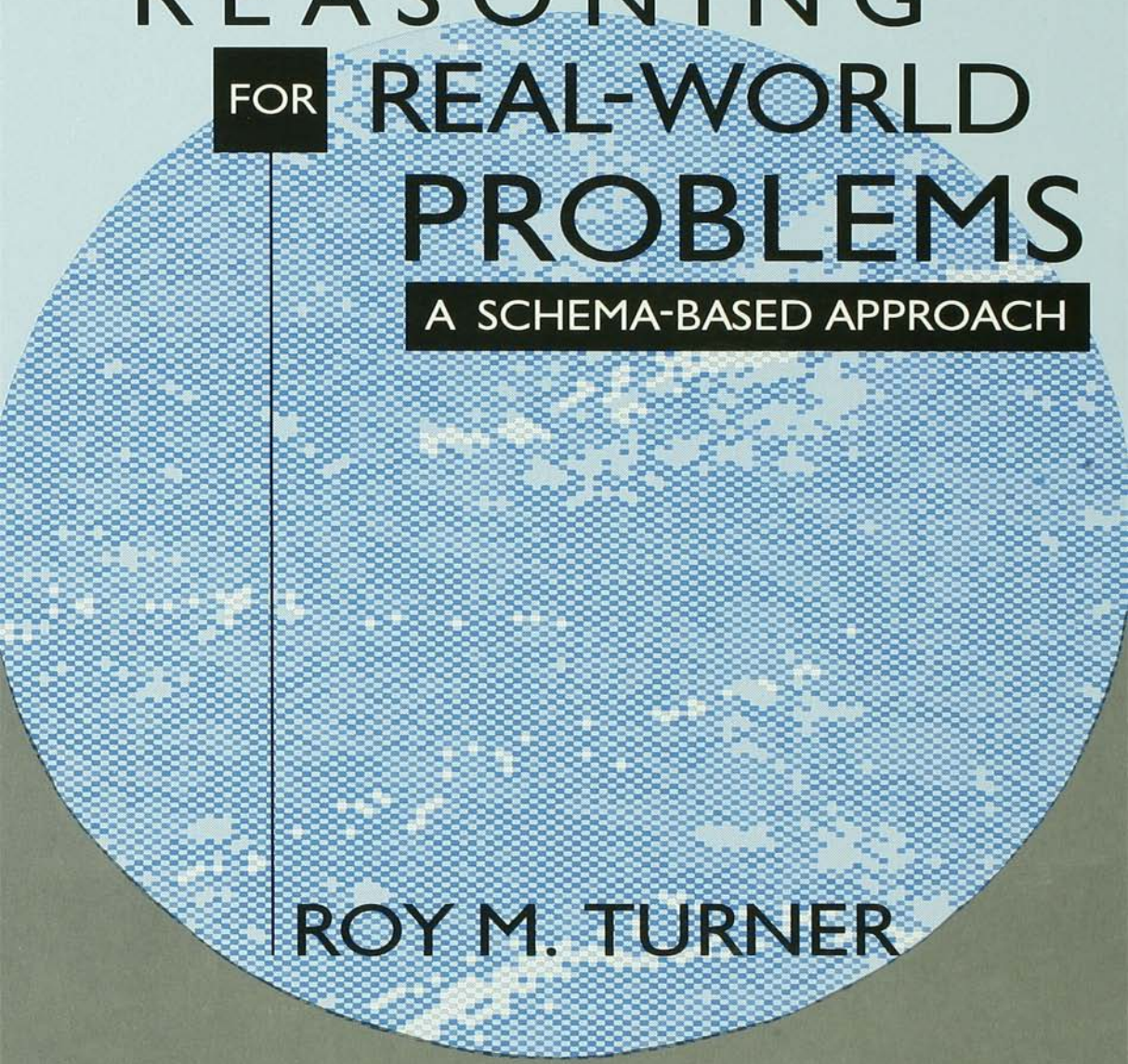
ADAPTIVE REASONING FOR REAL-WORLD PROBLEMS

A SCHEMA-BASED APPROACH



ROY M. TURNER

Adaptive Reasoning for Real-World Problems:

A Schema-Based Approach

Adaptive Reasoning for Real-World Problems:

A Schema-Based Approach

Roy M. Turner
University of New Hampshire


LAWRENCE ERLBAUM ASSOCIATES, PUBLISHERS
1994 Hillsdale, New Jersey Hove, UK

Lawrence Erlbaum Associates, Inc., Publishers
365 Broadway
Hillsdale, New Jersey 07642

Cover design by Kate Dusza

Library of Congress Cataloging-in-Publication Data

Turner, Roy M.
Adaptive reasoning for real-world problems : a schema-based approach / Roy M. Turner
p. cm.
Includes bibliographical references (p.) and index.
ISBN 0-8058-1298-9.
1. Artificial intelligence. 2. Reasoning. 3. Schemas (Psychology) 4. Adaptive control systems. I. Title.
Q335.T88 1994
006.3—dc20 93-39128
CIP

