Context-Sensitive, Adaptive Reasoning for Intelligent AUV Control: Orca Project Update*

Roy M. Turner
Marine Systems Engineering Laboratory
Marine Science Center
Northeastern University
East Point, Nahant, MA 01908
Phone: 603-862-2980 FAX: 603-862-3493
E-mail: rmt@unh.edu
WWW: http://cdps.cs.unh.edu

Abstract

Intelligent mission-level control of autonomous underwater vehicles demands an *adaptive reasoner*: a program that can create plans for accomplishing mission goals, but that does not overcommit to future details, that remains ready to interrupt what it is doing as the situation evolves, and whose behavior is always appropriate for the context in which the AUV finds itself. The goal of the Orca project is to create such a reasoner. Our approach is based on *schema-based reasoning*, an adaptive reasoning method that uses procedural and contextual schemas to represent all problem-solving knowledge. This paper describes the Orca program and discusses the project's current status and our future plans.

Intelligent control of autonomous underwater vehicles is a demanding task for which no completely satisfactory approaches have yet been developed. For AUVs that are designed to remain on-station for significant periods of time (e.g., participants in a long-duration autonomous oceanographic sampling network [Curtin *et al.*, 1993]), or those whose missions are otherwise of long duration (e.g., long-range AUVs), the task is even more difficult. Not only must the controller create and follow a plan to accomplish mission objectives, it must also pay careful attention to resource management. Unanticipated events, which can occur during missions of any duration, become much more likely the longer the mission; if the AUV will generally be far from a base, and hence from human intervention, handling events intelligently becomes more critical. More so than for short-range AUV control, the mission controller must be context-sensitive, ensuring that its specific actions and overall behavior are appropriate for the situations in which it finds itself. This includes not only actions that further the achievement of mission goals, but also the way it diagnoses and responds to unanticipated events and focuses its attention on what to do. It also includes such "background" aspects of behavior as automatic goal activation/deactivation by context and the setting of behavioral parameters, such as depth envelopes and so forth—these, too, are context-dependent.

The Orca project has as its goal the creation of a robust, intelligent controller for long-range and long-duration ocean science AUVs. The project is currently halfway through its initial phase, which has as its target creating a proof-of-concept version of such a controller and testing it in simulation experiments. The next phase will put the program aboard the Marine Systems Engineering Laboratory's EAVE AUVs (and MSEL's long-range AUV, if that is ready at that time) for in-water tests. The end result of the project will be a context-sensitive, adaptive reasoner for AUV control named Orca. The design and construction of the Orca program is an iterative process. In this paper, we will describe the current version of Orca and discuss plans for future versions that will be built during the initial phase of the project. We also discuss other plans for using Orca, in particular as a controller for the EAVE vehicles as they participate in multiagent cooperative distributed problem solving.

*The author is grateful to the National Science Foundation for grant BCS–9211914, which supported this work in part. The author is also affiliated with the Department of Computer Science, Kingsbury Hall, University of New Hampshire.

Requirements for Intelligent AUV Control

An intelligent mission controller for an AUV must be able to accept missions from users and ensure that those missions are carried out to the best of the AUV's capabilities. It must also care for the safety of the AUV (when that is a priority), which includes being able to abort the mission and take recovery actions as necessary.

These attributes give rise to several requirements. First, an intelligent AUV controller must be able to create and execute plans of action. There has been a good deal of recent interest in the more extreme form of reactive planning—that is, no planning at all—for robot control [e.g., Brooks, 1986]. Though this approach has had much success for robots accomplishing simple tasks, many (if not most) useful tasks for AUVs require the ability to commit to future actions, which in turn requires planning. Examples include taking a data sample at a particular time, coordinating activities with another agent to achieve a shared goal, constructing an underwater structure, and rendezvousing with a user or another AUV.

However, the lessons of reactive planning were well learned: an AUV controller cannot simply create a detailed plan, then go execute it. It may not be possible at all, due to incomplete knowledge, and even when it is possible, uncertainty and changes in the world will almost certainly cause the plan to be suboptimal or even to fail. Thus, a second requirement is that the mission controller be able to delay commitment to details of plans as long as possible. Practically, this means interleaving planning and plan execution.

