R)}$$
$$\oplus\ \overline{F, R} \to \overline{\Phi_1(R), F}$$

[1] Horizontal coordination is less an issue, because the knowledge used there may not be widespread.

[1] The current implementation of PSX2NL can analyze the behavior of a given system of ODEs only in a bounded region of the phase space.

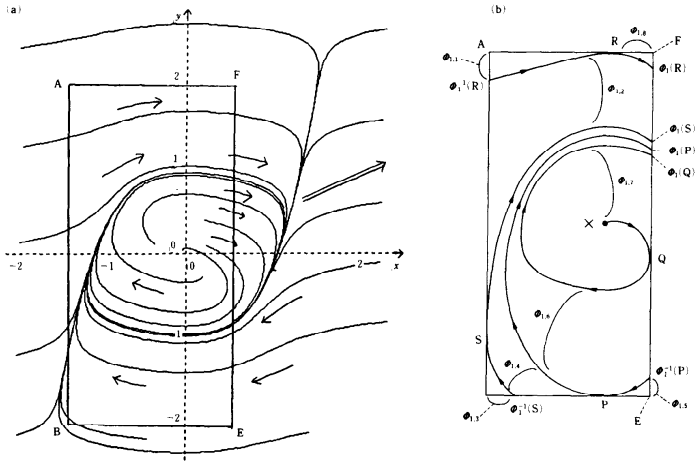


Fig. 2 Qualitative Analysis of the Local Flow of Van der Pol's Equation (2).

$\oplus P \rightarrow \Phi_1(P)$

$\oplus S \rightarrow \Phi_1(S)$

$\oplus R \rightarrow \Phi_1(R)$.

Thus, the flow in *ABEF* is represented as a sum of sixteen flow mappings, as follows:[1]

$$\Phi_1 = \Phi_{1,1}: \overline{A, \Phi_1^{-1}(R)} \rightarrow \overline{R, A} \tag{3}$$
$$\oplus \Phi_{1,2}: \overline{\Phi_1^{-1}(R), S} \rightarrow \overline{\Phi_1(S), \Phi_1(R)}$$
$$\oplus \Phi_{1,3}: \overline{B, \Phi_1^{-1}(S)} \rightarrow \overline{S, B}$$
$$\oplus \Phi_{1,4}: \overline{\Phi_1^{-1}(S), P} \rightarrow \overline{\Phi_1(P), \Phi_1(S)}$$
$$\oplus \Phi_{1,5}: \overline{E, \Phi_1^{-1}(P)} \rightarrow \overline{PE}$$
$$\oplus \Phi_{1,6}: \overline{\Phi_1^{-1}(P), Q} \rightarrow \overline{\Phi_1(Q), \Phi_1(P)}$$
$$\oplus \Phi_{1,7}: X \rightarrow \overline{Q, \Phi_1(Q)} \oplus \Phi_{1,8}: \overline{F, R} \rightarrow \overline{\Phi_1(R), F}$$
$$\oplus \Phi_1^{-1}(R) \rightarrow R \oplus R \rightarrow \Phi_1(R)$$
$$\oplus \Phi_1^{-1}(S) \rightarrow S \oplus S \rightarrow \Phi_1(S)$$
$$\oplus \Phi_1^{-1}(P) \rightarrow P \oplus X \rightarrow Q$$
$$\oplus Q \rightarrow \Phi_1(Q) \oplus X.$$

In order to construct a collection of flow mappings for a cell, we seek points at which the orbit is tangent to the boundary, as well as some other geometric clues such as the location and type of fixed points. Let us call a point on the boundary such that the orbit is tangent to the boundary a *point of contact*. Point of contact are further classified into *convex nodes* and *concave nodes*. At a convex node, the orbit lies outside the cell immediately before and after it reaches the point of contact. A concave node is defined similarly.

The qualitative behavior of a given system of ODEs is obtained by putting together a collection of flow mappings for each cell and examining the structure of the resulting collection of flow mappings. For example, the flow in the cell *FECD* of the phase space for Van der Pol's equation (see the top of Fig. 3) can be represented as:

$$\Phi_2: \overline{C, D, F, Q} \rightarrow \overline{Q, E, C}. \tag{4}$$

If we combine two sets of flow mappings for the two cells *ABEF* and *FECD* of the phase space for Van der Pol's equation (see the center and bottom of Fig. 3), we obtain

$$\Phi_2(\overline{F, Q}) = \overline{\Phi_2(F), Q} \subset \overline{\Phi_1^{-1}(P), Q} \tag{5}$$
$$\Phi_1(\overline{\Phi_1^{-1}(P), Q}) = \overline{\Phi_1(Q), \Phi_1(P)} \subset \overline{F, Q}, \tag{6}$$

and hence

$$\Phi_1 \circ \Phi_2(\overline{F, Q}) \subset \overline{F, Q}. \tag{7}$$

Fromula (7) means that all orbits passing through the interval $\overline{F, Q}$ never leave the interval, and hence we can conclude from the Poincaré-Bendixon theorem[1] that there exists an attracting bundle Φ_C of orbits containing at least one limit cycle that is transverse to $\Phi_1 \circ \Phi_2(\overline{F, Q})$.[2] It also follows that all orbits passing through the interval $\overline{F, Q}$ tend towards Φ_C as $t \rightarrow \infty$. See Fig. 3 for the entire derivation.

2.5 Top-down and Bottom-up Analysis on a Blackboard Model

For the procedure illustrated in the previous subsection to work, numerical and symbolic computation and a high-level qualitative reasoning procedure should closely interact with each other. It should be noted that neither low-level procedures nor high-level procedures are complete; high-level procedures are incomplete in the sense that they cannot yield any conclusion unless

[1]The first eight are essential. The others involve some subtlety, but this is not critical to our discussion.

[1]See p. 248 of [Hirsch and Smale, 1974] for more details.

[2]Note that $\Phi_1 \circ \Phi_2(\overline{F, Q}) = \Phi_1(Q), \Phi_1 \circ \Phi_2(F)$.

