SITUATION ASSESSMENT FOR AUTONOMOUS UNDERWATER VEHICLES USING A PRIORI CONTEXTUAL KNOWLEDGE

Robert P. Arritt and Roy M. Turner*
Department of Computer Science, University of Maine
5752 Neville Hall, Orono, ME 04469 USA
arritt@maine.edu, rmt@umcs.maine.edu

Abstract

This paper presents a technique for using *a priori* contextual knowledge for performing situation assessment for autonomous underwater vehicles (AUVs). What sets this technique apart from other assessment techniques is that it uses explicitly represented contextual schemas to describe discrete contexts that may occur in the world. A modified version of the INTERNIST–1/CADUCEUS [10] algorithm is then used to diagnose the situation as an instance of one or more of the set of known contexts. The schemas representing these contexts can then be merged to give a coherent assessment of the current situation that can serve as the basis for the AUV's behavior.

Introduction

An intelligent mission controller for autonomous underwater vehicles (AUVs) must be able to quickly and accurately assess its current situation in order to determine how to behave. Ideally, instead of reasoning about the situation *de novo* for each decision it needs to make (e.g., about which action to take to achieve a goal, which sensor to use, etc.), such a controller should always have "in mind" a good idea of what the current situation is. Then information about the current situation can be used to immediately inform and guide decision making.

Over the past few years, much work on the Orca intelligent mission controller [17; 13; 14; 16] has concentrated on devising mechanisms for doing just this. In this work, we make a distinction between a "situation", which includes all features of the world currently observable by the AUV, and a "context", which is a recognized type of situation. Situation assessment then becomes the problem of identifying the context.

There is much evidence that humans, and perhaps other higher animals, make use of *a priori* knowledge about contexts when understanding sensory input and selecting actions [3; 18; 8; 2; 7]. This makes sense. The world is to some extent predictable, with broad patterns that tend to recur. Knowing about these patterns can help an agent by allowing it to make decisions more quickly and accurately when the pattern is recognized.

In Orca, contexts and contextual knowledge are represented explicitly as knowledge structures that the agent can examine and use. Contexts that Orca knows about, each representing a class of situations, are represented by frame-like structures called *contextual-schemas* (c-schemas). The idea is that Orca is initially given a set of contextual schemas obtained by knowledge acquisition techniques from experts on AUV missions, then, over time, the program will add information about new contexts it encounters as well as adjust the information about existing contexts based on its experience.[1]

Each c-schema contains knowledge both about what its corresponding context is as well as what to do when in that context. The c-schema's *descriptive* knowledge provides information about expected features of situations that are instances of the context. This helps to assess the situation, and it also helps Orca disambiguate sensor input, make predictions about potential events, and determine the context-dependent meaning of such things as its fuzzy symbolic knowledge. The c-schema's *prescriptive* knowledge guides Orca as it handles unanticipated and anticipated events, decides how to focus its attention, sets behavioral parameters, and determines which actions are appropriate for the context.

At all times, Orca maintains an idea of what the current context is. This is done by one of its modules, ECHO (Embedded Context-Handling Object),

*This work was supported in part by the United States Office of Naval Research through grants N0001-14-96-1-5009 and N0001-14-98-1-0648. The content does not necessarily reflect the position or the policy of the U.S. government, and no official endorsement should be inferred. Thanks also to the other members of the Maine Software Agents and Artificial Intelligence Laboratory (MaineSAIL.umcs.maine.edu).

[1] The learning aspects of Orca have not yet been addressed in our work.

This paper appears in the *Proceedings of the 13th International Symposium on Unmanned Untethered Submersible Technology (UUST)*, Durham, NH, August, 2003.

