In Proceedings of the 1997 International and Interdisciplinary Conference on Modeling and Using Context *(CONTEXT-97); copyright 1997 Roy Turner.*

Determining the Context-Dependent Meaning of Fuzzy Subsets[*]

Roy M. Turner
Department of Computer Science
University of Maine
Orono, Maine 04469–5752
USA
Phone: 1.207.581.3909 Fax: 1.207.581.4977
E-mail: rmt@umcs.maine.edu

Abstract

Much progress has been made in the last few years in both the areas of context-sensitive reasoning and fuzzy reasoning. However, little work has addressed the intersection of the two, yet fuzzy knowledge, like other knowledge, is context-dependent. The meaning of a fuzzy "linguistic value" such as "deep"—that is, the shape of its membership function—depends very much on what the current context is.

In this paper, we describe a mechanism for determining the meaning of fuzzy values from the current context. In this approach, a reasoner uses information about the meaning of fuzzy values contained in *contextual schemas* (c-schemas), which are knowledge structures representing kinds of problem-solving situations. The reasoner retrieves appropriate c-schemas from its memory and merges their information to generate the *dynamic context* knowledge structure. Information from this, when combined with other contextual information, allows the reasoner to determine the context-dependent meaning of fuzzy values of importance to it.

The work is part of the Orca project. Orca is a schema-based, context-sensitive reasoner whose domain is intelligent autonomous underwater vehicle control.

The context-sensitive nature of natural language expressions such as "deep", "large", etc., is well known: "deep" might very well mean something quite different for an autonomous underwater vehicle (AUV) when in a harbor than in the open ocean, and it will undoubtedly mean something different for a shallow-water AUV than for a deep submersible. What is not as well examined, however, is the similar context-dependence of the meaning of fuzzy knowledge.

[*]The author would like to thank the U.S. National Science Foundation for their generous support of this work (grant BES-9696044) as part of the Orca project.

A fuzzy set [Zadeh, 1965] associates with all objects in some domain of discourse a *membership function*; for any object, the function will give a value from the interval [0,1] representing how strongly the object belongs to the set. This membership function is often considered to be comprised of other functions that define fuzzy subsets of the parent fuzzy set. These fuzzy subsets are generally referred to by meaningful names, such as "deep", "shallow", etc. We can think of the subsets being the values of a *linguistic variable*, in this case, "depth". The subsets can be thought of as *linguistic values*. (See, e.g., [Zadeh, 1994].)

In almost all work in the burgeoning field of fuzzy logic (or fuzzy reasoning, or fuzzy control, etc.), a linguistic value has a fixed membership function. That is, if a depth of 20m has 0.7 membership in the fuzzy subset "deep", then it will always have that value.

This is unrealistic, however, especially when we consider that the membership function defines the *meaning* of the linguistic value; for example, the membership function for "nominal" defines what it means for a particular entity (e.g., an AUV) to have a depth considered to be nominal. This meaning is highly context-dependent. As the situation changes, the membership function should change to reflect the fact that the agent (the AUV) changes its notion of what "nominal" means, as shown in Figure 1. In this case, the membership function should change based on the different target depth in each mission.

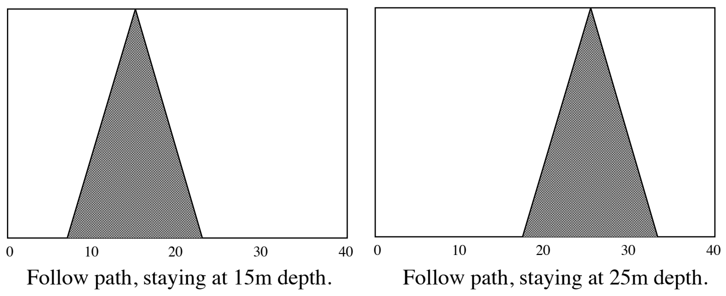


Figure 1: Different membership functions for "nominal" in different contexts.