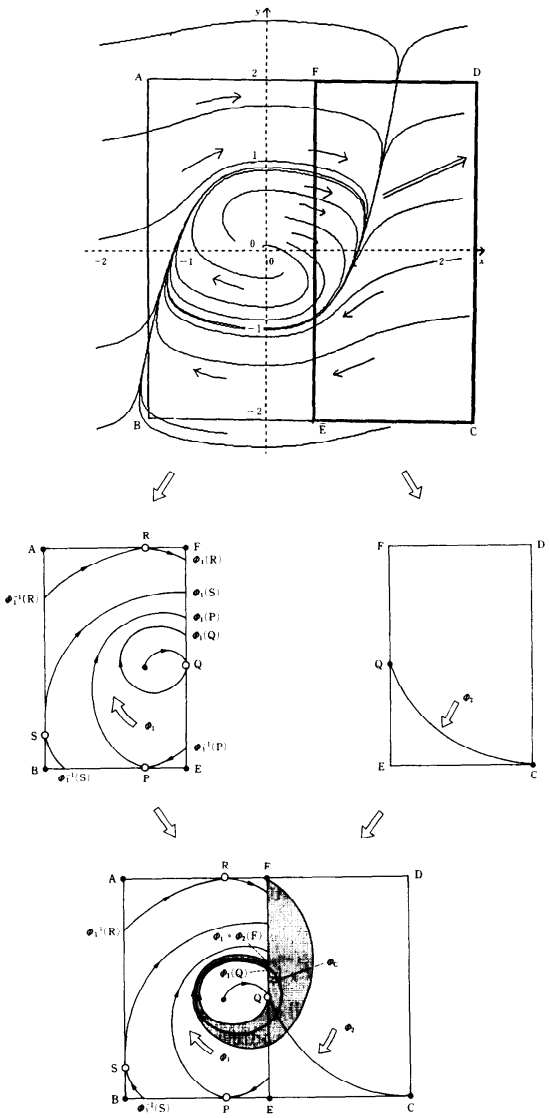


Fig. 3 Reasoning about Qualitative Behavior with Flow Mappings.

evidence is provided, whereas low-level procedures are incomplete in the sense that they cannot determine how accurate an answer they need to produce in order to avoid missing some important solution. In order to facilitate intensive coupling of the processes at different levels of abstraction, we have employed a blackboard model as a basis of a control scheme (Fig. 4). Thus, the system has a shared memory space (the blackboard) and a library of processes (KSs: knowledge sources). KSs interact indirectly by updating the contents of the blackboard.

The blackboard model makes it possible to implement multiple strategies. In normal situations, PSX2NL examines the flow in a bottom-up manner, attempting to build a qualitative description according to a prescribed fixed sequence: PSX2NL first examines whether the given flow has one or more fixed points; if so, it determines their type. If there is more than one fixed point in the given region, PSX2NL partitions the given region into several cells so that at most one fixed point may be contained in each cell; it then examines the geometric features of the flow, traces key orbits by numerical integration if necessary, generates a set of flow mappings for the given region, investigates the properties of the flow mappings, and derives conclusions about the qualitative behavior of the given region. Thus, an abstract description is gradually constructed from less abstract descriptions.

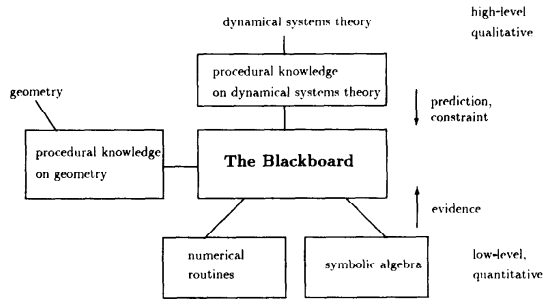


Fig. 4 A Blackboard Model as a Basis of the Control Scheme of PSX2NL.

If something goes wrong and the standard sequence turns out to be intractable, the analysis process switches to the top-down mode, trying to find the most plausible interpretation that matches the observations made so far. As a knowledge source, PSX2NL uses a *flow grammar*, which is a grammatical description of all possible behavior patterns.

The flow grammar provides PSX2NL with theoretical constraints, allowing it to operate in a top-down manner. More specifically, the flow grammar allows PSX2NL

1. To predict the existence of key orbits and to plan numerical computation to find their location: for example, given an observation as illustrated in Fig. 5(a), PSX2NL predicts the existence of a saddle node and attempts to find the location in the phase space (Fig. 5(b)), infers the location at which invariant manifolds of the predicted saddle node intersect the boundary of a given region by narrowing down the envelop enclosing the saddle node and associated invariant manifolds (Fig. 5(c)), and finally it infers the location of the saddle node (Fig. 5(d));
2. To focus numerical computation: for example, if there is more than one possible interpretation of an observation (upper half of Fig. 6), PSX2NL