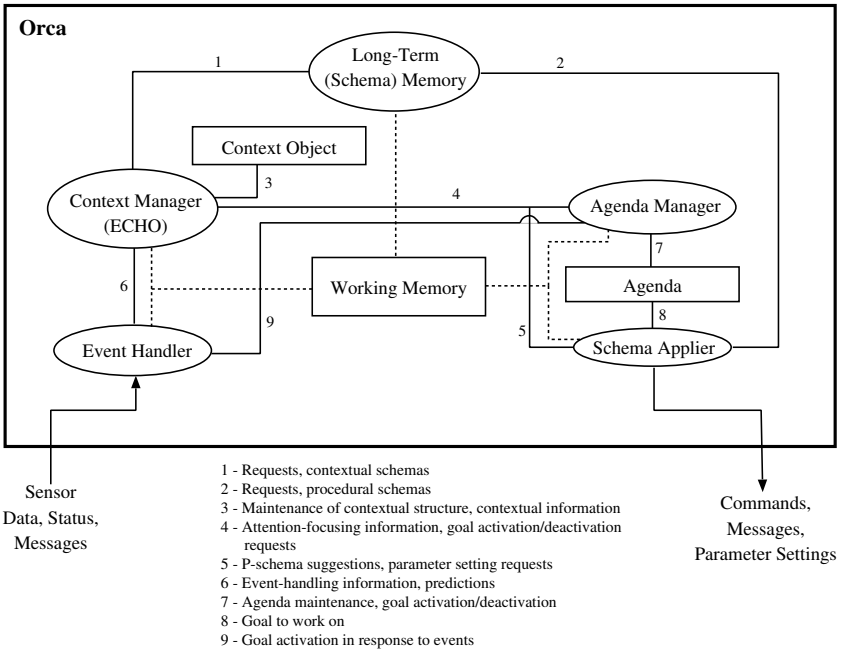

Figure 1: Orca's current architecture.

which uses its *a priori* contextual knowledge—its c-schema repertoire—to *diagnose* the current situation as an instance of one or more known contexts. Figure 1 shows ECHO's relationship to the other Orca modules.

Situation assessment is done by a diagnostic algorithm based on work done in artificial intelligence (AI) in medicine. This is reasonable, since in a very real sense, diagnosis is situation assessment: the purpose is to use the patient's presentation (i.e., his or her signs and symptoms, also called the *findings*) to determine what the clinical situation is so that the physician can know how to act. In our domain, the current situation's features can be viewed as the "signs and symptoms" of the underlying context to be identified. Thus, abductive (hypothetico-deductive) techniques that have been developed for diagnosing a disease from its *manifestations*, the signs and symptoms it causes, can be adapted to identify a context environmental and other features of the situation.

Our algorithm uses the current situation's features to find and return a set of c-schemas that each fit the current situation. These are then merged to give a coherent assessment of the current situation. Information from the merged context is then used by other components of Orca to determine how to behave.

In the remainder of the paper, we first discuss c-schemas in a bit more detail, then we describe our diagnosis-based situation assessment algorithm. We next discuss schema merger and the distribution of contextual information, then conclude and discuss future work.

Contextual Schemas

All of Orca's contextual knowledge is represented explicitly as contextual schemas. This is in contrast to the usual way context is encoded in reasoning systems, for example, as antecedent clauses in rules or as preconditions in plan operators. This has several benefits. It decreases redundancy, since a given context is represented once, not in multiple rules or preconditions. This facilitates acquiring information about context, as well as maintaining the knowledge base. It allows Orca to devote reasoning effort up front to identifying the context so that this can automatically inform all reasoning within the context. It will also ultimately allow Orca to examine and modify its own contextual knowledge.

Contextual schemas contain knowledge about the features of the situation (descriptive knowledge) and knowledge about how to behave when in the context (prescriptive knowledge). Descriptive knowledge includes:

- expected features of the situation: e.g., for a harbor, that the water column will be shallow;

```
C-HARBOR
   :ACTORS
     (actor (variable ?self) (frequency 1.0) (import 1.0))

   :OBJECTS
     (obj (variable ?setting) (description ^place) (frequency 1.0) (import 1.0))
     (obj (variable ?mission) (description ^mission) (frequency 0.5) (import 0.6))
     (obj (variable ?surface) (description ^surface) (frequency 1.0) (import 0.8))
     (obj (variable ?wc) (description ^wc) (frequency 1.0) (import 1.0))

   :DESCRIPTION
     (fx (description (depth ?wc shallow)) (frequency 0.8) (import 0.7))
     (fx (description (and (traffic-volume ?surface ?value)
                           (>= ?value some)))
         (frequency 0.65) (import 0.9))
     (fx (description (fish-density ?setting high)) (frequency 0.8) (import 0.7))

   :DEFINITIONS
     (shallow (linguistic-variable (:slot ^physical-object depth)
                                   (membership-function ((0 1) (10 0)))))

   :STANDING-ORDERS
     (so (condition t) (description (set-LLA-parameter depth-envelope (5 10))
                                    (when :during)))

   :EVENTS
     (ev (description (power-level ?self low)) (likelihood unlikely) (importance critical)
         (effects (^event-desc (description (status ?mission failed))))
         (response (description (achieve (^a-abort)))))

   :ACTIONS
     (a (description (^a-abort) (action (^p-abort-to-bottom))))
```