One could, of course, use different linguistic values or even different linguistic variables to achieve the same purpose. In the situation above, we might define linguistic values "nominal1" and "nominal2", each having the meaning of nominal in a particular context, and each with a different membership function. But this loses information—in particular, the information that somehow the two membership functions, though not the same, are related in their meanings.

More important from a pragmatic standpoint for an artificial reasoner, we are likely to have rules that should apply in both contexts, and that we would not like to change to have different versions for each context. For example, the rule:

```
if depth of AUV is less than nominal
then go down
```

is appropriate in either context. It would be inefficient, increase the difficulty of maintenance, and decrease the understandability of the program (either by a human or by the program itself) to have to have two rules, one for "nominal1" and another for "nominal2". Given that a particular reasoner might operate in many hundreds of different situations, the problem quickly becomes serious.

What is needed, then, is a method of referring to a particular fuzzy subset (linguistic value) whose meaning, as reflected by its membership function, is allowed to change with the context.

Though there has been increasing interest in context-sensitive reasoning in the last few years, including two workshops on the topic [Brezillon, 1993; Brezillon, 1995], most of the work has focused on formalizing context [e.g. McCarthy, 1993; Bouquet & Cimatti, 1995]. The work that has focused on real-world problems [e.g., Turner, 1989; Turner, 1993; Pinto *et al.*, 1995; Abu-Hakima & Brezillon, 1995] has not addressed the problem of how to change the meaning of linguistic values as context changes.

In the Orca project, we have begun to address this problem. Orca [Turner, 1994; Turner, 1995] is an intelligent AUV controller being built at the University of Maine for use with AUVs at the Autonomous Undersea Systems Institute (AUSI) and elsewhere. It is also a laboratory for the development of a variety of artificial intelligence techniques necessary for intelligent agent control. One of these techniques is context-sensitive reasoning.

In this paper, we describe a mechanism being developed to allow Orca to determine context-dependent meanings for fuzzy subsets representing values for linguistic variables. This mechanism is based on the use of *contextual schemas* (c-schemas) [Turner, 1989; Turner, 1994], which represent classes of problem-solving situations. Orca retrieves the most relevant contextual schemas and merges them into a structure representing the *dynamic context* of the agent. From that dynamic context, by a process of lazy (i.e., as-needed) evaluation, Orca constructs the membership functions of fuzzy subsets. This provides a mechanism for the reasoner always to know the context-appropriate meaning of a fuzzy subset.

Context-Sensitive Reasoning in Orca

Orca is an intelligent controller for real-world agents, in particular, AUVs. It is a *schema-based reasoner* [Turner, 1994], meaning that all of its problem-solving knowledge is represented in packets of knowledge called *schemas*. One kind of schema, the *procedural schema* (p-schema), contains planning knowledge equivalent to hierarchical plans, scripts, rules, and the like. The other kind of schema is the *contextual schema* (c-schema), which allows Orca to do context-sensitive reasoning. A diagram of Orca is shown in Figure 2.

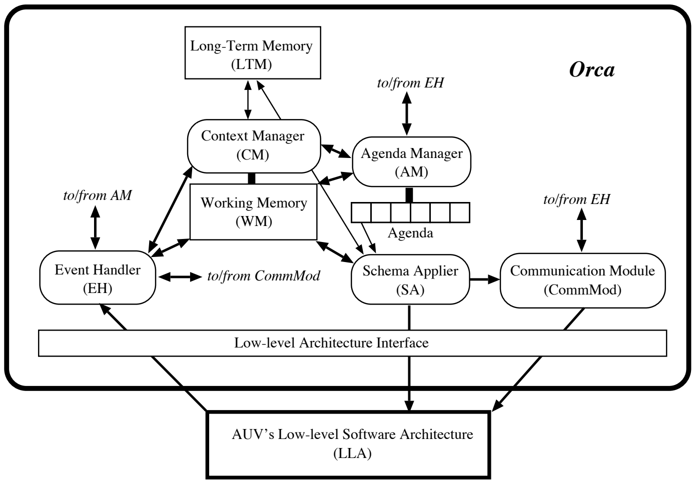


Figure 2: Internal structure of Orca.