Another requirement is that the mission controller be able to interrupt what it is doing when the situation changes. This may happen either because of changes in the world (e.g., new features are observed or new goals arise) or because the agent's view of the world changes. In either case, the agent must be able to interrupt its current plan, perhaps later to return to it, to handle the change.

A mission controller must also be context-sensitive. That is, its behavior should always be appropriate for the context it is in. Most artificial intelligence approaches to problem solving seek to do this, but for most of them the cost is high, either in terms of reasoning effort or space (e.g., increased memory needed to store redundantly-represented application conditions of rules or plans). Recently, context has become a research area in its own right [e.g., Brezillon, 1993; Brezillon, 1995]. The Orca project's contribution to this area is in proposing one way that context-sensitivity can be made automatic and relatively effortless for the reasoner.

A crucial requirement for a mission controller is robustness. The AUV is a real, not an ideal, machine, with all that implies for failures and imprecise response. The controller must be able to deal with failures and recover from imprecision; it must be able to handle uncertainty. And it must be able to interrupt what it is doing as the world (or its view of the world) changes.

An aspect of robustness that is often overlooked is the "brittleness" problem that plagues expert systems. When an mission controller has special-purpose, highly specific knowledge about how to behave (e.g., how to achieve a goal) in a particular situation, it should use it. But when it does not, it should not fail; instead, it should be able to bring to bear more general-purpose knowledge so that *some* solution to the problem is obtained, even if it is not necessarily the best solution.

A final requirement is efficiency. The mission controller need not be a hard real-time system, since it will be insulated from the vehicle hardware by one or more layers of low-level software in almost all cases. However, it should still be able to create a plan and to respond in a timely fashion to changes in the world. One way to help insure this is if the program can reuse previously-created plans, either from its own experience or from a human, to achieve new goals.

We call a program that can meet these requirements an *adaptive reasoner* [Turner, 1994], since it can adapt its problem-solving behavior to the needs of the evolving problem-solving situation.

Orca

Initial ideas for the Orca project began several years ago [e.g., Turner & Stevenson, 1991], but the project did not really begin until a year and a half ago. Orca was conceived as an intelligent mission-level controller to fill the need for such a program in the top-most layer of the MSEL/EAVE software architecture [Blidberg & Chappell, 1986; Blidberg *et al.*, 1991]. This layer is responsible for taking a high-level description of

mission goals from a user (e.g., an ocean scientist) and formulating and executing plans to accomplish the mission. By concentrating mission planning expertise onboard the AUV, the need is eliminated for forcing the user to become an "AUV programmer". In addition, the mission controller is responsible for ensuring the survival and return of the AUV, even in the presence of unforeseen circumstances.

Orca grew out of an earlier project of the author's, MEDIC [e.g., Turner, 1994]. MEDIC was a medical diagnostic program designed to interact with a human. Its reasoning style, *schema-based reasoning*, grew out of case-based reasoning and reactive planning [e.g., McDermott, 1978; Georgeff & Lansky, 1987]; as work on MEDIC progressed, it became obvious that the approach was at least as useful for real-world domains such as robot control.

Orca is an adaptive reasoner that uses schema-based reasoning. The basic idea behind schema-based reasoning (SBR) is simple: represent all of an agent's (e.g., an AUV controller's) problem-solving knowledge explicitly as packets of related knowledge called *schemas*, then retrieve the appropriate schemas based on features of the current problem-solving situation and use the knowledge in them to guide the agent's behavior in that situation.

Two kinds of schemas are used in Orca, procedural and contextual.[1] *Procedural schemas* (*p-schemas*) are similar to plans, scripts, or rules; they suggest actions that Orca should take to achieve goals. *Contextual schemas* (*c-schemas*) represent kinds of problem-solving situations Orca might encounter; they are used to ensure that all facets of the program's behavior are context-appropriate.