Figure 2: A contextual schema representing the context of being in a harbor.

- possible features of the situation: e.g., there may be swift currents;
- information about how to interpret sensory information: e.g., sonar contacts overhead will likely be boats or ships; and
- knowledge about the semantics of symbolic information the program uses: e.g., "deep" in this context does not mean the same as in the middle of the ocean, even though the same (fuzzy) term may be used in the same rules governing the depth envelope in both contexts [15].

Prescriptive knowledge includes information about:

- how to set behavioral parameters: e.g., what the appropriate depth envelope is;
- how to handle unanticipated events: how to recognize them, how to diagnose them, how to assess their importance, and how to respond to them;
- how to set goal priorities: e.g., on a rescue mission, goals having to do with self-preservation may have a lower priority; and
- how to select actions that are appropriate for the current situation.

C-schemas are represented in a locally-developed frame language. Figure 2 shows a c-schema representing the context of being in a harbor. The first three sections of the schema contain descriptive information about the context. Each of the three sections contains features expected in the context. Each feature is tagged with the frequency of its occurrence in situations that are instances of this context and its importance.[2] The *:ACTORS* section lists all of the agents that are expected in the context. A representation of the physical embodiment of Orca, bound to the variable `?self`, is the only one necessary in *C-HARBOR*. The *:OBJECTS* section lists all of the objects, both physical and abstract, that are referenced within the schema. The example schema requires a description of the AUV's locale (`?setting`), a representation of the water column (`?wc`), a mission (`?mission`), and a representation of the water's surface (`?surface`). The frequency value for each feature specifies how frequently the feature is associated with situations that are instances of the context, which is an estimate of how probable the feature is. For example, when in the context of a harbor, an AUV may or may not have a mission; this is reflected by a frequency of 0.5 in the mission object. The final piece of descriptive information that the schema gives is *:DESCRIPTION*. This slot contains predicate-like declarations about the expected state of the world. For example, in

[2]The importance values are currently context-independent but are linked into the schemas for convenience

C-HARBOR, we see that the expected depth of the water column is the fuzzy value `shallow`.

Following the descriptive sections are the prescriptive sections, which tell Orca how to behave when in the context. In our example, the first section, *:DEFINITIONS*, contains context-sensitive definitions for fuzzy values (e.g., [19]). In this example, `shallow` is defined as a membership function on depth ranging from zero to ten meters.[3] Next, the *:STANDING-ORDERS* section contains information about how the AUV should adjust its behavior when in the context via setting behavioral parameters. When an AUV is in a harbor, it should tighten its depth envelope, thus the example context contains a standing order to do so. The next section, *:EVENTS*, contains information about how to handle unanticipated events that may occur in the harbor context. The only event described in the example schema tells the AUV to activate a high-priority goal to abort (`^a-abort`) in case of low power. Finally, the *:ACTIONS* slot describes some of the actions that may be needed, in particular, actions that are appropriate in this context for goals that may usually be achieved by different actions. In this context, a good way to handle *^a-abort* is to abort to the bottom.

Contextual schemas are stored in Orca's long-term memory, which is an associative conceptual memory [6; 13]. Essentially, the memory consists of multiple interconnected discrimination networks in which c-schemas form multiple generalization–specialization hierarchies. When presented with a "probe" consisting of a subset of the current situation's features, the memory searches its indexing structure and returns a set of the most specific c-schemas that fit the probe. The probe is said to *evoke* these candidate c-schemas, and they become grist for the situation assessment algorithm described below.