Contextual schemas represent classes of problem solving situations. For example, in our domain a reasoner would have c-schemas representing: being in a harbor, being in the open ocean, conducting a search mission, operating under low power conditions, and so forth.

Orca's long-term memory (LTM) indexes its schemas by the features of the situations for which they are appropriate [Turner, 1994]. Based on the features of the current situation, LTM retrieves appropriate schemas and passes them to Orca's other modules. C-schemas are sent to the Context Manager (CM).

There are three kinds of context that Orca is concerned about. The *static context* is that portion of the agent, its environment, and its knowledge unlikely to change over the course of a mission (though it certainly is not static when considered over longer time scales). This includes "default" knowledge such as the AUV's normal operating envelope (e.g., crush depth, maximum speed, etc.). The *dynamic context* consists of the features of the agent, its environment, and its knowledge that do change during the mission. For example, the dynamic context is different when the agent is in a harbor than when it is in in the open ocean or aboard its support vessel. The *ephemeral context* is the context established by the focus of one of Orca's module's reasoning. For example, if one of them is querying working memory about the AUV's depth, then the ephemeral context includes the fact that the AUV's depth is under consideration. This is used, as we will see, to help find and/or create the linguistic value's meaning in the context.

When CM receives c-schemas from LTM, it examines them further to ensure that they are indeed appropriate for the current context. Those that are are used to model the agent's dynamic context; the knowledge structure created is also called the *dynamic context*.

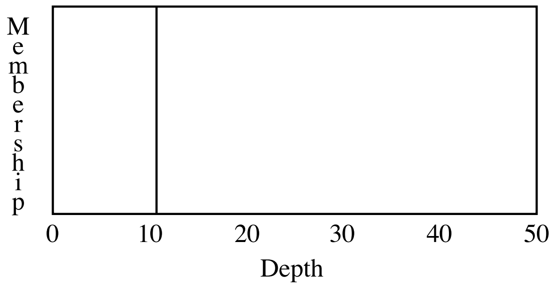
The dynamic context, like c-schemas themselves, contains a great deal of information about situations similar to the context it represents. This information includes: features of the world, agent, or mission that must be or must not be present for the context; predictions about possibly unseen features; predictions about events that might occur, as well as how to handle them (i.e., how to diagnose them, assess their importance, and respond to them); p-schemas that are appropriate for particular goals in the context; importance of working on particular goals in the context; and "standing orders" about what automatically to do when entering or leaving the context [Turner, 1994; Turner, 1995]. Of particular interest to us here is information contained in the c-schemas and dynamic context about the meaning of fuzzy subsets.

Fuzzy Information in Orca

Orca's knowledge representation is frame-based. Some information is represented directly as values of slots of frames; for example, an AUV's depth would be represented in slot `depth` of the corresponding frame (e.g., `^EAVE0`, representing one of the EAVE (Experimental Autonomous VEhicle) AUVs). Orca also represents facts that are not associated with any particular frame, or those that are about multiple frames, as predicate calculus-like expressions in its working memory (WM). Predicates themselves are represented as frames (e.g., `^depth`), so Orca has access to the meaning of the facts by using its frame system.

Orca can associate fuzzy values with either frame slots or facts in WM. For instance, the depth of a sunken wreck might be known only to be "deep", and this fuzzy value would be stored in the frame representing the wreck; likewise, the distance from shore to the wreck might be known only to be "far", which would be stored in WM in a fact about the relative distance. In the case of a frame slot, a description of the fuzzy values it can take on is indicated in the slot's `fuzzy-set` facet, and the actual fuzzy value appears in its `fuzzy-value` facet:

```
^EAVE0:
  depth: 10
    fuzzy-set: ^fs-depth-EAVE-magnitude
    fuzzy-value: ((10 0) (10 1) (10 0))
```



We refer to the things that can have fuzzy values as *linguistic variables* [Zadeh, 1994]. We allow symbolic *linguistic values* to refer to values for these variables. Linguistic values are names for distinguished fuzzy subsets, that is, for membership functions relating numeric values to membership in the subset. Fuzzy subsets' membership functions, in our approach, are *point lists*: lists of points that define a polyline representing the shape of the membership function.