Books published by Lawrence Erlbaum Associates are printed on acid-free paper, and their bindings are chosen for strength and durability

Printed in the United States of America
10 9 8 7 6 5 4 3 2 1

To Elise

Contents

List of Figures

Preface

This book is an outgrowth of my dissertation work on adaptive problem solving for medical diagnostic reasoning, which was done at the Georgia Institute of Technology. As that work progressed, it became clear that schema-based reasoning is applicable to any problem-solving task in which an agent must cope with incomplete information, uncertainty, and unanticipated events. A particularly fruitful opportunity to expand the work presented itself at the University of New Hampshire, which until recently had one of the premier autonomous underwater vehicle (AUV) laboratories in the world. (The Marine Systems Engineering Laboratory, MSEL, has since moved to Northeastern University.) It seemed natural to move from the fairly unreactive medical diagnostic domain toward the challenges to be faced in the real-world task of controlling AUVs as they perform useful scientific missions, such as global change monitoring. Based on my experience so far, I believe that schema-based reasoning will prove useful for solving problems in this and most other complex real-world domains.

The book discusses schema-based reasoning and two implementations of it. MEDIC is a medical diagnostic consultant for pulmonology that was developed several years ago. Orca, a robust intelligent controller for ocean science AUVs, is currently being implemented by me and other members of the UNH Cooperative Distributed Problem Solving (CDPS) research group. It is hoped that Orca will see application on MSEL's AUVs within two or three years.

I am indebted to colleagues at UNH and Northeastern, particularly Elise Turner and the other members of the CDPS research group, as well as Dick Blidberg and the staff of MSEL, for their insights and discussions throughout this work. I am also indebted to colleagues at Georgia Tech, including my advisor, Janet Kolodner, for their helpful discussions during my dissertation work. I would also like to thank my editors at Lawrence Erlbaum Associates, Amy Pierce and Teresa Faella, for their help and patience.

The work reported here was generously funded by several agencies, to whom I am grateful. For funding work on MEDIC, I thank the Army Research Office (contract number DAAG29–83–G–0016), the National Science Foundation (grants IST–831771 and IST–8608362), and the Lockheed AI Center (grant DTD 09–25–87). For funding work on Orca, I thank the National Science Foundation (grant number BCS–9211914).

The writing process would have been painful without excellent software tools such as the Free Software Foundation's GNU Emacs editor, TGIF, dvips,

and Leslie Lamport's LaTeX document preparation system (and Donald Knuth's TeX), with which the camera-ready copy was prepared; I thank their authors for making such high quality tools freely available.

Finally, I would like to thank Elise as wife as well as valued colleague. In both roles, her support, advice, and understanding are much appreciated.

Roy M. Turner

Chapter 1

Introduction

The real world is not a nice place, at least for intelligent artificial agents. Consider an autonomous factory robot whose job is loading trucks. A foreman tells it "Put the box of widgets on the Acme truck, then meet me back here at 3 o'clock." The robot dutifully begins considering how to do this. Two things are immediately obvious. First, some partial commitment to future actions is needed, or else the sequence of the two events will not be guaranteed to be correct. Consequently, a plan should be formulated. Second, the robot probably has not been given enough information—there may be several boxes of widgets and more than one Acme truck at the loading dock. Consequently, the robot must ask the foreman for additional information.

The plan the robot formulates might be something like: go to the loading dock; pick up the box; go to the rear of the truck; put the box in the truck; return to the starting position by 3 o'clock. This is a sketchy plan, of course, but any further commitment to details would risk plan failure as the world changes during the plan's execution. Then, too, some details cannot be filled in yet; for example, the robot may not know the exact location of the Acme truck or of the box until it can sense them itself. Instead, it makes more sense to refine and expand the sketchy plan as necessary and possible.

While on the way to the box, the robot may notice that the path it has chosen is blocked by construction. It must change its plan slightly to select a path around the area. While following the new path, a person may step in front of it; the robot must pause to let the human past. This, however, is not really a change of its plan, but only a momentary interruption. Perhaps later, it notices a recharging station it did not know was on the path; provided it has time, it may choose to again interrupt its plan to take the opportunity to recharge its batteries. Further along, it may notice that the motor controlling one of its wheels appears to be malfunctioning slightly; without changing or interrupting its ongoing plan, it changes its motion parameters to compensate and makes a note to have the motor repaired later.

When it reaches the loading dock, the robot might notice that someone (or something) has stacked another box on top of the one it is to move. This would lead it either to change its plan by inserting a step to clear the top of the box or to suspend its plan while pursuing another goal to clear the box; either way, the effect is the same. When it tries to pick up the box, though, it might notice that it cannot—the box is heavier than it anticipated. This may lead it to look inside to make sure that the box really *does* contain widgets. It may also lead to a change in its plan (assuming that it is the right box) to push the box rather than lift it.