A given c-schema is not intended to necessarily represent all facets of the situations that are instances of it. Rather, the idea is that several c-schemas will be merged to form a coherent picture of the current context. For example, the context representation for a situation in which an AUV is conducting a search mission in a harbor while its batteries are low might be comprised of c-schemas representing: conducting a search mission, being in a harbor, and operating on low power. Merging c-schemas is discussed below. Determining *which* c-schemas to merge is the function of the diagnostic algorithm.

[3]The other specifics of the membership function are defined elsewhere.

Ultimately, Orca's contextual schemas will come both from human experts and from its own experience. The encoding process starts with a domain expert determining the most salient descriptive features of each context. These features then become slots within the contextual schema that will later be used to help diagnose the current context. Next, prescriptive knowledge is added to the schema. In the future, Orca will be able to use knowledge acquisition techniques to create c-schemas based on its own experiences.

Diagnosing the Context

Orca's ECHO module is responsible for diagnosing the current context. When the program starts, and whenever the context has potentially changed, ECHO probes the long-term memory for a new set of c-schemas that might fit the current situation, then applies its diagnostic algorithm and merges the results to complete the process of situation assessment.

The diagnostic algorithm in our work is based on that implemented in the INTERNIST–1/CADUCEUS program [10]. INTERNIST was designed to perform diagnosis in the domain of general internal medicine. It performs differential diagnosis, unlike rule-based diagnostic approaches (e.g., MYCIN [11]). Differential diagnosis is the process by which all possible diagnoses for a given set of findings (signs and symptoms) are grouped and further work is then done to determine the actual diagnosis [4]. This allows diagnoses to be compared and played off against one another, guiding information gathering. When performing diagnosis in Orca, we view the features of the current world as manifestations of some underlying context; this context is then the "disease(s)" that we are trying to diagnose. While work on the actual diagnostic machinery in ECHO is still ongoing, we discuss here the algorithm that will be used.

The overall process is as follows:

1. Identify the set of possible diagnoses (contexts) *evoked* by the observations.
2. Group the contexts into logical competitor sets (LCSs) [1], or "problems".
3. Select one of the LCSs and "solve" it by gathering about the topmost diagnosis or to in some other way separate it from its competitors.
4. Repeat until all important signs and symptoms have been accounted for.

The result, in the case of INTERNIST, is a list of diseases that the patient likely has. For Orca, it is a list of c-schemas that can be merged to become the situation assessment.

The diagnostic process begins with Orca finding all relevant contextual schemas that the current manifestations evoke. This is done by gathering all of the appropriate facts from working memory (a short term memory that contains all of the information that Orca knows about the current world) and using them to construct a long-term memory probe. The information in this probe is then used to guide a depth-first traversal of Orca's long-term memory. This search will return the most specific c-schemas related to the situation.

The memory associates with each returned c-schema an *evoking strength*, a number that indicates how specific the set of findings are for the c-schema. In the medical domain, evoking strength measures to what degree a particular finding brings to mind (evokes) a particular disease hypothesis. It is related to the probability of the disease, given the finding. In our approach, the evoking strength of a particular finding for a c-schema is an indication of the likelihood of the current situation being an instance of the represented context. We are currently investigating whether these evoking strengths should be context-sensitive: that is, should an evoking strength for a particular finding be influenced by the presence of other findings?

The set of candidate c-schemas is just a starting point, however. Though evoked, there may be insufficient evidence to support some, and some may compete with others. Consequently, the set of candidates now undergo differential diagnosis.

A *context hypothesis* is created for each of the candidate c-schemas.[4] Each hypothesis contains additional information:

- a list of all of the features that are both present in working memory and predicted by the schema (called *present–explained*);
- a list of all of the features that are predicted by the schema but that are known to be absent in the current world (*absent–explained*);
- a list of all of the features that are present, but not described by the schema (*present–unexplained*); and
- a list of all of the descriptive features of the schema that nothing is known about (*unknown–explained*).