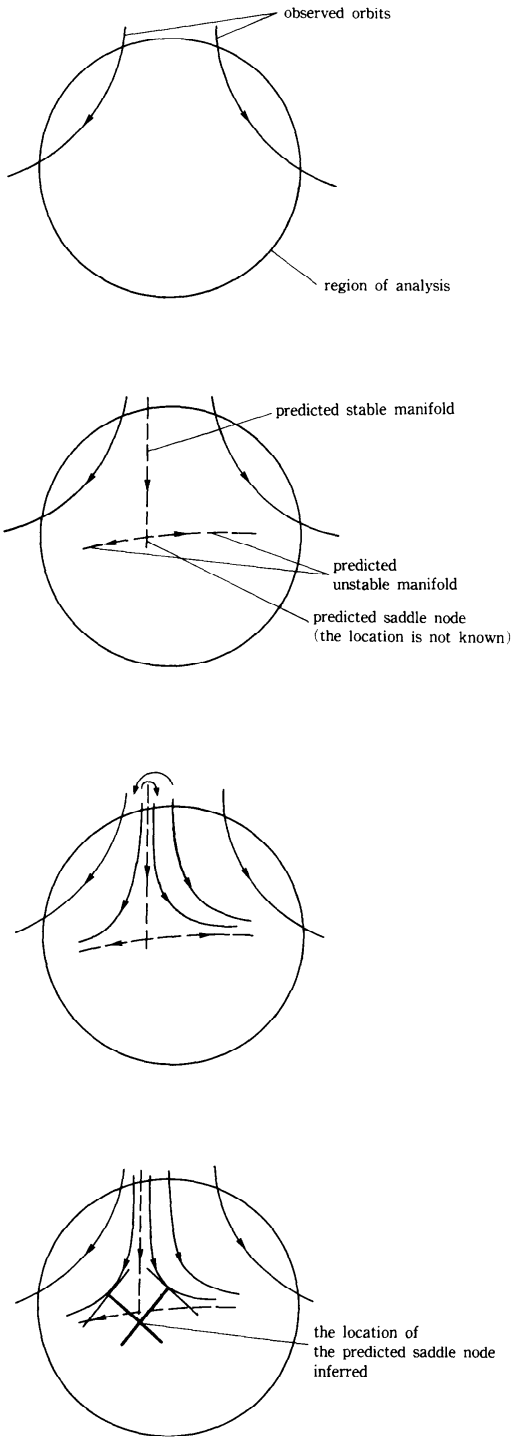


Fig. 5 Planning Numerical Computation.

will plan numerical computation that is expected to resolve the ambiguity (lower half of Fig. 6);

3. To rearrange the analysis process when an unexpected result is obtained: for example, when a symptom of an unexpected limit cycle is observed, as in Fig. 7(a), PSX2NL will divide a region into cells and try to prove the existence of a limit cycle by numerical and symbolic computation (Fig. 7(b));
4. To detect inconsistent numerical results and propose a plausible interpretation: for example, non-intersection constraints of orbits may be violated because of numerical errors, as in Fig. 8(a); in such cases, PSX2NL suggests the most plausible interpretation (Fig. 8(b));

To allow the complex control structure to be implemented easily, we have employed the following features:

1. Explicit representation of the control structure on the blackboard so that KSs can directly access and manipulate it;
2. Uniform representation of all objects on the blackboard as attribute-value pairs.

Our approach to designing and implementing PSX2NL has the following limitations:

1. All knowledge used for problem solving must be represented in a procedural form as KSs;
2. The terminology for representing objects and relations has not been carefully designed;
3. No attempt has been made to incorporate intuitive guidance, possibly obtained from statistics.

Thus, we have not addressed the horizontal coordination problem in designing and implementing PSX2NL.

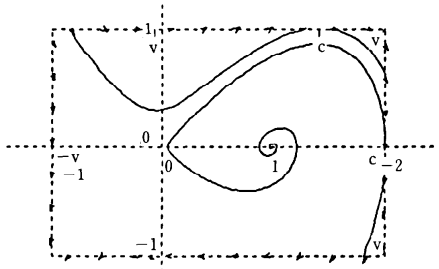
3. Towards Horizontal Coordination—The Knowledge Community

To make significant progress beyond what has been achieved in PSX2NL and its siblings, it is necessary to incorporate the large-scale knowledge on dynamical systems and their analysis techniques that has been set down in handbooks (e.g., [Zwillinger, 1989]). To this end, we have recently started the *knowledge community* project, whose aim is to establish a computational framework for integrating heterogeneous knowledge as a collection of autonomous agents. This section presents an overview of the project. Since the project is still in a preliminary phase, the focus will be on methodological and conceptual aspects, highlighting the scope of the project as well as major technical goals.

3.1 Approaches Taken for the Knowledge Community

Generally, it is quite expensive to build and maintain a single monolithic procedure that is very versatile. It is much easier to implement progrms that perform single tasks. Hence, a more attractive approach would be to

(a) Observation



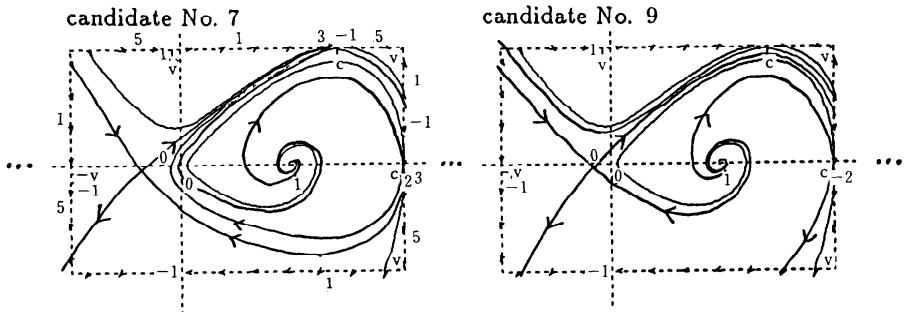
search for interpretation

f-rep inferred: ((− 1) (+ 5 1) (− 5 1) (+ 5) (− −1 1) (+ 5))

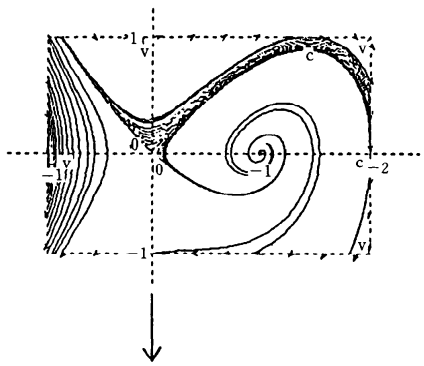
(b) Searching for flow patterns which explain the observation; candidates are listed from the simplest ones; in this case, twelve equally simple candidates are found;

```
12 patterns match.
> their f-reps are:
No. 1: ((- -1 1) (+ 5) (- -1 1) (+ 5 1) (- 5 1) (+ 5))
No. 2: ((- -1 1) (+ 5 3) (- 1) (+ 5 1) (- 5 3 1) (+ 5))
No. 3: ((- -1 3 1) (+ 5) (- 1) (+ 5 3 1) (- 5 1) (+ 5))
No. 4: ((- 5 3 1) (+ 5) (- 1 -1 1) (+ 5 5 3) (- 1) (+ 5 1))
No. 5: ((- 5 1) (+ 5) (- 3 1 -1 1) (+ 5 5) (- 1) (+ 5 3 1))
No. 6: ((- 5 -1 1) (+ 5) (- -1 1) (+ 5) (- -1 1) (+ 5 1))
No. 7: ((- -1 1) (+ 5) (- -1 3 1) (+ 5 1) (- 5 1) (+ 5 3))
No. 8: ((- 5 3 -1 1) (+ 5) (- -1 1) (+ 5 3) (- 1) (+ 5 1))
No. 9: ((- 5 -1 1) (+ 5) (- -1 3 1) (+ 5) (- 1) (+ 5 3 1))
No. 10: ((- 5 1) (+ 5) (- 3 -1 1) (+ 5) (- -1 1) (+ 5 3 1))
No. 11: ((- -1 5 1) (+ 5) (- 3 1) (+ 5 1) (- 5 1) (+ 5 3 1))
No. 12: ((- 5 1) (+ 5) (- 3 -1 3 1) (+ 5) (- 1) (+ 5 3 3 1))
```