At the truck, the robot might notice another flaw in its evolving plan when it cannot lift the box to place it in the truck. This again leads to changing the plan, this time to insert a step to find something to use as a ramp. This done, it finishes its plan by pushing the box into the truck, placing the ramp back where it found it, and keeping its appointment with the foreman.

If in the future the robot is given a similar goal, its job will be easier. It can make use of what it has learned from solving the first problem to avoid many of the pitfalls it previously encountered. Perhaps, after solving many similar problems, it will create a general plan or schema for how such problems should be solved.

Problem solving in the real world is complicated by several factors, most of which are interrelated to some degree. First is the problem of incomplete knowledge and uncertainty. An agent will seldom have complete knowledge, or even all it needs, about the problem at hand, the world it operates in, or even itself. Part of this is unavoidable in almost all interesting domains, because humans will likely not have all the knowledge to give the agent. This is the case in domains such as controlling autonomous underwater vehicles (AUVs), diagnosing medical problems, maneuvering around factory floors, and so forth. Even when the knowledge might be available, using or storing all of it would likely tax the (usually extremely) limited rationality of the agent. Part of the problem also occurs because some knowledge may not be available when the agent would like to create its plan; for example, when told to "find the sunken ship and photograph it," an AUV will not know the location of the ship ahead of time—finding it is part of its task. Uncertainty arises in part from incomplete knowledge, since predictions or inferences made on the basis of incomplete knowledge will lack precision and certainty. Uncertainty may also arise due to inherent limitations of the agent's sensors (e.g., an AUV's sonar is limited with respect to resolution and noiselessness) or conflicting information it is given or receives from its sensors over time.

Second, unpredictable changes in the world will occur during the solution of the agent's problem, forcing it to respond. This is partly due to incomplete and uncertain knowledge, perhaps about processes and agents it knows exist.

For example, any autonomous vehicle is a real rather than ideal object; failures, imprecise responses of the "plant" (as roboticists often refer to the vehicle), and unpredictable interactions with the world in which the vehicle moves (e.g., currents for AUVs, rough sidewalks for autonomous land vehicles, etc.) will all lead to unpredicted consequences of the agent's actions. Some unpredicted changes to the world will also occur due to unknown processes (again arising from incomplete knowledge) and other agents present in the world. Other agents can affect the agent by their actions, by telling the agent information (which may be false or conflicting with its own knowledge), or by giving it new goals (e.g., when the agent is part of a cooperative distributed problem solving system).

A third problem is that deciding what to do is complicated by the context-dependent nature of responses to unexpected events, which goal to work on and which actions take to achieve it, and other behavior. When an AUV's forward motion across the bottom unexpectedly stops, it could be due to a current, entanglement in a net, thruster failure, or any of myriad other causes; the correct interpretation depends on the context (e.g., when moving upstream in a tidal river during ebb tide, the AUV may reasonably expect strong currents). When a fire alarm goes off, the response differs depending on whether you were sitting at your desk working or whether you were testing the alarm. When an ambulance driver has a goal to pick up his or her laundry, whether or not the goal is pursued depends on if he or she is driving home or taking a patient to the hospital. When needing to determine its position, an AUV's action will depend on whether it is in the open ocean (surface and obtain a global positioning system (GPS) fix), in a test range (query an acoustic transponder net), or operating under ice (rely on inertial guidance). When deciding what its depth envelope or other behavioral parameters should be, an AUV will also need to consider its context (e.g., it should tighten its depth envelope when in a harbor to avoid bottom clutter and surface traffic).

Unfortunately, many of the techniques which have stood artificial intelligence (AI) in good stead for most of its history fail miserably when the task is complex and must be carried out in the real world. A detailed plan, such as all early and even most current planning systems produce (e.g., Fikes & Nilsson, 1971; Newell & Simon, 1963; Wilkins, 1984), is essentially useless under these circumstances, since the plan's details will need to be changed over and over during problem solving. Flexibility and adaptiveness, not provably correct behavior, are the important characteristics of real-world agents.

What is required is *adaptive reasoning*: the ability of an agent to intelligently change its behavior, both short-term and long-term, in response to the changing needs of its problem-solving situation. I say "intelligently" thinking about some of the recent reactive planning research of the more extreme

variety (e.g., Agre & Chapman, 1987; Brooks, 1986). Although reactivity is important, an intelligent agent must retain the ability to create plans of some level of detail, or at least the ability to commit to some future actions. It must be able not only to react, but to react appropriately, where what is appropriate is always conditioned by the *context* in which the agent finds itself and possibly also by the *strategy* the agent is following.

This book describes one approach to adaptive reasoning called *schema-based reasoning* that has been implemented in the medical diagnostic consultant MEDIC (Turner, 1988, 1989a, 1989b, 1989c, 1992) and is being implemented in Orca, an intelligent controller for autonomous underwater vehicles (Turner & Stevenson, 1991). The rest of this chapter first describes some of the requirements for adaptive reasoning, gives a brief introduction to schema-based reasoning, and then provides an overview of the rest of the book.