Next, each context hypothesis is given a score based on comparing the current findings to those predicted to be manifestations of situations characterized by the c-schema. The score is based on three real-valued versions of the functions used in INTERNIST. INTERNIST's scoring algorithm uses integer-based scoring, with evoking strength, manifestation strength (frequency), etc., all taking on small numerical values. In our approach, these are real numbers over the range [0,1]. We fitted INTERNIST's scoring to exponential functions over that interval.[5]

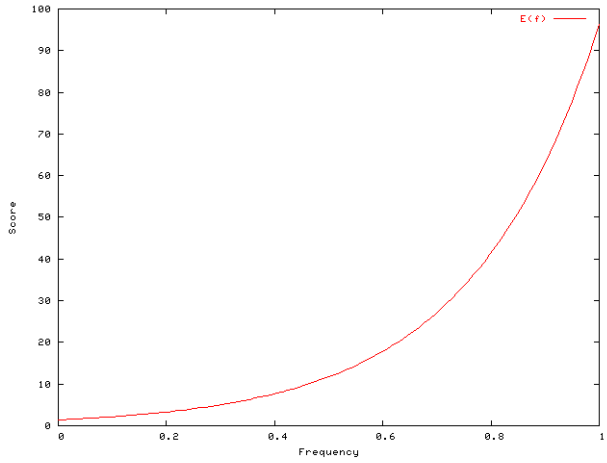

Figure 3: Scoring function based on the frequency of present–explained findings: $E(f) = \sqrt[21]{1600^{(1+12f_e)}}$.

A hypothesis' score should be proportional to the evoking strengths of findings it predicts that are actually observed. Our function for these present–explained manifestations is as shown in Figure 3, where f_e is the evoking strength of a finding f that is explained by the hypothesis.

The score also takes into account two other things. The first is a *frequency* value that the c-schema maintains about each of its manifestations. This is a measure of how often a manifestation occurs when the context occurs. What is used here, though, is the frequency value of each of the absent–explained manifestations. The function shown in Figure 4 is used to penalize the context hypothesis for each of these according to its predicted frequency. In the figure, m_f is the frequency of a manifestation m predicted by the context, but not present.

The second thing that modifies the score is the *import value* of a finding. An import value is a context-independent measure of how important a finding is.[6] For example, a sonar return would have a low import value, while the detection of a leak would have a very high value. A context hypoth-

[4]Called in INTERNIST a "disease hypothesis".

[5]Consequently, the functions may seem a bit odd. However, INTERNIST's functions work quite well in its domain, and so we have not been tempted to change them.

[6]In the future, we will examine context-dependent import values.

esis is penalized for each present–unexplained finding based on that finding's import value according to the function shown in Figure 5, where f_i is the import value of a finding f that is present but unexplained by the hypothesis.

The net score S_c for the hypothesis is the sum of these three scores for all findings:

$$S_c = \sum E(f_{pe}) + \sum A(m_{ae}) + \sum I(f_{pu})$$

where f_{pe} are the present–explained findings, m_{ae} are the absent–explained manifestations, and f_{pu} are the present–unexplained findings.

Once scored, all context hypotheses some threshold value below the top hypothesis are set aside. This threshold value can be tuned to improve the performance of the algorithm. A value too low will cause the algorithm to reason about infeasible hypotheses, while having a high threshold will cause the algorithm to throw out valid hypotheses. Hypotheses that have been set aside may be reconsidered later should their score increase due to further diagnosis.

Next, a competitor set for the top scoring hypothesis is created. A competitor set contains all hypotheses that compete with each other with respect to the findings that they explain. The INTERNIST algorithm makes use of a simple heuristic to create this set:

> Two diseases are competitors if the items not explained by one disease are a subset of the items not explained by the other; otherwise they are alternatives (and may possibly coexist in the patient). [10]

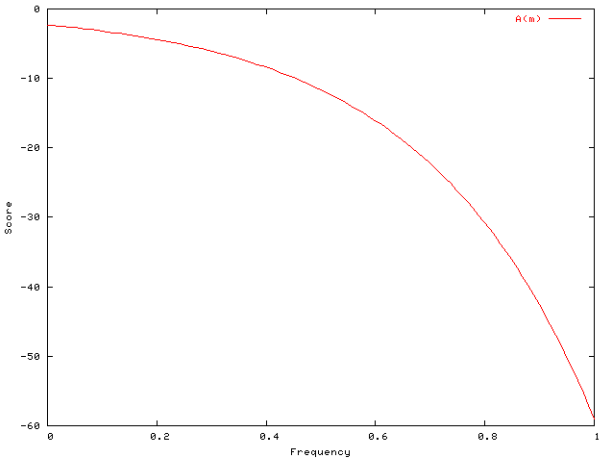

Figure 4: Scoring function based on the frequency of absent–explained findings: $A(m) = -\sqrt[5]{63 * 15^{(6m_f - 1)}}$.

