Context-Mediated Behavior

Roy M. Turner[1]

[1]School of Computing and Information Science University of Maine, Orono, USA, rturner@maine.edu

ABSTRACT. All effective agents are context-sensitive. This is true for biological agents, and it must also be true for artificial agents if they are to succeed. *Context-mediated behavior* (CMB) is one approach to endowing artificial agents with context-sensitive, context-appropriate behavior. It relies on explicitly representing contexts – important classes of situations – *as contextual schemas* and associating with them all the relevant knowledge about that context. The agent can then use these c-schemas to assess its context and their knowledge to behave appropriately. In this paper, we provide an overview of CMB, review the past work on it, discuss what we have learned from more than twenty years of working on the problem, and describe where we are and where we are going with CMB.

KEYWORDS. agent, contextual schema, agent reasoning, contextual knowledge, context-sensitive reasoning, reactive planner, context-aware applications, autonomous underwater vehicle.

Over the past few decades, there has been an increasing realization that an agent's context heavily influences its perception of the world, its decision processes, and its actions, as well as its interactions and communication with others. While this may have originated in linguistics (and computational linguistics) and philosophy, interest in context quickly spread to many other fields, including cognitive science and psychology, neuroscience, computer science, and computing applications engineering, to name only some.

The emerging consensus is that virtually everything an agent – natural or artificial – does is either affected or even determined by its context. For some agents, there is no conscious attention to the context; i.e., it is not something to be recognized and reasoned about, but rather the context implicitly affects behavior by sensory information being properly linked to behavior, for example, through operant conditioning or other learning.

Humans also are affected by context without often being consciously aware of it. Consider a person's behavior when entering a church or a library: it is likely that they will lower their voice, refrain from running, respect the authority of the priest or librarian, and so forth – all without conscious thought. However, we can explicitly recognize our context and base our actions on that knowledge. A good bit of effort may be expended, for example, to determine what our context is, if we don't immediately recognize it. For example, think about situation assessment on the battlefield or a doctor's effort to diagnosis a patient's problem so that he or she can treat the patient in a context-appropriate manner (prescribe the correct medications, etc.). And we can, if need be, even recognize and reason about contexts that would usually implicitly affect us, such as being in a church or a library.

The questions that have motivated our research on context-sensitive reasoning are: What advantages are conferred by explicitly recognizing the context? Should we emulate that ability in artificial agents? And, if so, how?

Explicitly recognizing the context would seem to cognitive require effort; what does that effort buy? Consider an agent whose behavior is conditioned on its context solely via its sensory input. Many different contexts may look the same based on the percept stream. We would not be very happy with a doctor who noted our cough, fever, pallor, and malaise, and immediately prescribed chemotherapy without considering other diseases, more common and more likely than lung cancer, such as influenza or just a common cold. Instead, the surface percepts (cough, etc.) might suggest one of several different disease contexts, and the doctor would then order tests to distinguish between them to arrive at the correct diagnosis. From there, he or she could prescribe the best treatment. Similar examples can be found in other domains, for example, distinguishing between the context of being ambushed from that of

unexpectedly encountering friendly forces, or for an autonomous underwater vehicle (AUV) between the contexts of being in a strong current and being stuck in seaweed.

When we consider other than sensory aspects of a human's context, then the utility of context recognition becomes even more evident. For example, if a person is in a kitchen in which a pot is boiling, his or her response (turn down the heat, leave it alone, tell someone) should differ depending on whether he or she is the person who put it there or if it was someone else: that is, based on the context, which in this case includes knowledge of goals and past actions.

So, in humans at least, it seems that the ability to recognize and explicitly reason about the context affords the ability to behave more appropriately for the context than if the context was unrecognized.

In addition, by treating contexts as first-class concepts, humans can share knowledge with others about how to behave in the context. If we consider having a patient with a particular disease as a context, then we can think of much of medical school and beyond as the process of communicating about contexts. The same is true for other more mundane situations. For example, when teaching someone to drive in winter, one may tell the learner such things as "if you are approaching a curve and you can't see the road ahead, slow down: there may be snow or ice"; this is describing a context and how to behave in it.

Finally, being able to refer explicitly to a context can help someone learn how to behave in that context. If some action often does not work in a particular context, then by being able to recognize the context as an entity allows the person to not to try the action (or at least be wary of it) in the context in the future.

What about artificial agents and context? Some early work did represent contexts as first-class objects in formal logic, most notably McCarthy [25], Guha [17], and Suvač and colleagues [6], and some work in applications also represented contexts explicitly in a limited fashion, for example, Aikins [1] and Chandrasekaran and colleagues [14]. However, such work was the exception. Throughout much of the early history of artificial intelligence (AI). For example, context certainly affected the behavior of planners, theorem provers, neural networks, and rule-based reasoners, but it was not usually through representing the context as a distinct thing. Instead, context was encoded, somewhat reminiscent of the implicit use of context by animals, as preconditions on plan steps, antecedent clauses in implications, left-hand sides of rules, or weights in networks.