1.1 Adaptive Reasoning Requirements

An adaptive reasoner needs to respond to changes occurring on two different time scales. *Short-term adaptation* involves responding to changes occurring during a single problem-solving session. For example, a person might walk in front of a robot, forcing it to pause or to modify its plan to go around the person. A doctor treating a patient for pneumonia might find out late in the diagnostic session that the patient has AIDS, causing him or her to change the diagnostic plan he or she was considering. *Long-term adaptation* involves adapting over a longer time scale as the domain knowledge and environment changes over time. For example, a person used to getting on to an interstate from an entrance on a particular street will have to change his or her schema (plan, script, etc.) if the ramp is permanently closed. In a domain such as medical diagnosis where the existing knowledge changes rapidly, a new test or procedure might be developed, forcing doctors to change the way they diagnose some of their patients.

So far, work on schema-based reasoning has concentrated almost solely on short-term adaptation, but with an eye toward facilitating later work on long-term adaptation; future work will focus on how to adapt a reasoner's behavioral repertoire over time to fit its domain and environment. Some of the most important requirements for short-term adaptation are described in the following paragraphs.

Paying attention to the current context. To a large extent, this is the most important requirement for an adaptive reasoner because it colors all of its other behavior. As discussed above, real-world agents must pay attention to their context in order to make reasonable inferences, appropriately select goals to work on, react intelligently to unanticipated events, select actions to

achieve goals, and modulate their behavior.

An adaptive reasoner must always have firmly in mind what its current context is. It is best if it explicitly recognizes the context, because this saves time. Once the context is recognized, then until it changes, information about the context can quickly and efficiently be used to control the agent's behavior. The alternative is to reason about the context each time the agent is deciding what to do. This would be analogous to a human thinking each time he or she decides to say something, "Am I in a library? Am I outdoors in the wind? Am I addressing an audience?" and so forth. Recognizing the context explicitly is analogous to a human realizing that he or she is in a library; actions usually taken in that context, such as speaking, can then be automatically modulated accordingly. This argues for a "context-centric" organization of an adaptive reasoner's knowledge.

Maintaining an appropriate focus of attention. At any time, an agent will likely have more goals to achieve than it has resources with which to achieve them, thus some of them must wait until others have been achieved. Consequently, an adaptive reasoner needs to know which of its goals it makes sense to focus its attention on at each point during problem solving.

This is complicated by changes in the agent's world and in its knowledge. The agent must be ready to change its focus of attention when its problem-solving context demands it.

Choosing appropriate actions: Avoiding detailed planning, but also avoiding purely reactive planning. Obviously, any reasoner needs to choose appropriate actions to take to achieve its goals. The actions must be as specific as possible not only to the goal but also to the current context. An adaptive reasoner must also walk the thin line between over- and undercommitment to its actions. It makes a great deal of sense to commit to some future actions rather than simply reacting, as argued above. However, the actions selected cannot be completely detailed, or the inevitable changes in the world will cause the agent to have to replan its actions.

Ability to interrupt or change ongoing actions. Actions taken to achieve goals must be interruptible if they take significant time to complete, since the world may change during their execution. If the "actions" are plans or plan-like entities, then they should perhaps be interruptible between steps. Interruptibility argues for explicit representation of the actions/plans as well, since not only must plans be interrupted, but they must also be resumed. Consequently, the internal state of their execution must be accessible to the reasoner so it can decide how (or whether) to resume them. Explicit representation is also argued for by the need to change a particular course of action based on changes to the world, for example, by adding, deleting, or changing plan steps.

Reacting appropriately to unanticipated events. Unanticipated events are the hallmark of real-world domains. By "unanticipated," I do not mean that the event is totally unexpected, but rather that its exact character or occurrence cannot be predicted. For example, an AUV controller may know that a leak is one of the possible malfunctions of its hardware, yet not expect in any real sense a particular occurrence of a leak.

Unanticipated events must be first recognized, that is, they must be diagnosed to the level necessary to handle them. "Motion stopped" may be an unanticipated event for an AUV, but simply noticing it does not go far enough; instead, the AUV controller must diagnose the cause of the event—that is, the real event. Once diagnosed, the event's importance must be assessed to decide if a response is warranted. If so, then an appropriate response should be selected, perhaps as a new goal to be achieved (e.g., "surface and signal for help"). Unanticipated events should sometimes cause the agent immediately to change what it is doing. This means that the agent must sometimes interrupt its current plan and refocus its attention on the new goal activated by the event. All phases of event-handling are dependent on the current context and should be guided by information the agent associates with its knowledge of the context.

Seizing opportunities and recovering from failures. Opportunistic reasoning, in the sense of seizing opportunities as they arise (cf. Hayes-Roth, 1985), is one kind of event-handling behavior. As with other unanticipated events, an opportunity must be recognized, assessed, and possibly responded to. Similarly, failures can be viewed as just another kind of unanticipated event. Only their consequences and the actions the agent must take in response differ.