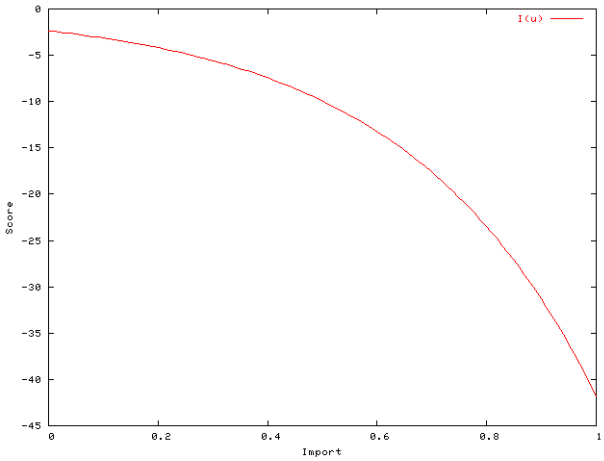

Figure 5: Scoring function based on the import value of present–unexplained findings: $I(f) = -\sqrt[5]{2^{(10f_i+3)}3^{(2-2f_i)}5^{(6f_i)}}$.

With respect to context, this heuristic basically means that if two contexts together explain no more about the current findings than they do when taken separately, then they are competitors.

Once a competitor set is created for the topmost hypothesis, the algorithm checks to see if the top-ranking hypothesis in the set has a score that is more than a threshold value (currently 100 points) greater than the nearest competitor. If this is the case, then the algorithm concludes that the competitor set has been "solved": The hypothesis has been confirmed and the other competitors are discarded. The corresponding c-schema characterizes the situation, and so it becomes one of the c-schemas that will later be merged.

Currently, we follow INTERNIST's algorithm at this point and remove from further consideration all findings explained by this hypothesis. This is not ideal, however, since here the analogy to medical diagnosis becomes somewhat strained. Whereas "cough" is explained by "lung cancer" and can therefore usually be ignored when trying to determine what else is wrong with the patient, something like "shallow depth", though predicted by the c-schema for being in a harbor, is not necessarily caused by it in the same way. Consequently, future work will focus on determining which findings to remove from consideration at this point and how to treat the others. Ideas from other medical diagnosis approaches (e.g., [12]) will likely be useful here.

Once the topmost context hypothesis is solved and its manifestations dealt with, the algorithm continues, creating a new list of hypotheses, a new competitor set for the topmost, and so on. This contin-

ues until all findings with an import value greater than a threshold (currently 0.4) are explained.

If no conclusion can be made, then the algorithm enters a question-asking mode. In the original INTERNIST algorithm, this would require the system to enter one of three questioning modes. If the topmost diagnosis has a score that is almost conclusive (close to 100 points greater than its nearest competitor), the system would enter "pursuit mode", in which questions are asked regarding unknown–explained manifestations with high evoking strengths that belong to the topmost hypothesis. If multiple hypotheses have scores that are close to the topmost hypothesis', the system would enter "ruling-out mode", in which questions are asked about unknown–explained manifestations with high frequency values in the competitors. Finally, if only a few hypotheses are close to the topmost hypothesis, "discrimination mode" is entered, in which questions are asked about unknown–explained manifestations in all of the competitors in the hope that it will spread the scores out.

Orca will use the same modes as INTERNIST, but instead of always using questions, it will use procedural schemas (p-schemas) to gather information. A p-schema [13] can be thought of as an executable plan to achieve some set of goals. All actions in Orca are the result of applying p-schemas based on the situation and its goals. To gather information about a finding, ECHO can post a goal to obtain the desired knowledge. Orca's agenda manager, which controls its focus of attention, and its schema applier, which takes actions based on p-schemas, then would decide whether to work on the goal and how to achieve it.