Since the meaning of linguistic values—i.e., their membership functions—change with context, the symbolic names do not usually appear directly as values for linguistic variables. Instead, when a fuzzy value is asserted for a variable, the current membership function—a point list—is inserted into the variable. The symbolic form of the value is only used when asserting a fuzzy value or when querying the value that is there (e.g., to see if the value is "deep"). This way, even when the context changes, the value that was asserted in (and had meaning in) the old context retains an absolute meaning as defined by the point list. This value can then be *reinterpreted* in the changed context.

For example, suppose that Orca is told that the AUV's depth is "nominal" in the context of a mission requiring it to maintain depth at 5 meters. The actual fuzzy value stored in the `depth` slot of the frame might be something like:

```
((0 0) (5 1) (10 0))
```

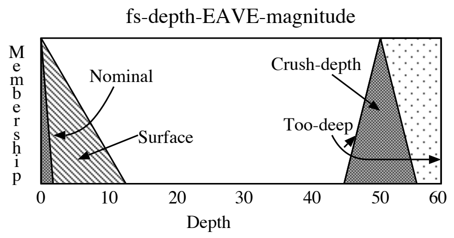
rather than the symbol "nominal". When the context changes to require the AUV to maintain a depth of 10 meters, the point list remains the same and will probably be taken to be "too shallow" in the new context. Had the symbolic form of the linguistic value been used, then Orca would erroneously believe that the depth is still nominal in the new context.

Fuzzy Information in the Static Context

Orca's "static context" is the information it has about the world that does not change over the course of its mission, what we might call its "default" knowledge. Part of this knowledge includes information about fuzzy values.

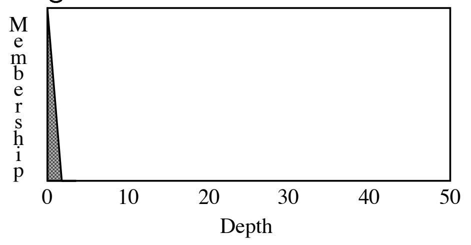
Some information about fuzzy values is predefined in Orca as part of its frame knowledge. Fuzzy sets whose subsets commonly provide values for linguistic variables are defined as frames. For example, the `depth` slot of the frame representing the EAVE, as shown above, is a linguistic variable that is defined by the fuzzy set ^fs-depth-EAVE-magnitude.[1]. This frame has defined subsets, corresponding to distinguished linguistic values, including "crush-depth", "too-deep", and "nominal" (in the default case, at the surface, as a safety measure):

[1]The carat in front of the name denotes that this is a frame.



Each of the defined subsets is in turn represented as a frame. For example, linguistic value "nominal" is defined by frame ^fz-depth-EAVE-magnitude-nominal:[2]

```
^fz-depth-EAVE-magnitude-nominal:
    name: nominal
    domain: (0 2)
    shape: left-triangle
    membership-function: ((0 1) (2 0))
    combination-type: (replace-by any)
```



With each subset is an indication of how to merge its membership function with other values from the dynamic context. For example, ^fz-depth-EAVE-magnitude-nominal specifies that any other information known about the "nominal" linguistic value in the context of the AUV's depth should replace its own membership function. This is a failsafe mechanism in this case: if nothing else is known (e.g., no mission, no other salient context), then Orca should go to the surface.

Fuzzy Information in C-schemas

Contextual schemas provide information about the context-specific meaning of certain fuzzy subsets, in particular, those whose meaning changes in some significant way in the context the c-schema represents. The new meaning is provided in the form of information about what the fuzzy subset's membership function should be in the context the c-schema represents.