candidate No. 7 candidate No. 9

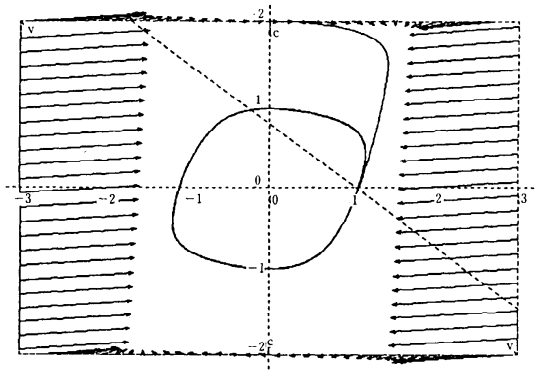


(c) In order to resolve ambiguity, numerical computation is planned and executed;

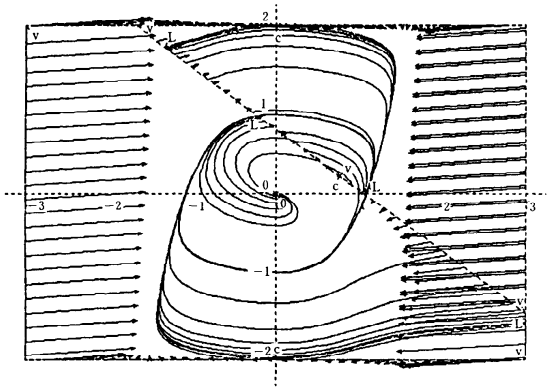


(d) Candidate No. 9 is selected as the most plausible interpretation.

Fig. 6 Focusing Numerical Computation.



(a) Symptom of unexpected limit cycle is detected
⇒ divide the region



(b) Analyze flow in each region individually and merge the results

Fig. 7 Rearranging the Analysis Process When an Unexpected Result Is Obtained.

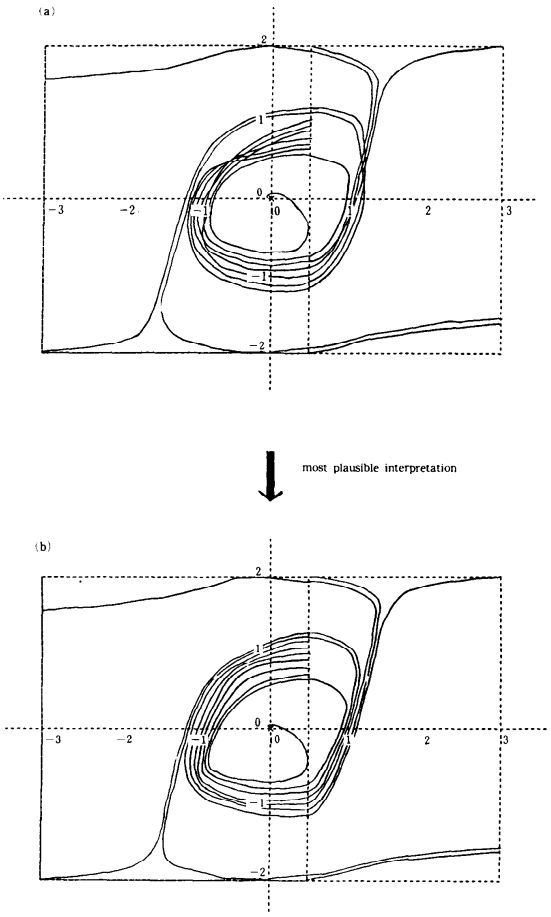


Fig. 8 Correcting Inconsistency Due to Numerical Errors.

build a large knowledge base as a large collection of simple processes, as suggested by Minsky [Minsky, 1985].

To seek how this approach works, consider the following engineering question:

"How can I remove the mist from (the surface of) a glass (without directly touching it)?" (8)

This is a typical question[1] that calls for horizontal coordination, because a wide range of knowledge must be accessed in order to answer it. A problem reduction method, familiar as a classic AI technique, suggests repeatedly decomposing complex questions like this into less complex questions until all of the subproblems can be solved by primitive problem solving procedures. For example, question (8) may be decomposed into subquestions as shown in Table 1.

[1]This is *not* a typical question in the context of dynamical systems analysis, but this kind of question may be asked in a wider context in engineering.

Table 1 Questions Asked in the Course of Answering (8) by Using Problem Reduction.

- Questions about the pjhysical world:
  - —What is the physical entity of mist on glass?
  - —Where does it come from?
  - —What parameters are relevant to heat radiation from the surface of physical objects?
  - ...
- Questions about general methods of problem solving:
  - —How can one prevent an undesirable event from happening?
  - ...
- Questions about physical devices:
  - —What device produces heat?
  - ...
- Questions about mathematics:
  - —What is the solution of a given system of simultaneous equations?
  - —How can I minimize the value of a function?
  - ...

Table 2 The Functions of Agents.