This part of the situation assessment process is still in progress. In the near future, work will focus on issues related to deciding whether to gather information or not, such as: how to determine how important the information is, how to estimate how long the corresponding p-schema will take to execute, and how many resources it will use, etc. After the information has been gathered (or it has been decided not to gather it), the diagnostic algorithm is run again and, as described before, will continue until all important manifestations have been explained. Upon completion the algorithm will return a list of all concluded schemas.

Merging C-Schemas

Once diagnosed, the set of c-schemas that represent the context must be merged to provide a complete assessment of the situation. This is harder than it may sound and is an active research topic in our laboratory.

Like most merging operations (e.g., [9; 5]), the challenge of merging c-schemas is that they may contain conflicting information. For example, the harbor context, c-harbor, dictates that an AUV should tighten its depth envelope in order to avoid collisions and entanglement. Likewise, the context of a rescue mission, c-rescue-mission, allows an AUV to operate at any depth in order to perform its mission. Now if an AUV is in a harbor and on a rescue mission the diagnostic algorithm will return both c-harbor and c-rescue-mission. If these schemas are to be merged, Orca will have to realize that the depth envelope from c-rescue-mission is more important and use it in the context. How this can be done is still an open question, but we do have some ideas regarding the process.

A naïve approach to merging these schemas would be to give each piece of prescriptive information an importance value. Then when a conflict arose during the merging phase, the most important piece of information could be chosen. For example, a knowledge engineer could tag all of the information within a rescue schema as very important. This would force information from this schema to have precedence over any other contextual information. The problems with this technique are obvious, however: two pieces of information may have the same importance; a choice of one or the other may not be the best way to handle the problem; and the importance of a context may not be a reliable estimate of the salience of its information for a particular situation when compared to another c-schema—for example, when performing a rescue in a harbor, information from the harbor c-schema about currents and bottom conditions should be preferred.

A better method would be to use the frequency values already present in the c-schema along with the diagnostic score S_c of the c-schema to determine which c-schema to use as the source of the information. The scores of the c-schemas give a indication of the relative "fit" of the c-schemas to the current situation, and the frequency value for the information in question would allow a decision to be made about which schema has better information about the particular piece of information.

Still better might be to associate with each kind of information in a c-schema a set of preferred merging operations for the information. These merging operations might be specific to the context, the knowledge contained within a slot, or both. For example, two pieces of fuzzy knowledge might be merged by preferring one over the other, by adding their mem-

bership functions, etc. Allowing merger operations to be specified would give ECHO more options when determining how to merge two disparate pieces of information. This technique obviously would require much reasoning, but we believe it would give better results and be more extensible. It could also handle any type of knowledge that may exist within the schemas.

Distributing Contextual Information

Once the global context has been created, ECHO must distribute the contextual information to the appropriate modules. This can be done in one of three ways. The first technique is to allow ECHO to act as an "oracle" that other modules may ask context-related questions. This has the advantage of being simple to implement, but it requires Orca's other modules to have some idea about context and what they can get from a representation of it. Another way is to have ECHO send specific information to each module as needed. This technique, while more complex, allows for a finer grain of control and leaves the task of understanding context to the context manager. The final technique is to allow ECHO to serve as a sort of information filter that has the ability to "massage" information within Orca in order to add context-sensitivity in a transparent way. Each of these techniques has its pros and cons, which leads us to believe that the final context distribution algorithm for Orca will be a mixture of all three techniques.

Conclusions and Future Work

In this paper we have presented a technique for performing situation assessment using *a priori* contextual knowledge. Specifically, we have discussed representing context and contextual information explicitly as c-schemas, then diagnosing the AUV's current situation as being an instance of one or more c-schemas using a modified version of the INTERNIST–1/CADUCEUS algorithm. The schemas are then merged to give a complete assessment of the situation. This information is then distributed to the rest of the reasoner to guide its behavior.