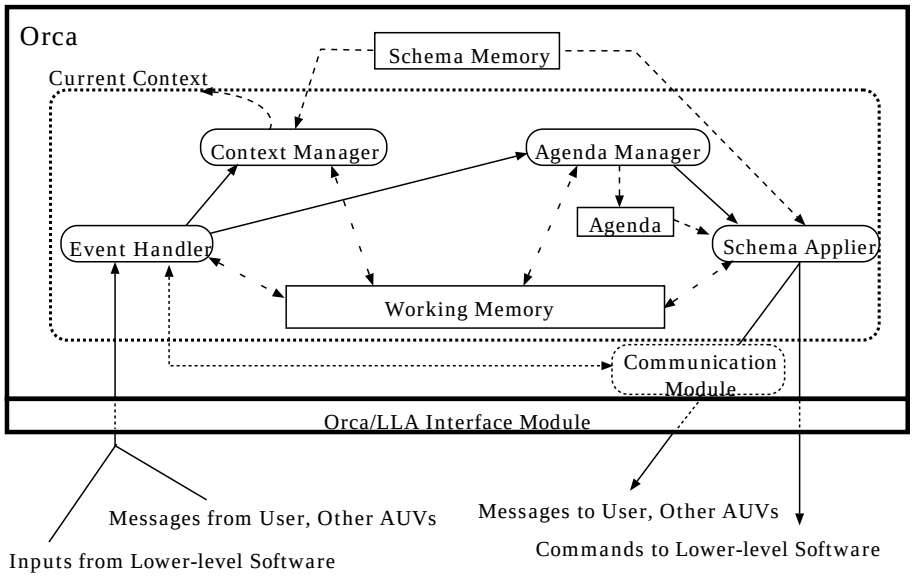

Figure 1: Internal structure of Orca.

The internal structure of the current version of Orca is shown in Figure 1. Agenda Manager (AM) is responsible for maintaining the current focus of attention, that is, for identifying the goal or goals that Orca is currently working on. Schema Applier (SA) is responsible for finding and interpreting, or *applying*, p-schemas to achieve the goal(s) in focus. Based on knowledge in the p-schema it is using, it takes actions such as making inferences, sending messages to the user or other agents (via the communications module, the subject of a related project [E. Turner *et al.*, in press]), and sending commands to the lower-level software (i.e., the Navigator) on board the AUV. Event Handler (EH) is responsible for managing all input to Orca. It is responsible for situation assessment as well as detecting and handling both anticipated and unanticipated events. Context Manager (CM) maintains Orca's view of what its current context is and sends information to the other modules to ensure that their behavior is appropriate for that context. It builds a knowledge structure, called the *current c-schema*, that forms a backdrop for all the other modules' behavior. Working

[1] The role of MEDIC's strategic schemas is being combined into Orca's contextual schemas.

Memory (WM) holds Orca's knowledge about the situation and the current state of problem solving, the agenda holds the active goals, and the Long-Term Memory (LTM), or schema memory, holds Orca's schemas.

Attention Focusing

Attention focusing is necessary because Orca, like any other agent, will likely have more goals active at any given time than it has attentional and other resources available with which to achieve them. Consequently, a subset of the current goals needs to be identified on which it is appropriate to work.

Attention focusing in Orca is performed by the Agenda Manager. AM does this by assessing the relative priority of the goals on Orca's agenda, a knowledge structure that holds all active goals. The goal or set of goals with the best priority is selected for Schema Applier to work on.

We are building a fuzzy rule-based system to do much of the work of attention focusing. This mechanism was selected to cope with the incomplete and uncertain knowledge present in the AUV domain. The RBS, which will be run in a backwards direction for goal priority assessment, will get its rules not only from a static (within a given mission) rule base, but also will receive rules from the current c-schema via Context Manager. This will tailor the RBS', and hence, AM's, behavior to fit the current context, thus ensuring that the goals in focus are appropriate ones for the situation. We anticipate that the RBS will be complete by the time of publication.

Procedural Schemas and Schema Application

Orca takes actions based on procedural schemas in a process called *schema application*. Each p-schema contains a description of the situation for which it is appropriate and knowledge of how to achieve a goal in that situation. The latter knowledge is comprised by the p-schema's steps and its ordering information.

A step in a p-schema can be one of: (1) a p-schema (i.e., a subschema), in which case it is applied recursively; (2) a goal, in which case a p-schema is found for it and applied; or (3) an executable action, or *xact*, that Orca directly carries out, perhaps by sending a command to the low-level software or by sending a message to the user or another AUV.