1.2 Schema-Based Reasoning

Schema-based reasoning uses knowledge structures called *schemas* to capture all of an adaptive reasoner's knowledge. Schemas capture patterns existing in a domain or in an agent's procedural knowledge. They not only guide all facets of the agent's behavior, but also organize almost all of its knowledge. *Procedural schemas* are similar to hierarchical plans, though more general than usual. They are interpreted by the agent's reasoning mechanism to carry out actions to achieve goals. They are only expanded as much as necessary, can be specialized both before and during execution, and are interruptible and resumable. *Contextual schemas* represent contexts the agent may encounter, as well as store information to guide the agent in that context. In particular, a contextual schema holds information about responding to events, focusing attention, setting behavioral parameters, and selecting actions (procedural

schemas) to use to achieve goals. *Strategic schemas* hold information that defines the agent's problem-solving strategies for various kinds of problems. They can be thought of as a kind of domain-independent contextual schemas. Knowledge from strategic schemas is used to control the agent's reactivity and to help determine the focus of attention.

The schema-based reasoning mechanism will be described more thoroughly in the next chapter and the rest of the book. Essentially, though, a schema-based reasoner uses its schemas to:

- react appropriately to unanticipated events, including opportunities and failures;
- focus attention on appropriate goals to achieve;
- select appropriate actions to take to achieve goals;
- set parameters that modulate the character of the agent's behavior;
- allow commitment to some future actions while retaining the flexibility to change the details and to interrupt the plan;
- integrate the use of stereotypical ("canned") procedures, general procedures, and "from-scratch" problem-solving methods;
- use memories of previous cases to help solve new problems (i.e., case-based reasoning); and
- learn from experience by adapting problem-solving knowledge to meet the overall demands of its task and domain.

Schema-based reasoning is a computer model of adaptive reasoning. It is not a cognitive model, although many of the insights which led to the model and guided its development came from cognitive psychology. So far, work has concentrated on developing that portion of the model concerning short-term adaptation, though long-term adaptation has been kept in mind and has heavily impacted the way short-term adaptation is done.

1.3 Overview of the Book

The rest of the book presents schema-based reasoning (SBR) in much more detail. Chapter 2 gives an overview of the schema-based reasoning process. The next few chapters delve into aspects of the process in more detail: Chapter 3 looks at the knowledge representation needed to support SBR; Chapter 4 describes the attention-focusing process, which relies heavily on information from contextual and strategic knowledge; Chapter 5 discusses how procedural schemas are used to achieve goals; Chapter 6 looks at how contextual and strategic knowledge is used to guide the agent as it responds to unanticipated events; and Chapter 7 discusses a memory mechanism to support SBR. Chapter 8 describes two implementations of SBR, MEDIC and Orca. Chapter 9 presents what is hopefully an objective evaluation of SBR and includes

a discussion of how our research fits in with related work in the AI literature. Chapter 10 presents our conclusions and directions for future work.

Chapter 2

Schema-Based Reasoning

The basic idea of schema-based reasoning (SBR) is very simple: represent most or all of an agent's problem-solving knowledge explicitly as declarative knowledge structures called *schemas*, then use those schemas to guide all facets of the agent's behavior. The motivation is to allow a reasoning system to capitalize on patterns existing in its world to quickly, automatically, and appropriately tailor its behavior to its problem-solving situation.

Patterns exist both in the world around the agent and in problem solving itself. An autonomous underwater vehicle, for example, will likely operate in many similar environments: similar harbors, coastlines, and so on. Likewise, it will again and again find itself in similar circumstances: on similar search or exploration missions, operating when power is low, when there is time pressure, and so on. Patterns in the agent's own problem solving develop over time as the agent performs similar actions to accomplish similar goals; such patterns have long been at least implicitly recognized in AI and represented as plans, scripts, and rules.

We use schemas to represent patterns appearing (or potentially appearing) in the world as well as those that have been or that are projected to be useful during problem solving. Currently, and realistically for the near future, schemas are given to the agent by humans based on their expertise or experience. Ultimately, however, schemas will be created and modified by the agent using them based on its own experience and evolving expertise. Schema-based reasoning is, then, a kind of generalized case-based reasoning (e.g., Hammond, 1989a; Kolodner, 1987; Kolodner et al., 1985) where generalized knowledge contained in schemas mostly supplants the records of single cases of past problem solving used in more "mainstream" CBR.

Schema-based reasoning combines explicitly represented schemas with a flexible reasoning mechanism to produce an adaptive, context-sensitive approach to real-world problem solving. This chapter gives an overview of the schema-based reasoning process. In the following chapters, aspects of the process are presented in more detail.