Future work will focus on completing the implementation of the diagnostic process and fully integrating it into Orca's operation. In particular, we will need to look closely at the issues associated with information gathering in service of situation assessment. We will also need to examine our choices of the threshold values described above to tune ECHO's diagnostic process to the domain. How to merge c-schemas is a major open question that will be the subject of a PhD dissertation in our laboratory. The exact information distribution mechanism will need to be decided upon and implemented as well. We will also look at the question of when to re-diagnose the context: when has the situation changed enough to warrant the effort and time to re-diagnose? Orca will also require many more c-schemas than we currently have, and these will have to be obtained from experts. In the very long term, learning mechanisms will be investigated for maintaining Orca's c-schema repertoire and acquiring new c-schemas.

This will all be done simultaneously with our current project to re-implement Orca in a more modular form for use by other researchers and, ultimately, aboard AUVs.

References

[1] P. J. Feltovich, P. E. Johnson, J. A. Moller, and D. B. Swanson. LCS: The role and development of medical knowledge and diagnostic expertise. In W. J. Clancey and E. H. Shortliffe, editors, *Readings in Medical Artificial Intelligence*, pages 275–319. Addison–Wesley Publishing Company, Reading, Massachusetts, 1984.

[2] E. C. Ferstl. Context effects in syntactic ambiguity resolution: The location of presuppositional phrase attachment. In *Proceedings of the Sixteenth Annual Conference of the Cognitive Science Society*, 1994.

[3] A. N. Glass and K. J. Holyoak. *Cognition*. Random House, New York, second edition, 1986.

[4] R. D. Judge, G. D. Zuidema, and F. T. Fitzgerald. *Clinical Diagnosis: A Physiologic Approach*. Little, Brown and Company, Boston, fourth edition, 1982.

[5] S. Konieczny and R. Pino-Pérez. On the logic of merging. In A. G. Cohn, L. Schubert, and S. C. Shapiro, editors, *KR'98: Principles of Knowledge Representation and Reasoning*, pages 488–498. Morgan Kaufmann, San Francisco, California, 1998.

[6] J. H. Lawton, R. M. Turner, and E. H. Turner. A unified long-term memory system. In *Proceedings of the International Conference on Case-Based Reasoning (ICCBR'99)*, Monastery Seeon, Munich, Germany, July 1999.

[7] G. Mantovani. Social context in HCI: A new framework for mental models, cooperation, and communication. *Cognitive Science*, 20:237–269, 1996.

[8] B. A. Mellers and A. D. J. Cooke. The role of task and context in preference measurements. *Psychological Science*, 7(2):76–82, 1996.

[9] T. Meyer. Merging epistemic states. In *Pacific Rim International Conference on Artificial Intelligence*, pages 286–296, 2000.

[10] R. A. Miller, H. E. Pople, and J. D. Myers. INTERNIST–1, an experimental computer-based diagnostic consultant for general internal medicine. *New England Journal of Medicine*, 307:468–476, 1982.

[11] E. H. Shortliffe. *Computer-based Medical Consultations: MYCIN*. Elsevier, New York, 1976.

[12] J. Sticklen. *MDX2: An Integrated Medical Diagnostic System*. PhD thesis, Department of Computer and Information Science, The Ohio State University, 1987.

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

[14] R. M. Turner. Context-sensitive, adaptive reasoning for intelligent AUV control: Orca project update. In *Proceedings of the 9th International Symposium on Unmanned Untethered Submersible Technology*, Durham, NH, September 1995.

[15] R. M. Turner. Determining the context-dependent meaning of fuzzy subsets. In *Proceedings of the 1997 International and Interdisciplinary Conference on Modeling and Using Context (CONTEXT-97)*, Rio de Janeiro, 1997.

[16] R. M. Turner. Context-mediated behavior for intelligent agents. *International Journal of Human–Computer Studies*, 48(3):307–330, March 1998.

[17] R. M. Turner and R. A. G. Stevenson. ORCA: An adaptive, context-sensitive reasoner for controlling AUVs. In *Proceedings of the 7th International Symposium on Unmanned Untethered Submersible Technology (UUST '91)*, pages 423–432, 1991.

[18] A. Tversky and D. Kahneman. Judgments under uncertainty: Heuristics and biases. *Science*, 185:1124–1131, 1974.

[19] L. A. Zadeh. A theory of approximate reasoning. In J. Hayes, D. Michie, and L. Mikulich, editors, *Machine Intelligence 9*, pages 149–194. Halstead Press, 1979.