Orca cannot be sure that the information contained in a c-schema should completely supplant the default meaning of a subset as provided by its static context: the fit between the c-schema and the current situation may not be exact. In addition, Orca can (and most often will) use more than one c-schema to represent the current dynamic context. For example, the current situation may equally well be viewed as an instance of "in harbor", "on search mission", or "operating under low power". Consequently, information from all sources—static context and all the relevant c-schemas in the current dynamic context—must be merged into a coherent meaning for the subset in the current context.

A c-schema contains information about linguistic values of linguistic variables having particular meaning in the context it represents when that meaning is is likely to be different from that in the static context or in other c-schemas. In our approach, c-schemas contain the following information about each pertinent linguistic value:

- the membership function itself (represented as a point list);
- an *index* that uniquely specifies which linguistic variable/value combination is being referred to;
- the degree of belief (certainty) in the fact that the membership function is as stated; and
- the way in which the membership function should be merged with other knowledge about the membership function that Orca may have at that moment.

[2]The "shape" and "domain" slots are used when creating the frame to define the membership function.

There are three types of indices, each corresponding to a different way Orca can specify linguistic variables and values. First, as we saw above, some frame slots are linguistic variables, such as "depth". The index for these would consist of a description of the slot and the value being referred to, for example: `(:slot ^EAVE0 depth)/deep`. The first part of this index is the variable's *canonical address*: i.e., a unique identifier for the variable.

Second, Orca allows non-frame assertions to be made about the world; these facts, though frame-based (the predicates themselves have meaning as defined by frames), are stored as first-order predicate calculus-like formulae in working memory. It is desirable for these to specify linguistic variables as well. For example, consider the fact:

```
(distance-between ^EAVE0 ^BARGE1 small)
```

In this fact, `small` is the value of a linguistic variable associated with this fact; the meaning of `small` would change with context. The index for linguistic values associated with this fact could be one of:

```
(:fact (distance-between ^EAVE ^BARGE))/small
(:fact (distance-between ^EAVE0 ^BARGE))/small
(:fact (distance-between ^EAVE0 ^BARGE1))/small
```

or others, with the difference being that the frame names not ending in numbers represent classes of things (eg., barges) rather than a particular individual. Thus the first index refers to the value "small" for the linguistic variable associated with the distance between any EAVE and any barge, the second between a particular EAVE and any barge, and the third between a particular EAVE and a particular barge. Each of these combinations may have different implications for what the membership function of "small" should be in different contexts.

Third, other fuzzy information in Orca resides in its frames representing fuzzy sets and subsets. Fuzzy sets specify some attributes of linguistic variables that frame slots (e.g.) use to describe the linguistic variable they represent. Fuzzy subsets describe the default attributes and meaning of some of the values for those variables. One way to indicate sweeping context-related changes of meaning is by allowing a c-schema to temporarily redefine a fuzzy subset. To do this, it may use indices such as:

```
^fs-depth-EAVE-magnitude/nominal
^fz-depth-EAVE-magnitude-deep
```

The first index provides a new membership function for anything that refers to the "nominal" distinguished subset of the fuzzy set; the second directly provides a new membership function for one such subset.

Information about how to merge fuzzy contextual information is in the form of directives that the Context Manager will follow when creating the dynamic context. These include:

- replace other membership functions;
- take the union of this with other membership functions (the current default); or
- use the information only if there is no other information about this linguistic value.

In any case, if no other information exists, the membership function is assumed to be what is specified in the c-schema.

Forming the Dynamic Context

When the context changes (at system start-up, when a new c-schema becomes relevant, or when an old c-schema is no longer relevant), Context Manager's task is to create a dynamic context structure that adequately represents the current context. Part of this consists of taking all information about each linguistic value of interest in the current context and merging it to form the context-specific meaning of that value.