1. Information service; the following are typical functions displayed as question-answer (Q-A) or request-answer (R-A) pairs:
   (a) (Providing facts)
   Q: What is the coefficient of linear expansion of copper at 20°C?
   A: 16.5×10^{-6}.
   (b) (Providing cases)
   R: Show me typical examples of flip-flop circuits.
   A: (typical flip-flops are shown)
   (c) (Providing generic facts)
   Q: How much heat energy is radiated from object surfaces?
   A: $E=kT^4$, where E: energy radiated, k: constant, T: the absolute temperature of the surface.
   (d) (Meta-level question)
   Q1: What can you do?
   Q2: What is the input range?
   Q3: How accurate is the answer?
2. Inference engine services such as:
   database management systems, theorem provers, language processors, belief management systems, expert system shells, learning algorithms, and so on.
3. Computation and problem solving service such as:
   numerical and symbolic computation, data analysis, diagnosis, scheduling, design, and so on; theoretically, this can be regarded as a product of an information source and an inference engine.
4. Knowledge community management and maintenance, involving:
   (a) Orientation: looking for an agent that has a requested function;
   (b) Interfacing between different agents;
   (c) Protocol translation: translating between local protocols;
   (d) Accommodating existing software into the knowledge community;
   (e) Human interface: interfacing the knowledge community with users.

The technical issues involved in this approach are as follows:

1. A computational framework for implementing the idea
2. A language and common protocol for enabling agents to interact.

3.2 The Knowledge Community—A Computational Framework

This subsection describes a computational framework, currently under development, for integrating heterogeneous knowledge. In *the knowledge community*, a computation element is called an *agent*. Each agent is a module that encapsulates closely connected, meaningful chunks of primitive processes and is small and simple enough to be implemented without much trouble; it provides facts, cases, generic information, computation and reasoning facilities, meta-level information, or information about other agents' abilities. The types of functions of agents are summarized in Table 2. Physically, agents are realized as processes of computational units interconnected by a computer network.

Agents communicate with each other by exchanging messages. For computational reasons, I have assumed

(1) trial

```
((from-agant-name asking-agent-1
  (from-domain here)
  (to-agent-name answering-agent-1
  (to-domain-name there
  (session-id session-1)
  (message-id asking-agent-1-message-1)
  (message-type request)
  (request-type information-retrieval-from-who-s-who)
  (return-value (y z))
  (return-mode all)
  (such-that (and (true ((?  x) class human))
                  (true ((?  x) name (?  y)))
                  (true ((?  x) specialty (?  z))))))
```

(2) response

```
((from-agent-name answering-agent-1
  (from-domain there)
  (to-agent-name asking-agent-1
  (to-domain-name here
  (session-id session-1)
  (message-id answering-agent-1-message-1)
  (message-type reply)
  (in-reply-to asking-agent-1-message-1)
  (requested-items (y z))
  (such-that
      (and (true ((?  x) class human))
           (true ((?  x) name (?  y)))
           (true ((?  x) specialty (?  z)))))
  (answer-type list-of-tuples)
  (answer (("Riichiro Mizoguchi" knowledge-based-systems)
           ("Riichiro Mizoguchi" intelligent-cai)
           ("Riichiro Mizoguchi" speech-recognition)
           ("Toyoaki Nishida" qualitative-physics)
           ("Hitoshi Ogawa" knowledge-based-systems)
           ("Katsuhito Yamazaki" case-based-reasoning)
           ("Katsuhito Yamazaki" computer-architecture))
           ...)
```

Fig. 9 Information Retrieval.

that each agent can only send messages to a finite number of agents at any one time.

Agents may be aggregated from various viewpoints. A physical aggregation of agents (a *domain*) is defined on the basis of physical location, while logical aggregation is defined from functional points of view, and dynamical aggregation results from the structure of message passing at execution time.

In order to accumulate and coordinate a large number of agents, we need a common language and protocol that allow agents to exchange information. In addition, we need to develop a toolkit for testing and evaluating the knowledge community, involving a language for defining agents and domains.

Durrently, the design and implementation of KC-0, the first prototype of the knowledge society, is in progress. As a common protocol, language, and terminology, KC-0 involves

- KCP (knowledge community protocol): a protocol for controlling the sequence of information exchange

(1) U requests P to open a new session;

```
(...
 (message-type request)
 (message-id user-message-1)
 (request-type open-session)
 (session-type prolog))
```

(2) P acknowledges and opens a new session;

```
(...
 (message-type reply)
 (result session-opened)
 (session-id prolog-session-1)
 (message-id system-message-1)
 (with-respect-to user-message-1))
```

(3) U gives P severals facts;

```
(...
 (message-type request)
 (request-type with-session-do)
 (session-id prolog-session-1)
 (message-id user-message-2)
 (messages ((assert ((human turing)))
            (assert ((human socrates)))
            (assert ((fallible (? x))
                     (human (? x)))))))
```

(4) P acknowledges;

```
(...
 (message-type reply)
 (session-id prolog-session-1)
 (message-id system-message-2)
 (with-respect-to user-message-2)
 (result all-actions-processed-successfully)
 (contents-of-reply (((human turing))
                     ((human socrates))
                     ((fallible (? x))
                      (human (? x))))))
```

(5) U asks P a question;

```
(...
 (message-type request)
 (request-type with-session-do)
 (session-id prolog-session-1)
 (message-id user-message-3)
 (messages ((fallible (? x)))))
```

(6) P responds with a solution;

```
(...
 (message-type reply)
 (session-id prolog-session-1)
 (message-id system-message-3)
 (with-respect-to user-message-3)
 (result all-actions-processed-successfully)
 (session-id prolog-session-1)
 (contents-of-reply ((found ((x . turing))))))
```

(7) U asks if there are any more solutions;

```
(...
 (message-type request)
 (request-type with-session-do)
 (session-id prolog-session-1)
 (message-id user-message-4)
 (messages (more)))
```

(8) P responds with another solution;

```
(...
 (message-type reply)
 (session-id prolog-session-1)
 (message-id system-message-4)
 (with-respect-to user-message-4)
 (result all-actions-processed-successfully)
 (contents-of-reply ((found ((x . socrates))))))
```

(9) U asks P to look for a further solution;

```
(...
 (message-type request)
 (request-type with-session-do)
 (session-id prolog-session-1)
 (message-id user-message-5)
 (messages (more)))
```

(10) P replies that there are no more solutions;

```
(...
 (message-type reply)
 (session-id prolog-session-1)
 (message-id system-message-5)
 (with-respect-to user-message-5)
 (result all-actions-processed-successfully)
 (contents-of-reply (no-more-answer)))
```

(11) U requests P to close the current session;

```
(...
 (message-type request)
 (request-type close-session)
 (session-id prolog-session-1)
 (message-id user-message-6))
```

(12) P acknowledges and closes the current session;

```
(...
 (message-type reply)
 (session-id prolog-session-1)
 (message-id system-message-6)
 (with-respect-to user-message-6)
 (result session-closed))
```

U stands for an agent ("user") asking for a Prolog interpreter service and P stands for another agent that works as a Prolog interpreter.