Steps are ordered by *ordering constraints* present in the p-schema. Our schema representation language is rich enough to allow sequential, parallel, or nondeterministic execution of steps, as well as to allow if–then–else structures, looping constructs of various sorts, explicit failure-handling instructions, and explicit requests to wait for some condition to be met. In the future, additional constraints having to do with resource use and constraints on the kinds of actions that can execute together will be added to the step ordering repertoire.

The Schema Applier (SA) is responsible for finding and applying p-schemas to achieve goals. The schema application process is shown in Figure 2. When a new goal arrives, it is placed on the agenda, and at some point, Agenda Manager will focus attention on it. Schema Applier first finds an appropriate p-schema for the goal in the current situation. It does this by requesting that Long-Term Memory (LTM) search for such a schema based on features of the current problem-solving situation. Once found, the schema is instantiated (i.e., to bind its variables) as part of an "action record" which holds a record of its application, and it is then applied.

SA applies a p-schema by expanding it into its steps. The first step is identified and examined. If it can be directly executed, i.e., it is an xact, then SA takes the appropriate action. If it is instead a p-schema, then SA asks LTM to see if there is a more specialized version of it for the particular situation, and then recursively applies the resulting p-schema by expanding its steps, etc. If the step is a goal, then a p-schema is found for that goal and applied. The expansion process continues until an xact is found.

Schema application is the process of partially expanding and executing a p-schema; that is, at no time are inappropriate commitments made to future details of the plan the p-schema represents. This helps ensure that the plan won't have to be changed as the situation changes; instead, future details will be decided upon at the time they must be, not before.

This is one form of least-commitment in Orca. Another is in the step specification. Orca allows three levels of commitment to what to do to accomplish a step; ranked from most to least commitment, they

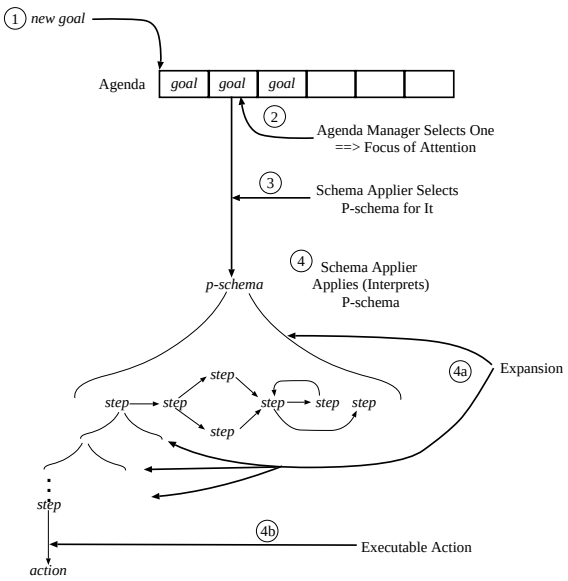

Figure 2: Schema application process in Orca.

are: xacts, p-schemas, and goals. Steps that call for p-schemas partially commit to how to accomplish a subgoal; this is often missing from other work on hierarchical planning, but has implications for efficiency of the planning process and for being able to represent conventional (as opposed to goal-directed) aspects of behavior.

The schema-application process is interruptible after each execution of an xact. SA and AM coordinate their use of the agenda via a semaphore. SA has as its main loop:

```
loop forever {
   Down(semaphore);
   Expand the p-schema for the current goal;
   Execute xact;
   Up(semaphore);
}
```

When AM wishes to (re)focus attention, it does a "Down" on the semaphore. The next time SA executes an xact and does an "Up", AM will grab the semaphore and thus the agenda, and can proceed; SA will meanwhile be blocked.

When SA resumes (i.e., when AM finishes), it may have a different goal in focus. In this case, it switches to applying the p-schema for that goal, thus interrupting work on the previous one. Later, perhaps, the previous goal will become in focus again, and SA will resume where it left off (after checking to ensure that the p-schema is still appropriate for the situation).

Event Handling

Above, we have discussed what happens normally when Orca is conducting a mission. However, things do not always go smoothly: changes in the world or in Orca's perception of the world give can give rise to events that must be detected and handled to ensure the mission's success or, occasionally, the AUV's survival.

There are two kinds of events. *Anticipated events* are those that have been predicted by Schema Applier on the basis of schema application. For example, when SA encounters an xact that orders the AUV is to move to (x,y,z), it posts a prediction to Working Memory (WM) that at a particular time in the future, the

AUV will be at that location, and a prediction that the command it sends to the low-level architecture will be complete. Later, when the AUV reaches that location and/or the low-level architecture notifies Orca that the command is done, the prediction has been met—i.e., this was an anticipated event.