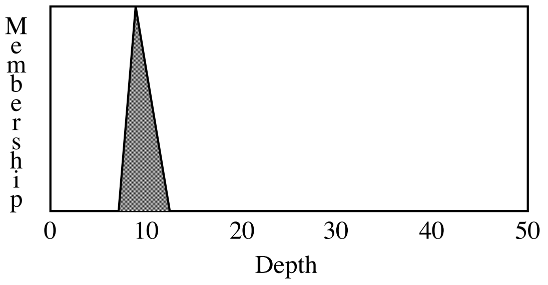
When CM merges information about a linguistic value, it stores it in a table of membership functions in Orca's working memory to allow other modules to access the new meaning of the linguistic value in the context. Two things are important to consider about this table: how the information can be accessed, and how—and under what circumstances—the table is updated.

Access to the table's information is via indices describing the linguistic variable the linguistic value is for. These are the same as the indices described above that appear in c-schemas. To find the meaning of a particular linguistic value, the module wishing to do so first determines what the index is for the linguistic variable in question, then presents it to WM, which returns the membership function. For example, should we want the current membership function for the linguistic value "deep" in the current dynamic context and in the context of the linguistic variable "depth of `^EAVE0`" (the *ephemeral context*; see below), we would use the index: `(:slot ^EAVE0 depth)/deep`.

Updating the information in the table involves merging, for each linguistic value, the membership functions specified in the static context and in all c-schemas comprising the dynamic context that specify information about the linguistic value. CM does this according to the directives present in the context-dependent information.

For example, suppose we are interested in the meaning of "nominal" for the depth of `^EAVE0`. The static context for this linguistic variable is shown above. If we are in the context of a mission requiring the AUV to maintain station at 10 meters depth, then a c-schema that is part of the dynamic context might provide information about this linguistic value of the following form:

```
(lvalue (address (index (obj-desc (:slot ^EAVE depth))
                        (lvalue-desc nominal)))
        (membership-function ((0 0) (8 0) (10 1) (12 0)))
        (combination-type (union)))
```



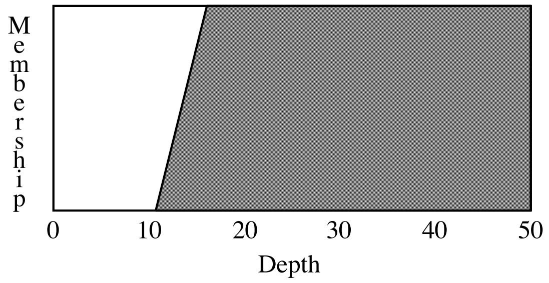
In this case, the dynamic context's information would replace that in the static context because of the static context's combination directive, `(replace any)` (see above).

As another example, suppose EAVE has a crush depth of 50 meters. The static context may indicate this by defining an linguistic value representing "at crush depth" as well as one representing "too-deep"; the latter might have a membership function such as:

```
((45 0) (50 1))
```

as shown above, and the combination type `union`, meaning to take the union of this membership function with any other. The same mission context as before might define:

```
(lvalue (address (index (obj-desc (:slot ^EAVE depth))
                        (lvalue-desc too-deep)))
        (membership-function (10 0) (15 1)))
        (combination-type (union)))
```



The union of these two membership functions would be taken, to give a combined entry in the membership function table of the form:

```
((10 0) (15 1) (50 1))
```

The task of merging membership function information is potentially quite time-consuming, especially when one considers that the context is likely to change often during the course of a mission. To ameliorate this problem, we take a *lazy evaluation* approach. Information is placed in the table and updated only as needed.

We can determine need in one of two ways: when there is no information for the specified index already in the table, and when information is in the table, but is out of date. The first case is easy to detect. To help detect the second, CM generates a unique *context token* each time it changes the dynamic context. When information about linguistic values is placed in the table, it is tagged with the current token. When information is needed from the table, the token associated with the existing table entry is compared to the current context token. If they are the same, the existing information is used; if not, CM is asked to compute a new estimate of the membership function for the linguistic value.