Fig. 10 Interaction with an Agent That Works as a Prolog Interpreter.

- KCRL (knowledge community representation language): an information representation language for representing objects and relations in the domain
- KCT (knowledge community terminology): a terminological system of representing concepts in the information representation language,

and as a language for defining experimentation environment

- KAPL (knowledge agent programming language): an experimental environment building tool.

3.2.1 KCP—A Protocol for the Knowledge Community

KCP provides a protocol for controlling the interaction between agents. The design criterion that we employed for KCP was to allow information to be exchanged at the knowledge level. Although efficiency is a secondary matter, we would like to avoid choices that might accidentally cause an increased computational load. Thus, KCP is much simpler than KIF [Genesereth and Fikes, 1990], Ontolingua [Gruber, 1991b], and the contract net protocol [Smith, 1980]. Each message is in the form:

```
((from-agent-name agent-name)
 (from-domain domain-name)
 (to-agent-name agent-name)
 (to-domain domain-name)
 (session-id session-id)
 (message-id message-id)
 (with-respect-to message-id)
 (message-type message-type)
 message-contents
 . . .)
```

where the name of the agent and its domain should be explicitly specified. I have assumed that some agents with an orientation facility may be useful for finding an appropriate agent and its domain.

The above does not imply that I have excluded contract net and KIF. On the contrary, it is permissible to introduce into the knowledge community one or more agent that simulates the function of the contract net protocol or KIF. It should be noted, however, that the effect of such a choice is limited to the local and not to the global context.

In normal situations, a meaningful unit of interaction consists of a sequence of messages called a *session*. Each session consists of pairs of trial and response messages. There are several types[1] of trial message such as: `open-session` (initiating a session), `close-session` (closing a session), `request` (requesting various kinds of information), `interrupt` (interrupting the execution of the current action), `resume` (resuming the interrupted action), `abort` (aborting the execution of the current action), and `status?` (asking for the current status of execution). In contrast, there is only one type of response message: `reply`. The reply is either success or failure.

In the KCP protocol, one needs to know the name of the agent before messages are sent. Furthermore, it is assumed that the names of domains should be globally unique and that the names of agents should be locally unique in each domain. This is a strong constraint, though it does not impose an extra burden on the message-switching subsystem. In order to relax the constraint, we incorporate into KCP the following features:

- There is a domain named `the-inter-domain-manager`, which is specifically used for orientation and domain name management.
- Domains are hierarchically organized from the general to the specific.
- Each domain contains a special agent, `the-domain-manager`, which manages the names of agents in the domain.

Figure 9 shows the way in which information is retrieved by using KCP, Figure 10 shows a more complex sequence, in which interaction is made with an agent that works as a Prolog interpreter.

In each case, it should be noted that KCP is concerned only with controlling the sequence of messages, not the contents.

3.2.2 KCRL—A Language for Representing Objects and Relations in the Domain

KCRL is a language for representing objects and relations in the domain, and is used to represent the contents of messages. Thus, KCRL is intended to be an *external* language for exchanging information, rather than an *internal* language for effective problem solving.

The role of an external language in the context of the knowledge community is to allow a wide variety of information to be transferred from one agent to another. Generally, there are two approaches to gaining expressibility: one is to enrich the syntax and semantics, while the other is to enrich the vocabulary and underlying ontology. I have taken the latter option for KCRL, so it has simple syntax and semantics and a rich and complex vocabulary. A KCRL expression is a collection of tuples

$$\{\ldots, \langle f_i, r_i, v_i\rangle, \ldots\}$$

that corresponds to the logical form:

$$\ldots \wedge r_i(f_i, v_i) \wedge \ldots$$

For example, a question (8) can be formally posed as:

```
{ ⟨r, class, request⟩,
  ⟨r, contents, c⟩,
  ⟨c, class, resolve⟩,
  ⟨c, object, s⟩,
  ⟨s, class, state⟩,                (9)
  ⟨s, true, e⟩,
  ⟨e, class, being-misty⟩,
  ⟨e, object, o⟩,
  ⟨o, class, glass⟩ }
```

[1] The type of a message is specified in the `message-type` field.

(1) Predicate logic

class(r, request)
∧contents(r, c)
∧class(c, resolve)
∧object(c, s)
∧class(s, state)
∧true(s, e)
∧class(e, being-misty)
∧object(e, o)
∧class(o, glass)

where, class, r, request, ... are constants.

(2)Frame

```
frame r:
      class : request
      contents : c

frame c:
      class : resolve
      object : s
...
```

(3) Semantic network

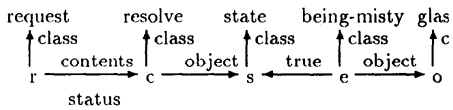


Fig. 11 Conventional Representation Equivalent to (9).

As this shows, KCRL expressions can be easily translated into conventional knowledge representations such as frames or semantic networks, and in fact they are much simpler than those. For example, see Fig. 11 for the equivalent conventional representation.

We regard conventional representations as macros for increasing readability, and do not use them as formal representations.[1]

3.2.3 KCT—A System of Terms for Describing the Domain

KCT defines a system of terms (a terminology) used in KCRL, allowing proper information transfer to be made between those agents that have posed questions and those that are capable of answering them. The central issue is the trade-off between expressive power and computability, which has been a topic of discussion in the research community with regard to terminological languages. The approach taken in the design of KCT was to define a very rich vocabulary with simple syntax and semantics and a rich ontology.[2] Each term is characterized as a feature complex consisting of attribute-value pairs. Although the expressive power of a feature complex is quite limited in terms of logic, I suspect that a feature complex would be useful, provided that it were equipped with a comprehensive terminology. The details of KCP are topics for future research.