Unanticipated events occur when changes have not been predicted. This does not mean that the event is novel. Someone stepping in front of you as you drive home is not a novel event, but one that is unanticipated: it could not be adequately planned for ahead of time. This is the sense of "unanticipated" we are concerned with in Orca.

Events are handled by Orca's Event Handler (EH) module, as shown in Figure 3. One of EH's main tasks is *situation assessment*, which it does in cooperation with the Context Manager. Within the context of the current situation, it then examines new information to determine if events have occurred. Event detection can be a difficult process, since often events have gradual onset or intermittent presentation. Event detection also uses information about the context from CM. Anticipated events, once detected, are handled by sending a message to AM to notify it of a satisfied prediction. Unanticipated events, however, require further work by EH.

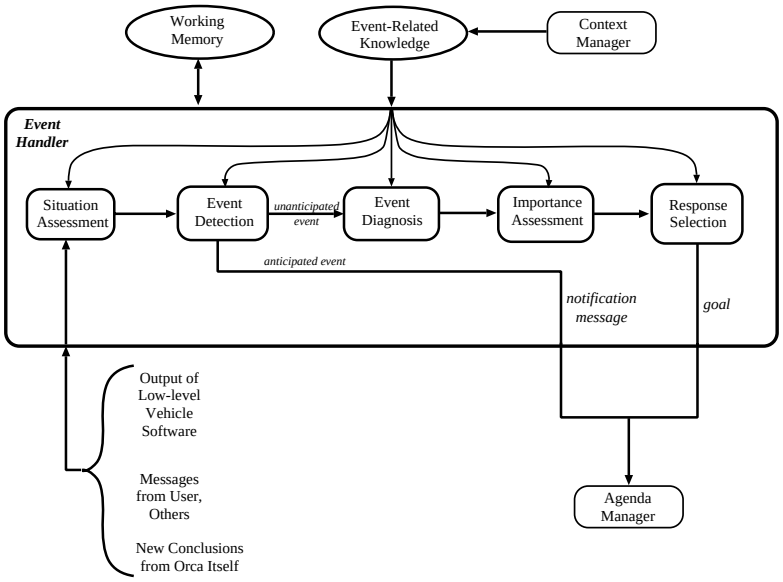

Figure 3: Orca's event-handling process.

When an unanticipated event is detected, it must be diagnosed to determine the underlying event that caused it. For example, the event "motion stopped" may have many underlying causes, including: caught in a net, entangled in kelp, aground, out of power, in a strong current, or thruster failure. The appropriate response depends on which one it actually was. Diagnosis also depends on contextual information; the cause of an observed event can often be partially predicted by the context, and different diagnostic knowledge is appropriate in different situation.

Once diagnosed, an event's importance must be assessed to see if it is worth responding to. The importance of an event also depends on the context. For example, "incipient power failure" is, in most contexts, a very important event that should immediately be handled, or the vehicle will possibly be lost. However, in some contexts, such as rescue missions, vehicle survival may be less important than the successful accomplishment of the mission; if enough power is available for the remainder of the mission, the event of detecting low power should perhaps be ignored.

Finally, if an event is important enough, EH selects a response. In Orca, response to events are goals that are sent to Agenda Manager, along with an estimate of the goal's priority. For example, if power is failing, a high-priority goal to abort the mission and return home might be sent to AM. AM would treat this the same as any other goal; that is, whether or not the response is taken will depend on what else Orca is

doing, as we would expect. Since it has high priority, however, it is likely to immediately become the focus of attention.

Response selection also depends on contextual knowledge, since events should be handled differently in different situations. For example, "power failure" will be handled differently depending on whether: the AUV is still tethered at mission outset (e.g., just send a message to the user); it is transitting to the work site in the open ocean (e.g., surface and radio for help); it is in a harbor (e.g., land and release buoy, so as not to be run over); or it is under ice (e.g., try to get out from under the ice, then land or surface).

Each of the pieces of event handling is handled by a fuzzy rule-based system.[2] Rules for these systems come from the current c-schema via Context Manager, which ensures that EH's behavior is appropriate for the situation.