More recently, and especially with the formation of the community of researchers associated with the CONTEXT conference series [5] in 1997 as well as those associated with ubiquitous computing/context-aware applications, context as a thing in its own right has become a very active area of research. While not all of this spate of research is focused on explicit context representation for agents, several strains of research have been, notably that of Brézillon (e.g., [4]), Guha and Lenat (e.g., as discussed in [32]), and Gonzalez (e.g., [16]).

The author's own work on the explicit representation of context began in his PhD research on medical diagnostic reasoning in the mid-1980s. This evolved into the approach to contextual reasoning called *context-mediated behavior* (CMB) [42], part of schema-based reasoning [39] and an off-shoot of case-based reasoning (e.g., [22]).

Context-mediated behavior is based on the idea of explicitly representing an agent's contexts as contextual-schemas (c-schemas), each containing all the agent's knowledge about the context represented. In our approach, a context is a class of situations that has implications for an agent's behavior [42]. For example, "being in a harbor" is a context for an autonomous underwater vehicle that has implications for how it maneuvers, how it understands sensor data, and so forth. Many different situations are instances of this context, for example, being in New York harbor, being in a harbor where there are strong tides, being in Norfolk harbor last Tuesday, and so forth.

When an agent enters a new situation, it attempts to diagnose the situation as an instance of one or more known contexts. These are merged into an explicit context representation, which is then used by the agent to guide all facets of its behavior, including typically such things as setting behavioral parameters, handling unanticipated events, activating/deactivating and scheduling goals, achieving goals, and even more abstract facets of reasoning, such as a context-dependent understanding of concepts. This happens fairly automatically once the context is identified, requiring little additional reasoning on the part of the agent, and indeed likely saving quite a bit of effort. The result is behavior that is automatically context-appropriate.

This paper reviews the past almost three decades of work on context-mediated behavior, concentrating on lessons learned and promising directions for the future. We first discuss the principles we believe should underlie any approach to context-sensitive reasoning, based on our experience with developing CMB. We then describe context-mediated behavior, including where it came from, what the overall process is, and what its contextual schemas contain. We next look at using CMB in agents using several different reasoning styles. The acquisition and learning of c-schemas is next discussed, followed by a look at the current status of the work and future directions.

1. Principles for context-sensitive reasoning

Over the years, we have had cause and opportunity to reflect on the principles that we believe should underlie any approach to contextual reasoning.

Contextual knowledge should inform all of an agent's reasoning. It is difficult to imagine arguing for an agent not taking into account its context when reasoning about how to behave. Almost by definition, successful behavior is behavior that is appropriate for the current situation (context). This is true not only of overt behavior – taking action – but also of the "internal behavior" of the agent, such as making sense of its surroundings, making inferences and decisions, and so forth.

Contextual knowledge and contexts themselves should be explicitly-represented, first-class objects. Some kinds of reasoning make use of contextual knowledge, but do not explic-itly represent contexts per se. For example, rule-based reasoners and planners represent contextual knowledge as antecedent clauses and preconditions (respectively), but do not except in rare cases (e.g., [2]) have anything like a notion of what the context actually is.[1]

There are several important benefits to representing contexts as first-class objects. Doing so allows the agent to reason about contexts as entities in their own right, and so be able to acquire targeted knowledge as well as to learn from its own experience how to behave in the context. Explicit contextual representation also allows an agent to commit to being in a context, in the sense of believing it to be in the context. This can bound the agent's inferences, potentially even allowing the agent to hold contradictory, but useful, beliefs by compartmentalizing them into different contexts (see, e.g., [24, 17]).

While the above benefits can accrue to a reasoner that simply asserts that it is in a named context (e.g., as a result of theorem proving), if contexts are instead considered to be knowledge structures, then there is an additional benefit. A context knowledge structure can contain all the agent's knowledge about a particular context in one place. This makes reasoning about how to behave in the context more efficient, since reasoning can be local to the context, involving primarily the appropriate contextual knowledge. While some work must be done to determine what the context is, after that, reasoning can be quick and sometimes even automatic.

[1]We consider the "context = x" assertion in some rule-based reasoners to be a somewhat degenerate case, since this is used mostly as a filter condition for rules. Similarly for the "filter" conditions in some planner's plan schemas.

While the above benefits can accrue to a reasoner that simply asserts that it is in a named context (e.g., as a result of theorem proving), if contexts are instead considered to be *knowledge structures*, then there is an additional benefit. A context knowledge structure can contain all the agent's knowledge about a particular context in one place. This makes reasoning about how to behave in the context more efficient, since reasoning can be local to the context, involving primarily the appropriate contextual knowledge. While some work must be done to determine what the context is, after that, reasoning can be quick and sometimes even automatic.

Contextual knowledge can be acquired from others, but should also be learned from the agent's own experience. Like any other kind of knowledge, contextual knowledge, including knowledge of which contexts are important, can be acquired from others. Grouping all knowledge about a context together facilitates knowledge acquisition, and explicitly representing the context allows the human to focus on targeted questions, e.g., "in the context of being in a harbor, what depth should be considered nominal?"