2.1 Schemas

Schema-based reasoning uses three kinds of schemas, corresponding to the three types of knowledge necessary for adaptive reasoning. A *procedural schema*, or *p-schema*, is used for taking action, usually to achieve a goal. P-schemas are in many ways similar to hierarchical plans or scripts (Cullingford, 1981; Schank & Abelson, 1977) they are more general than either in some respects. A p-schema specifies steps to take, each of which can be either a primitive action (i.e., one the agent can directly carry out), a subgoal to be achieved, or another p-schema. In the latter case, the p-schema (subschema) can be specialized at run-time if needed to one that is more specific for the current problem-solving situation. A *contextual schema*, or *c-schema*, represents a problem-solving context or a portion of a context; c-schemas can be viewed as representing generalized problem-solving episodes. They are used to modulate an agent's behavior appropriately for its current situation, including its event-handling and attention-focusing behavior. A *strategic schema*, or *s-schema*, represents a problem-solving strategy; whereas a c-schema represents the domain-dependent aspects of a situation, an s-schema represents the domain-independent aspects having to do with strategic behavior.

Figure 2.1 shows an example of each type of schema. The contextual schema, *c-harbor*, is from the domain of Orca and represents the context of being in a harbor. Note that in addition to describing the context, *c-harbor* also provides information about how the agent should handle unanticipated events, focus its attention, and select ways to achieve goals. Both the procedural schema, *p-consult*, and the strategic schema, *s-HDreas*, are from MEDIC. *P-consult* is a very high-level p-schema that guides MEDIC during a consultation. *S-HDreas* represents the hypothetico-deductive reasoning strategy used by expert clinicians.

2.2 The Schema-Based Reasoning Process

Schema-based reasoning uses at its core a process that interprets, or *applies*, p-schemas to take action, but that remains ready to respond to the needs of the changing problem-solving situation. Added to this is functionality to take into account the agent's current context and strategy when handling unanticipated events and focusing its attention so that its behavior is automatically appropriate for its problem-solving situation at all times. Figure 2.2 shows the SBR process.

Information from the current contextual and strategic schemas is used to focus the reasoner's attention on one of its active goals, which is then pursued. A procedural schema is selected, using any information present in the

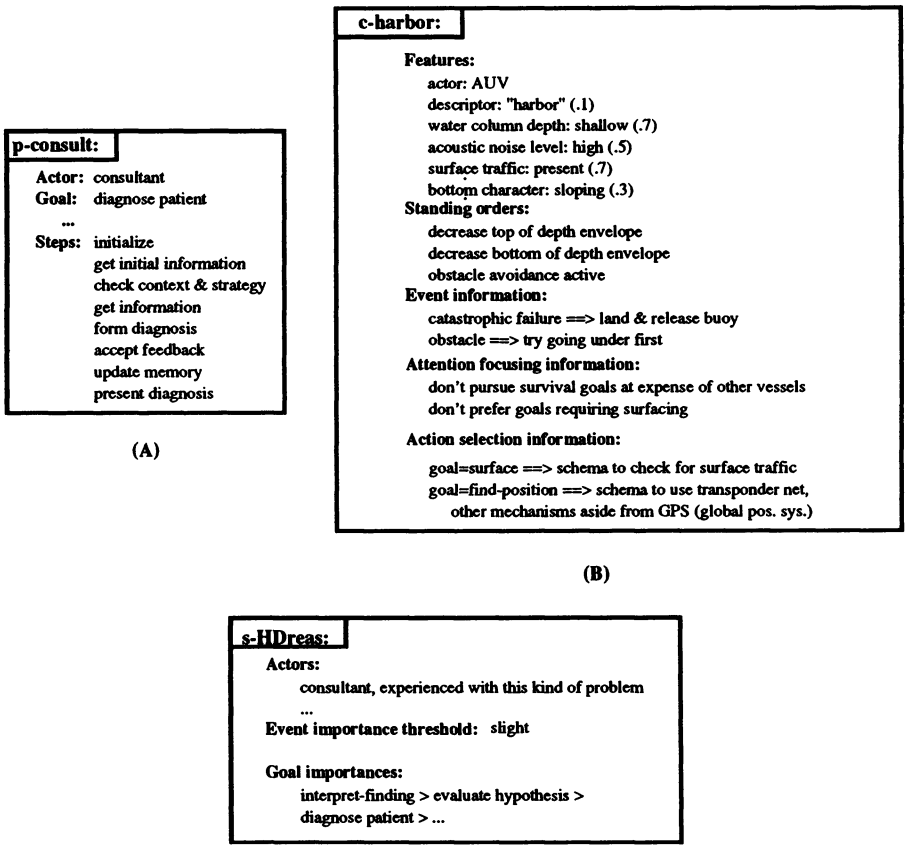


Figure 2.1: Examples of schemas: (A) a procedural schema; (B) a contextual schema; and (C) a strategic schema.

contextual schema about appropriate ways to achieve the goal in the current situation. The procedural schema is then applied, achieving the goal. The process repeats until there are no more unsatisfied goals.