The process outlined above is somewhat simplistic; in particular, it is too linear. Event handling often must extend over time, with tentative diagnoses and responses activated until new information arrives, at which time new ones will replace them. The above process also ignores the different way events should be handled in cooperative settings, where information and actions can be obtained from other AUVs. For a discussion of future directions for rectifying these problems, see Turner & Eaton [1994] and Turner *et al.* [1994].

Context Schemas and Context Management

Contextual information plays an important role in Orca's behavior, as we have seen above. Context Manager is responsible for maintaining Orca's view of its current context and for sending information to the other modules as necessary to change their behavior to fit the situation.

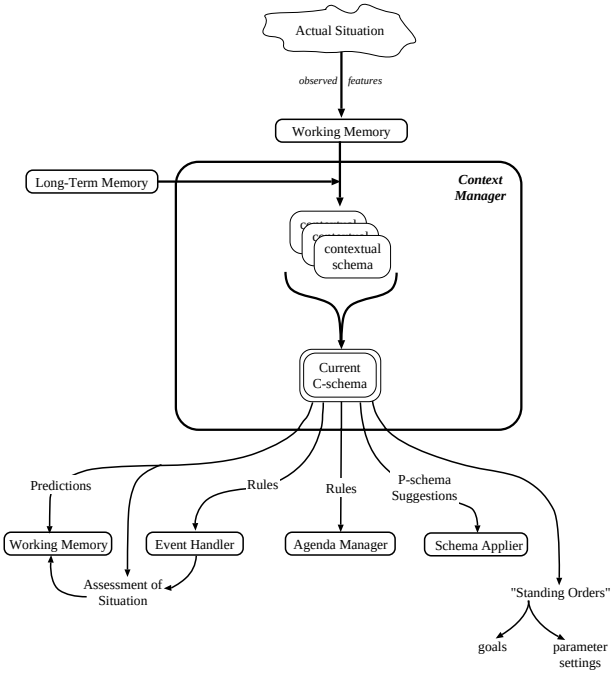

Figure 4: Context management in Orca.

Figure 4 shows how Context Manager manages contextual information.[3] CM watches the evolving problem-solving situation, as reflected in Working Memory, and, with LTM, retrieves contextual schemas

[2] These are expected to be completed by the time of publication.

[3] This is not yet implemented in the current version of Orca, though most of it was implemented in an earlier program, MEDIC [Turner, 1994]. We anticipate implementation of this to begin before the time of publication.

that fit the situation.

A contextual schema represents a context or an important facet of a context—where importance is gauged by the implications for Orca's behavior. For example, if Orca is controlling EAVE during a search mission in Portsmouth Harbor on an incoming tide when power is low, the c-schemas retrieved to fit that situation might be ones representing: "on a search mission", "in Portsmouth Harbor", "operating in currents", and "operating when power is low".

Once a set of appropriate c-schemas is found, they are merged by CM to form a coherent picture of the current problem-solving situation. This results in the construction of a knowledge structure, the *current c-schema*, to represent the current context.

CM sends information from the current c-schema to Orca's other modules to influence their behavior. For example, it sends predictions to WM to help EH disambiguate incoming information as it assesses the situation. Situation assessment is also facilitated directly by the current c-schema: the current c-schema is, in a sense, an assessment of what the situation is. Fuzzy rules from the current c-schema are sent to EH and AM to help them in their tasks, and suggestions of appropriate p-schemas for goals in the current context are sent to SA.

"Standing orders" are also implemented when the context changes. Standing orders are things that should be done automatically when entering a particular context. In human terms, this corresponds to how one would know to lower one's voice when entering a library. These include goals and parameter settings that enforce context-appropriate behavior. For example, when the context becomes "in a harbor", parameter settings may be made to tighten the AUV's depth envelope to avoid bottom clutter and surface traffic.

The overall effect of Context Manager's actions is to tailor all aspects of Orca's behavior to make it appropriate for its context.