Using the Information: The Ephemeral Context

When an Orca module needs information about a linguistic value's membership function, it must identify the linguistic value it is interested in by more than just that linguistic value's name: it must also specify the linguistic variable for which it is a value. This in turn depends on what the module is "thinking about": if the query is about the AUV's depth, then the linguistic variable is "depth of ^EAVE0", if about the depth at which a sample was taken, then "depth of ^CTD-sample", etc.

We call this the *ephemeral context*: the information about the query used to uniquely identify which linguistic variable is being referred to. This, plus the linguistic value itself, is what WM uses as the index with which to look for information in the table of linguistic value membership functions.

For example, suppose Orca's Event Handler (EH) is trying to determine if the AUV's depth is too deep; if so, then it will recommend the action of ascending in the water column. EH's query to WM might look like:

```
(depth $self too-deep)
```

where `$self` refers to the reasoner itself (e.g., ^EAVE0).

Based on knowledge it has about what the "depth" predicate means[3], WM will determine that this query has a canonical address of: `(:slot ^EAVE0 depth)`. This, along with the linguistic value name "too-deep", becomes the index used to find the proper membership function.

[3]From Orca's frame-based definition.

Predicates that do not correspond to frame slots also have canonical addresses that WM can use as indices. For example, the fact[4] `(distance-between ^EAVE0 ^BARGE0)` has the canonical address:

```
(:fact (distance-between ^EAVE0 ^BARGE0))
```

that is, the fact itself.

WM often needs to do more work than this, however. For example, suppose the query was instead one of the following:

```
(> (depth $self) nominal)
(and (depth $self ?depth) (> ?depth nominal))
```

two different ways of specifying "depth is greater than nominal". In these cases, WM must realize that the ephemeral context is still the AUV's depth, even though the clause specifying this is separate from the linguistic value's name in the query.

Example

Suppose that Orca controls an EAVE AUV with frames as described above. Suppose further that the Event Handler module has the following rules:

```
R1: if depth is deeper than nominal     R2: if depth is shallower than nominal
    then go up                              then go down
```

The antecedent of each contains the query to WM:

```
(> (depth $self) nominal)
```

When the AUV is first launched, EH will analyze the current situation and recognize that the depth is 0. It will query WM to determine how this relates to the nominal depth. The current dynamic context likely will provide no information about the depth. Consequently, WM uses the ephemeral context of the query, looks in the dynamic context structure, finds nothing, and uses the information about the nominal depth from the static context. By this criterion, the depth is nominal, and so neither rule fires.

Suppose that now Orca is given a mission: move to (100,100,10) and hover for twenty seconds, recording temperature data. The dynamic context during the move to the new location should be "moving to waypoint", which will likely not provide any depth information.[5]

However, when hovering, the dynamic context will provide the information that the nominal depth is 10 meters. Assuming that the AUV is at this depth, then neither rule will fire. However, suppose a sudden current moves the AUV to a depth of 9 meters. Now when EH asks WM about the depth in relation to nominal:

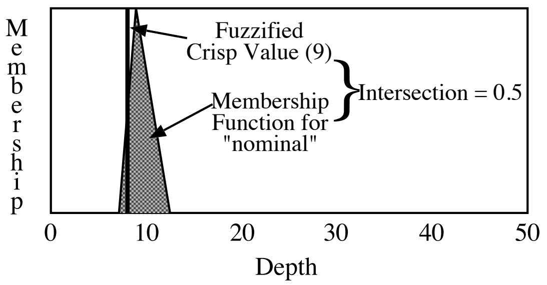
- WM uses the ephemeral context and linguistic value of interest to form the index:

  ```
  (:slot ^EAVE0 depth)/nominal
  ```

[4]This is actually a partial fact—a way of referring to a fact in WM. The full fact would also contain the value for the distance.

[5]Or, rather, the information it does provide will be more complex than what is being discussed here, instead being a motion envelope; adhering to the bounds of such an envelope is left for future work and is beyond the scope of this paper.