A schema-based medical consultant, for example, would proceed as follows. After the client presents a sketch of the problem ("diagnose a patient"), the consultant finds in its memory *c-consult*, a contextual schema representing the context of "a consultation"; this c-schema is relevant because it is in such contexts that one encounters the goal of diagnosing a patient's problem. Since there are no abstract features (yet) in this problem-solving session to cause the consultant to favor one strategy over another, a default strategy, perhaps suggested by *c-consult*, would be used. For an experienced consultant, this would likely be *s-HDreas* (Figure 2.1c), hypothetico-deductive reasoning. There is

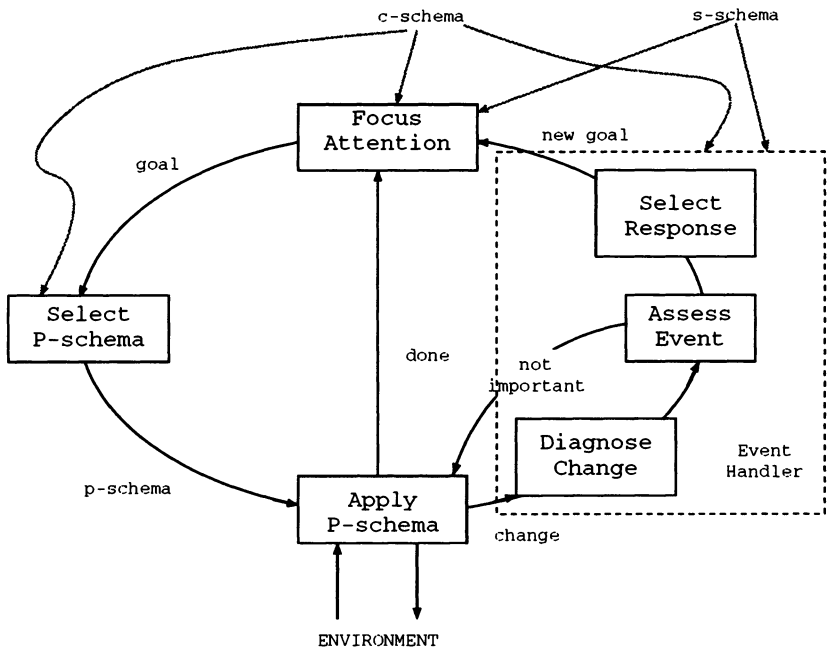


Figure 2.2: The schema-based reasoning process.

only one goal ("diagnose patient"), so that is selected. The consultant then finds an appropriate p-schema using information from its current c-schema along with its schema memory. In this case, it would likely find *p-consult* (Figure 2.1a), a high-level p-schema that encodes knowledge of how to do consultations. This p-schema would then be applied by interpreting its steps to take actions such as asking questions, making hypotheses, and presenting the diagnosis.

Handling Changes

Since a schema-based reasoner is designed to operate in the real world, very rarely if ever will it be able to apply a p-schema from start to finish without some change occurring in the world that needs to be addressed. If nothing else, the process of applying the schema will itself change the world sufficiently to impact its further application. When a change occurs, the agent may need to interrupt the application of the current p-schema while it decides what to do. If the change is unimportant, it can resume application of the interrupted p-schema. If not, it may decide to respond to the change, possibly by activating a new goal. In this case, or if the situation has otherwise changed sufficiently due to the change, it must refocus its attention. The goal selected to pursue as

a result may be the same one, in which case the application of its associated p-schema can be resumed (possibly after taking some additional actions to compensate for the changed situation). Alternatively, the changed situation may dictate selecting a different goal, either the newly activated one or one of the others the agent needs to achieve. In this case, work on the original goal is delayed until the situation changes to again warrant it.

As with everything else, when a change occurs, a schema-based reasoner uses its schemas to decide what to do. Included in a c-schema is information about events (changes) that may occur in the context represented. For example, *c-harbor* (Figure 2.1b) describes the event of encountering an obstacle. Some of this information can be used by the agent to diagnose the cause of the event. For example, "motion stopped" is an unanticipated event, but one that is symptomatic of other events; the agent must determine which of these caused motion to be stopped in order to do the right thing in response.

Other information stored with the event description in the current c-schema helps the agent decide whether or not the event is important and, if so, how to respond. By storing the information here, the agent can automatically make context-specific decisions about events, since the information is already available for use without extensive search. For example, in the context of exploring an area, the event of detecting an object is important (the agent may wish to photograph the object or avoid it). In the context of docking with its support vehicle, however, detecting an object is not as important (the object may be the support vehicle, which obviously should not be avoided). The current s-schema also participates in deciding whether or not to respond to an event. An s-schema has information about how reactive the agent should be when using a particular strategy; for example, *s-HDreas* (Figure 2.1c) suggests that the reasoner be very reactive by specifying a low "event importance threshold."