Schema Memory

The Long-Term Memory (LTM) module is responsible for storing and retrieving Orca's schemas. Both p-schemas and c-schemas are organized in generalization–specialization hierarchies, as shown in Figure 5 for p-schemas.[4] More general-purpose schemas organize, or *index*, more specific versions of themselves and related schemas. The links between the schemas are feature/value pairs, where the features are aspects of the current situation (including goals) pertaining to the more general schema, and the values are drawn from the indexed schema. The result is a content-addressable, conceptual memory [e.g., Kolodner, 1984; Turner, 1994] similar to those used by many case-based reasoners that allows retrieval of schemas based on features of the current situation. The generalization–specialization hierarchy structure of memory allows the best schema to be found for a particular purpose, but allows *some* schema to be found even if there is no highly-specialized one available.

For example, suppose the current mission is a cooperative adaptive sampling task in which the AUV needs to work with others to gather data in an area. Schema Applier would ask LTM for an appropriate schema. LTM would begin its search at the top of the memory, with `p-achieveGoal`, a very general (and currently hypothetical) p-schema. Features of the task would be used to traverse indices to find other, more specialized p-schemas, such as `p-sample` and `p-cooperate`; this process would continue until the highly-specialized p-schema `p-cooperativeAdaptiveSample` is found. This p-schema would be returned and used.

However, suppose that such a specific schema is not available. Then LTM would retrieve ones intermediate in specificity, such as `p-adaptiveSample` or `p-cooperativeSample`, which could still be used—though possibly with increased effort to apply or with decreased quality of the solution. In the worst case, `p-achieveGoal` would be retrieved; this schema would then lead Orca to compose a plan *de novo* (in essence, a new p-schema) to accomplish the mission.

The result is that the best (i.e., the most specific) possible p-schema is retrieved automatically, but if there is no highly-specialized p-schema, Orca does not fail. Instead, increasingly general-purpose knowledge can be used to assure *some* solution.

The same process provides similar benefits for contextual schemas. If Orca has highly-specialized prior knowledge of a particular context in the form of a specialized c-schema, then that c-schema will be found

[4] The actual memory in Orca is contains different p-schemas than this figure, which was prepared for expository purposes.

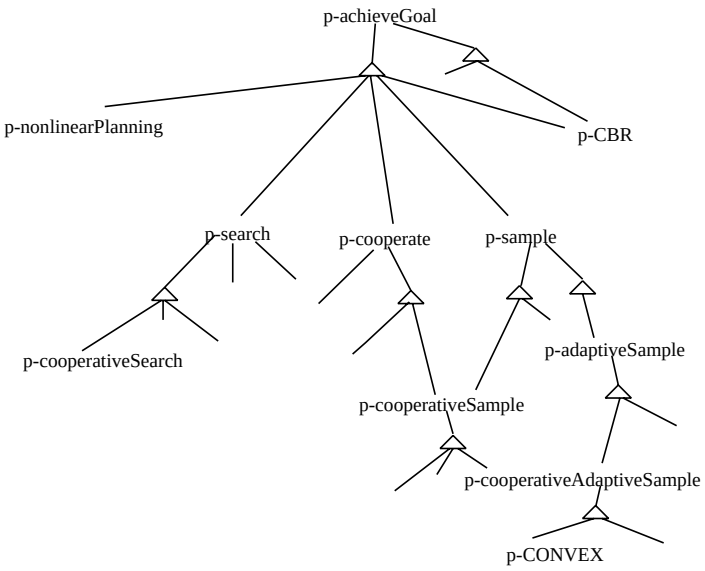

Figure 5: Procedural schema memory structure.

to characterize its current situation. However, if this is not the case, then more general c-schemas will be found and used. For example, if the AUV finds itself in Boston Harbor, but it has no c-schema for how to behave there, it may instead find one for "operating in Portsmouth Harbor" or even "operating in a harbor" and apply information from them to help it behave appropriately in the new context.

Conclusions and Future Work

The Orca project's goal is to create an intelligent mission controller for AUVs and other real-world systems. The task of intelligent mission control demands an adaptive reasoner, that is, one that can change its behavior based on the AUV's current situation. The Orca program is such a reasoner. It is a schema-based reasoner that uses procedural schemas to decide how to achieve goals and contextual schemas to ensure that its behavior is always appropriate to its problem-solving situation. It does not over-commit to future details of its plans, and it remains interruptible should changes occur.