- WM retrieves the membership function for this from the dynamic context and compares the current depth (9 meters) to it:



- WM reports that the AUV's depth has only 0.5 membership in the subset nominal; and
- rule R2 fires with a confidence level of 0.5, so EH causes Orca to go down (by asking Agenda Manager to activate a goal to go back to the nominal depth).

We must stress that the actual event-handling process is *much* more complicated than this (see, e.g., [Turner, 1994]), which is reminiscent of a simple fuzzy controller. Note, however, that this technique should be appropriate for such controllers as well.

Conclusion and Future Work

Context-sensitive reasoning is necessary to adequately deal with fuzzy knowledge in realistic domains. The membership functions associated with values of linguistic variables change depending on the context. Without some sort of context-dependent mechanism to determine the membership functions needed, there would be an explosion in the number of linguistic values needed, and reasoning with fuzzy knowledge would quickly become impractical.

In this paper, we have described one such mechanism for determining the context-dependent meaning of linguistic values. Knowledge about the dynamic context of problem solving, from contextual schemas representing aspects of the current situation, is merged with unchanging fuzzy knowledge (from the static context) to generate a membership function appropriate for a linguistic value in a particular situation. As the reasoner needs to compare existing crisp or fuzzy values with known linguistic values, or as it needs to assert that a linguistic variable has a particular value, it looks up that value in a table in the dynamic context using a description of the variable and value as an index. A lazy evaluation mechanism is used to prevent unproductive work updating the table when the context changes rapidly.

The mechanism described here is part of a larger context-sensitive reasoning project, the Orca project. Orca will use this mechanism for event handling and attention focusing. Each of these tasks involves fuzzy rule-based systems that rely on the mechanism described here to evaluate their rule antecedents and to assert fuzzy values by their consequents.

The work reported here is preliminary and in progress. At the current time, Orca's fuzzy rule-based systems are implemented; the mechanism for context-dependent determination of linguistic values reported here is being implemented at the time of writing and should be complete by the time of publication.

Future work will extend this mechanism, particularly with respect to the way in which membership functions can be combined and in resolving the inevitable conflicts that arise whenever merging information from disparate sources. In addition, we will begin to explore whether the mechanism reported here is a reasonable start toward addressing the general problem of context-dependent meaning of knowledge other than fuzzy knowledge.

References

Abu-Hakima, S. & Brezillon, P. (1995). Principles for the application of context in diagnostic problem solving. In *Proceedings of the 1995 IJCAI Workshop on Modelling Context in Knowledge Representation and Reasoning.*

Bouquet, P. & Cimatti, A. (1995). Formalizing local reasoning using contexts. In *Proceedings of the 1995 IJCAI Workshop on Modelling Context in Knowledge Representation and Reasoning.*

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

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

McCarthy, J. (1993). Notes on formalizing context. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 555–560, Chambéry, France.

Pinto, N., Stephens, L., & Bonnell, R. (1995). Organizing domain theories for geographical reasoning using context. In *Proceedings of the 1995 IJCAI Workshop on Modelling Context in Knowledge Representation and Reasoning.*

Turner, R. M. (1989). When reactive planning is not enough: Using contextual schemas to react appropriately to environmental change. In *Proceedings of the Eleventh Annual Conference of the Cognitive Science Society*, pages 940–947, Detroit, MI.

Turner, R. M. (1993). Context-sensitive reasoning for autonomous agents and cooperative distributed problem solving. In *Proceedings of the IJCAI Workshop on Using Knowledge in its Context*, Chambéry, France.

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

Turner, R. M. (1995). Intelligent control of autonomous underwater vehicles: The Orca project. In *Proceedings of the 1995 IEEE International Conference on Systems, Man, and Cybernetics.* Vancouver, Canada.

Zadeh, L. A. (1965). Fuzzy sets. *Information and Control*, 8:338–353.

Zadeh, L. A. (1994). Fuzzy logic, neural networks, and soft computing. *Communications of the ACM*, 37(3):77–84.