Once the agent has decided that an event is important in the current context and under the current strategy, it uses other information in its current c-schema to help it decide how to respond. This information is in the form of links between goals expected to arise in the context and actions that are appropriate for them. For example, in the context of an AUV being in a harbor (represented by *c-harbor*), an appropriate response for a catastrophic failure is to land and release a buoy; surfacing here would risk getting run over by surface traffic. In a different context, for example, operating in the open ocean, surfacing may be the preferred response; in still other contexts, such as operating under ice, still other responses would be needed. By using the current c-schema to help it decide what to do, the agent can very quickly make the right decision. This can be important in real-world problem solvers, since delay can jeopardize the mission and possibly even the survival of the agent.

Conditioning Behavior to the Context

Decisions take two forms: large ones having to do with such things as deciding how to achieve goals and respond to unanticipated events, and small ones having to do with modulating behavior. It is important that both kinds of decisions be made as quickly and as appropriately as possible. A schema-based reasoner makes use of its knowledge of the current context and its current strategy to *automatically* make the appropriate decisions.

We can think of contextual and strategic knowledge as conditioning the reasoner's behavior for its current situation. One way the current c-schema does this is by providing parameter settings, called "standing orders," that affect aspects of the agent's behavior. For example, the standing orders in the context of being in a harbor (*c-harbor*, Figure 2.1b) specify tightening the AUV's depth envelope to prevent hitting bottom clutter or being hit by surface traffic. No deep reasoning needs to be done; instead, when the AUV enters the context, it automatically tightens its depth envelope. The other way that the current c-schema and s-schema condition behavior is by providing information about how to focus attention, handle unanticipated events, and select actions to use to achieve goals.

Along with the process described previously, a schema-based reasoner must also ensure that its current view of its context and its current strategy are appropriate for its problem-solving situation. The reasoner does this by constantly examining the current situation and using information about it to "probe" its schema memory to find the best c-schema(s) and s-schema—that is, the ones that most closely match the current situation.

Generally, a reasoner will have only one strategy in use at a time. However, it may find more than one c-schema that matches the current situation. This is intentional. If the reasoner were to try to represent every possible context in which it might find itself, the number of c-schemas needed would be too great to store or retrieve. Instead, our approach is to represent views of contexts as c-schemas, then allow these views to be combined into coherent pictures of the reasoner's current context at run-time. (This is how Orca will work; MEDIC uses a simpler scheme that will be described later.) For example, when an AUV finds itself operating with low power on a search mission under ice, it would retrieve from its memory c-schemas representing "low power," "search mission," and "operating under ice." These would then be merged into its current "c-schema," which is really a composite structure composed of several component c-schemas. As the situation changes, some of these component c-schemas may no longer be apropos (e.g., the AUV may emerge from beneath the ice), while new ones will be found (e.g., "operating in the open ocean"); when this happens, the reasoner's view of the current situation will change as its *context manager* updates the current c-schema appropriately.

Schema Memory

A schema-based reasoner must store its schemas in such a way that they can easily be retrieved based on features of the situation. To facilitate long-term adaptation, the schema repertoire must be capable of change, in terms of both adding and changing schemas as well as changing the way they are organized. Any schema memory that meets these requirements can be used for a schema-based reasoner. The memory we use is a dynamic conceptual memory (Schank, 1982) patterned after the CYRUS program (Kolodner, 1984). This kind of memory will be discussed in more detail in Chapter 7.

2.3 Implementations of SBR

Throughout this book, we will use examples drawn from our two primary implementations of schema-based reasoning, MEDIC and Orca. MEDIC is a pulmonary consultant that was constructed at the Georgia Institute of Technology. Orca is an AUV controller that is currently being designed and built at the University of New Hampshire; Orca will ultimately be fielded on the Northeastern University Marine Systems Engineering Laboratory's EAVE AUVs. The two programs are different in many respects, caused both by the differing needs of their domains as well as by the fact that work on Orca is benefiting from the lessons learned from MEDIC.

Both programs are more thoroughly discussed in Chapter 8. Here, we give an example session with MEDIC to give the reader the flavor of what schema-based reasoning is like. A complete session with MEDIC is given in Appendix A.

MEDIC's behavior is constantly under the control of schemas. If no problem is being solved, a procedural schema called *p-resting* is in control. Its purpose is to gather enough information about a problem to allow MEDIC to begin solving the problem. We see this p-schema being applied below:

```
[1]:  Hi, I'm Medic.
[2]:  What can I do for you?
[3]:  : (DIAGNOSE-PATIENT)
[4]:        Looking for a c-schema containing '(^DIAGNOSE-PATIENT)'...
[5]:        ...found ^C-CONSULT.
[6]:        Selecting strategy...
[7]:        ...setting strategy from current c-schema to be ^S-HDREAS.
```

Once a problem is presented (in the example by the client telling MEDIC to diagnose a patient),[1] MEDIC puts the particulars of the problem in its short-term memory (STM). It then tries to find a contextual schema representing

[1]MEDIC has no natural language capabilities; the input to MEDIC is in a form close to its internal representation scheme.