At the time of this writing, Orca 1.1 is in use in our SMART simulation testbed [Turner *et al.*, 1991] by another project. This version of Orca has a very simple Event Handler and Agenda Manager and no Context Manager. Orca 2.0, the version currently under construction, will have better event handling and attention focusing due to the inclusion of fuzzy rule-based systems in EH and AM. The fuzzy RBS should be complete and integrated into EH and AM by the time this paper is published. This version of Orca will also have a Context Manager.

Work on the next version of Orca, Orca 3.0, will concentrate on uncertainty management, a more complete model of event handling [Turner *et al.*, 1994], temporal and spatial reasoning, and fault and error handling.

The next major phase of the Orca project, planned for a year to two years from now, will involve porting Orca to the EAVE vehicles for in-water tests. The versions of Orca in the current phase are being written in the Lisp programming language; it is likely that translation to C or C^{++} will be necessary when putting Orca on actual AUVs.

In addition to single AUV control, we will be combining Orca with work done on communication [e.g., E. Turner *et al.*, in press] to create a competent agent for cooperative distributed problem solving [Turner & Turner, 1991]. We will also be investigating applying Orca to other tasks, such as using it as the basis for an Internet interface agent ("softbot"). This, in addition to Orca's control of AUVs and MEDIC's success in the medical domain, will allow us to determine the generality of schema-based reasoning.

References

Blidberg, D. R. & Chappell, S. G. (1986). Guidance and control architecture for the EAVE vehicle. *IEEE Journal of Oceanic Engineering*, OE–11(4):449–461.

Blidberg, D. R., Turner, R. M., & Chappell, S. G. (1991). Autonomous underwater vehicles: Current activities and research opportunities. *Robotics and Autonomous Systems*, 7:139–150.

Brezillon, P., editor (1993). *Proceedings of the IJCAI Workshop on Using Knowledge in its Context*, Chambéry, France. IJCAI.

Brezillon, P., editor (1995). *Proceedings of the IJCAI Workshop on Modelling Context in Knowledge Representation and Reasoning*, Montreal, Canada. IJCAI.

Brooks, R. A. (1986). A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, RA–2(1):14–23.

Curtin, T., Bellingham, J., Catipovic, J., & Webb, D. (1993). Autonomous oceanographic sampling networks. *Oceanography*, 6(3).

Georgeff, M. P. & Lansky, A. L. (1987). Reactive reasoning and planning: An experiment with a mobile robot. In *Proceedings of the Sixth National Conference on Artificial Intelligence*, pages 677–682, Seattle, Washington.

Kolodner, J. L. (1984). *Retrieval and Organizational Strategies in Conceptual Memory*. Lawrence Erlbaum Associates, Hillsdale, New Jersey.

McDermott, D. (1978). Planning and acting. *Cognitive Science*, 2:71–109.

Turner, E. H., Chappell, S. G., Valcourt, S. A., & Dempsey, M. J. (in press). COLA: A language to support communication between multiple cooperating vehicles. In *AUV '94*.

Turner, E. H. & Turner, R. M. (1991). A schema-based approach to cooperative problem solving with autonomous underwater vehicles. In *Proceedings of the 1991 Conference of the IEEE Oceanic Engineering Society (Oceans '91)*.

Turner, R. M. (1994). *Adaptive Reasoning for Real-World Problems: A Schema-Based Approach*. Lawrence Erlbaum Associates, Hillsdale, NJ.

Turner, R. M. & Eaton, P. S. (1994). Handling unanticipated events during collaboration. In *Proceedings of the Sixteenth Annual Conference of the Cognitive Science Society*, Atlanta, Georgia.

Turner, R. M., Eaton, P. S., & Dempsey, M. J. (1994). Handling unanticipated events in single and multiple AUV systems. In *Proceedings of the IEEE Oceanic Engineering Society Conference (Oceans '94 OSATES)*, Brest, France.

Turner, R. M., Fox, J. S., Turner, E. H., & Blidberg, D. R. (1991). Multiple autonomous vehicle imaging system (MAVIS). In *Proceedings of the 7th International Symposium on Unmanned Untethered Submersible Technology (AUV '91)*.

Turner, R. M. & Stevenson, R. A. G. (1991). ORCA: An adaptive, context-sensitive reasoner for controlling AUVs. In *Proceedings of the 7th International Symposium on Unmanned Untethered Submersible Technology (AUV '91)*.