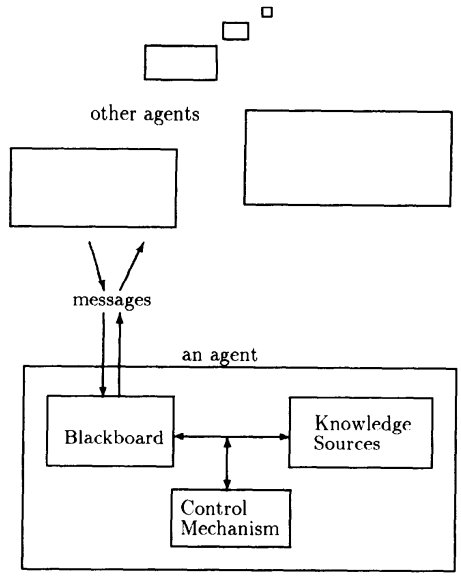


Fig. 12 KAPL is based on a blackboard architecture.

3.2.4 KPAL—A Programming Language for Defining an Experimentation Environment Based on the Knowledge Community

KAPL provides a standard programming environment for building experimental environments based on the concept of the knowledge community. It provides a means for defining agents and domains.[1] As shown in Fig. 12, a typical agent has as a common memory a blackboard that is repeatedly updated by knowledge sources.

Figure 13 shows how a domain, an agent, and a knowledge source are defined by using KAPL.

3.3 Sample Scenario Revisited

The example I gave in Section 3.1 may be handled by an agent such as SolvingCommonSensePhysics. KSs involved in SolvingCommonSensePhysics will decompose (9), a KCRL representation of (8), into simpler questions to external agents. For example, the KS SearchForCauseAndRemoveIt may decompose the question by using knowledge about causality, roughly saying that:

to terminate a state, identify the cause and prevent it from taking place (Fig. 14).

This causes a KS, SearchForCauseOf-

[1]It should be noted that we are thinking about inter-agent communication, and not human interface.

[2]Hobbs proposes *ontological promiscuity* and discusses semantic issues [Hobbs, 1985].

[1]The knowledge community might well be constructed by using a different method.

(1) Defining a domain

```
(define-domain name :manager name-of-manager
                    :agent-list list-of-agents ...)
```

(example):

```
(define-domain "psx@kuis.kyoto-u.ac.jp"
               :manager 'test
               :agent-list '(psx2pwl psx2nl psx3))
```

(2) Defining an agent

```
(define-agent name
              :ks-list list-of-KS's)
```

(example):

```
(define-agent psx2nl
              :ks-list '(search-for-a-fixed-point
                         determine-the-type-of-a-fixed-point
                         search-for-points-of-contact
                         construct-invariant-manifolds
                         divide-a-cell
                         ...
                         construct-flow-mappings))
```

(3) Defining a KS

```
(define-ks name :pattern pattern
                [:additional-condition predicate]
                :method method
```

(example):

```
(define-ks search-for-a-fixed-point
           :pattern
              (and (true ((? message-id) class message))
                   (true* ((? status)
                           (? message-id)
                           status
                           waiting-for-processing))
                   (true ((? message-id)
                          request-type
                          fixed-points-are-searched-for)))
           :method #'(lambda (ws-name a-list) ...))
```

Fig. 13 Definition of a Domain, an Agent, and a Knowledge Source in KAPL.

MistOnAPhysicalObject, to consult a physics knowledge base, in which, as a result, it may be found that

if the temperature of the glass is lower than that of surrounding air, mist on the glass may result.

This causes another update of the blackboard as in Fig. 15.

Eventually, the process may end up finding a solution that suggests attaching a heating coil to the glass. The process will become quantitative once a qualitative answer has been found.

In Fig. 16, I give the schema of horizontal coordination envisioned in the knowledge community project. A large portion of the vertical coordination will take place in a domain or an agent. For example, PSX2NL will be incorporated as an agent that interacts with a few other agents such as those doing numerical or symbolic computation. Currently, we are implementing the scenario on a apartial implementation of KC-0.

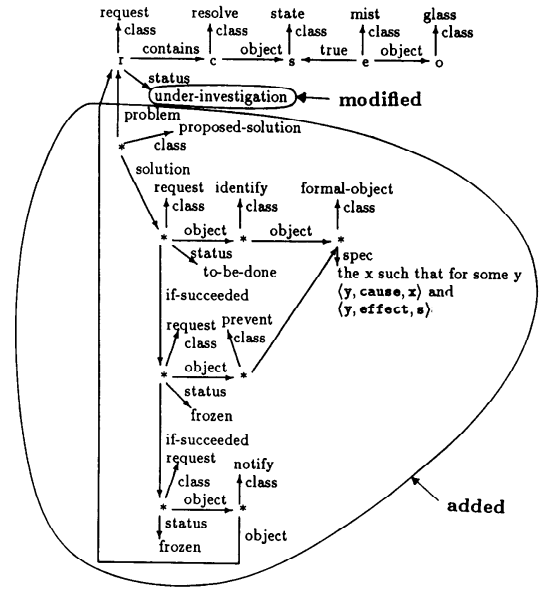


Fig. 14 Information Structure Grows as KSs Do Small Pieces of Work-1.
—after addition and modification by agent SearchForCauseAndRemoveIt.

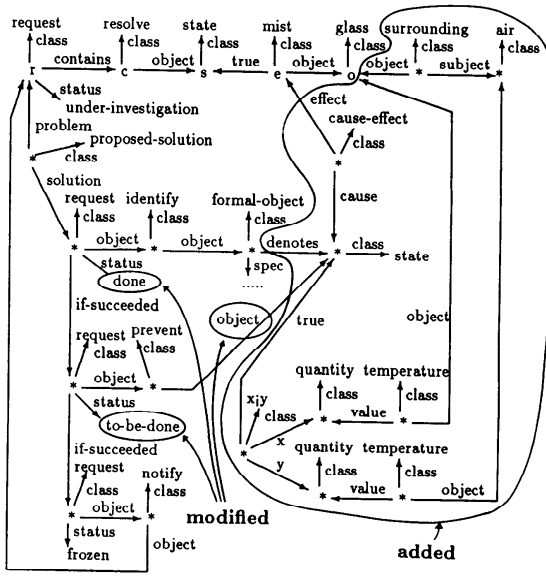


Fig. 15 Information Structure Grows as KSs Do Small Pieces of Work-2.
—after addition and modification by agent SearchForCauseOfMistOnAPhysicalObject.

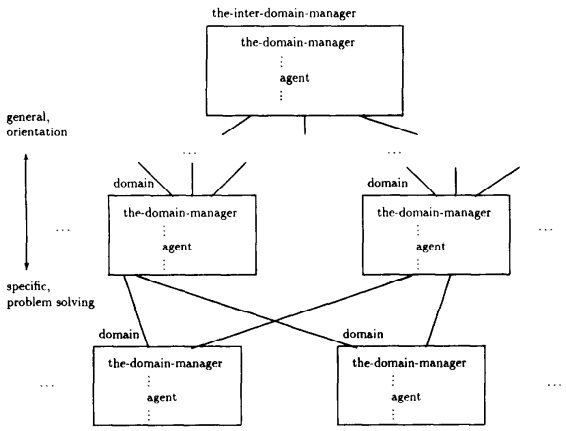


Fig. 16 Schema of Horizontal Coordination in the Knowledge Community.

3.4 Related Work

Efforts to build large-scale knowledge bases have become popular in recent years. An early example is the CYC project [Lenat and Guha, 1989], whose purpose is to build a very large knowledge base for the everyday common sense domain. Although we share the emphasis on large-scale accumulation of knowledge, our approach differs in several respects. Our effort is directed more to building a specialized, trans-domain knowledge base, ensuring executability, and providing a common external language and protocol. As a computational basis, we have employed cooperative distributed computation, rather than attempting to build one huge centralized knowledge base.

Our approach has much in common with recent activities in knowledge sharing and reuse [Neches *et al.*, 1991; Genesereth and Fikes, 1990; Gruber, 1991a; Gruber, 1991b]. The main difference is a rather technical one related to the underlying computational framework.

4. Concluding Remarks

Large-scale integration of knowledge is necessary for progress in Artificial Intelligence [Lenat and Feigenbaum, 1990]. In this paper, I have discussed issues related to the integration of heterogeneous knowledge, from the perspectives of vertical and horizontal coordination. My discussion of vertical coordination has been based on our experience with PSX2NL. I also presented the knowledge community as a step toward horizontal coordination. An interesting application of the knowledge community would be construction of an "executable" handbook for dynamical systems analysis. Many research goals remain for the future, including the (semi-)automated construction and maintenance of knowledge and terminology, incorporating in each agent an ability to reason about itself, other agents, and the knowledge community, and autonomous adaptation of protocols.[1]

References

[Genesereth and Fikes, 1990] GENESERETH, M. R. and FIKES, R. Knowledge interchange format version 2.2 reference manual. Technical Report Logic-90-4, Computer Science Department, Stanford University, 1990.

[Gruber, 1991a] GRUBER, T. R. Ontolingua: A language to support shared ontology (draft). Personal communication, 1991.

[Gruber, 1991b] GRUBER, T. R. The role of common ontology in achieving sharable, reusable knowledge bases. In J. A. Allen, R. Fikes, and E. Sandewell, editors, *Principles of Knowledge Representation and Reasoning—Proceedings of the Second International Conference*, Morgan Kaufmann (1991), 601–602.

[Guckenheimer and Holmes, 1983] GUCKENHEIMER, J. and HOLMES, P. *Nonlinear Oscillations, Dynamical Systems, and Bifurcations of Vector Fields*. Springer-Verlag, 1983.

[Hirsch and Smale, 1974] HIRSCH, M. W. and SMALE, S. *Differential Equations, Dynamical system and Linear Algebra*. Academic Press 1974.

[Hobbs, 1985] HOBBS, J. R. Ontological promiscuity. In *proceedings of 23rd Annual Meeting of the Association for Computational Linguistics* (1985), 61–69.

[Lenat and Feigenbaum, 1990] LENAT, D. B. and FEIGENBAUM, E. A. On the thresholds of knowledge. *Artificial Intelligence*, **47** (1990), 185–230.

[Lenat and Guha, 1989] LENAT, D. B. and GUHA, R. V. *Building Large Knowledge-based Systems*. Addison-Wesley, 1989.

[Minsky, 1985] MINSKY, M. *Society of Minds*. Simon & Schuster, Inc., 1985.

[Mizutani *et al.*, 1992] MIZUTANI, K., NISHIDA, T. and DOSHITA, S. Automated behavior analysis of piecewise linear ordinary differential equations in three-dimensional phase space. *Technical Report AI91-65, IEICE*, 1992.

[Neches *et al.*, 1991] NECHES, R., FIKES, R., FININ, T., GRUBER, T., PATIL, R., SENATOR, T. and SWARTOUT, W. R. Enabling technology for knowledge sharing. *AI Magazine*, **12**(3) (1991), 36–56.

[Nishida and Doshita, 1991] NISHIDA, T. and DOSHITA, S. A geometric approach to total envisioning. In *Proceedings IJCAI-91* (1991), 1150–1155.

[Nishida. *et al.*, 1991] NISHIDA, T., MIZUTANI, K., KUBOTA, A. and DOSHITA, S. Automated phase portrait analysis by integrating qualitative and quantitative analysis. In *Proceedings AAAI-91* (1991), 811–816.

[Smith, 1980] SMITH, R. G. The contract net protocol: High-level communication and control in a distributed problem solver. *IEEE Trans. on Computers*, **29**(12) (1980), 1104–1113.

[Zwillinger, 1989] ZWILLINGER, D. *Handbook of Differential Equations*. Academic Press, 1989.

(Received September 17, 1991)

[1] This was also recently suggested by Toru Ishida.