An agent should always have an understanding of what its current context is. That is, it should always some context knowledge structure identified as representing the current situation. This is important for the agent to perform reasoning quickly and efficiently. For real-world agents, this is critical, since often unanticipated events occur that demand an immediate response, or the agent's mission, the agent, or both may be in jeopardy. By having a good idea of what the context is, knowledge about how to respond to events is immediately at hand for the agent. In the most extreme case, and depending on the underlying reasoning method of the agent, the knowledge contained in the context representation may serve as a set of reflexes that the agent can carry out in response to events with no reasoning needed at all.

There are times, of course, when an agent is in an unfamiliar situation and may have no specific context representation for it. There are two ways a context-aware agent can still have the advantage of using contextual knowledge, even in this case. First, it is important for contexts to be composable, so that a situation that is similar to several known contexts can be handled by finding and composing those contextual representations into a new one that can provide guidance in the situation. For example, an AUV may know about being in the context of having low power and also the context of being in Portsmouth (NH) harbor, but has never encountered the situation of being in that harbor while power is low; by merging these two context representations, however, it may be able to determine how to behave appropriately in the novel situation.

Second, it is important for context representations to be related to one another by, in particular, generalization–specialization relations. This is a natural outcome of using a dynamic memory such as we mentioned above: the memory organization structure is such that more general schemas index more specific ones in a tangled discrimination net-like scheme. Such a memory then returns automatically the most similar schemas to the current situation. If this type of memory is not used, it is still important to be able to have generalization–specialization relationships between the context representations so that, starting at a very general context representation (e.g., "we're in some context"), other, more specific ones can be found using features of the situation. This way, even if the current situation is not an instance of any specific known context, it will be an instance of some context that can provide at least some guidance. For example, an AUV may never have been in Brest (France) harbor, but if it knows about being in harbors in general, it will have a pretty good idea of how to behave; if it knows about more particular contexts (being in French harbors, which may have different rules, etc., from ones it is used to), then it may be able to behave even more appropriately.

Context representations should represent the smallest useful contexts that cannot be re-created easily from others. One reason for this is to address the potential for an explosion of context representations as the agent gains experience. As mentioned previously, an agent should be able to use multiple context representations to represent a novel situation by merging the contextual knowledge

they contain. The agent *could* just store this merged representation as a new context knowledge structure. However, if it does this for each novel context that is different from existing ones, it will ultimately end up with a plethora of contexts, potentially one for every possible combination of the ones it began work with.

A better approach is to store a new context representation only when it adds something new to the agent's context repertoire. Thus if the knowledge from a merged context worked perfectly in the new situation, there is no reason to store it: it can just be re-created from the same representations should the situation be encountered again. On the other hand, if some knowledge was needed but not present in the merged context, or if predictions failed or behavior was inappropriate, then the merged context *should* be remembered so that the new knowledge can be found or the errors avoided the next time an instance of it is encountered.

An argument can be made for storing some merged context representations, however, to "operationalize" contextual knowledge. Context merger can be quite difficult and time-consuming. In cases where the merger was particularly difficult, it may make sense to store the merged representa-tion to safe work in the future. One should be careful of this, however, since anything new learned about one of the component contexts will not affect the stored context next time it is used, whereas it would have had the context not been stored, but rather re-created.

Context assessment is diagnosis. Assessing the current context involves matching the currently-observed set of situational features to those predicted by the known context representations to find the best fit. This is inherently an *abductive* or hypotheticodeductive process (e.g., [29]) that involves more than just a simple pattern-matching process. It is almost exactly analogous to the problem faced by, for instance, a doctor when deciding what is wrong with a patient: features of the situation (in that case, signs and symptoms, lab reports) are compared to the doctor's known contexts ("patient has the flu", "patient has tuberculosis", etc.) to find the best fit.

Thus, context assessment is a diagnostic process. This means that a context-sensitive agent's designers can benefit from the vast body of research on medical diagnosis. One thing that doctors do that is very beneficial for context assessment is *differential diagnosis* (e.g., [10]), in which similar disease hypotheses are compared to determine information that can rule in/out some of them. This focused information gathering is also of use in assessing the context. For example, if an assistive agent's patient is complaining of not feeling well, the agent might hypothesize two different contexts to explain it: the patient is ill, or the patient did not take his medicine. Differential diagnosis would identify a simple action (count the pills remaining) that could distinguish between the two hypotheses.

Identifying and using appropriate contextual knowledge should be as automatic as possible. Ideally, identifying and using contextual knowledge should require no effort on the part of the agent, which is, of course, unrealistic. However, we should strive for the process to be as automatic and efficient as possible, since we are ultimately interested in the agent carrying out its tasks rather than spending time identifying its context. We have touched on one aspect of this above: the agent should expend effort identifying the context only once, then use the contextual knowledge associated with the context representation until the situation changes sufficiently to warrant a new round of context diagnosis. We should also find the most efficient way for context representations to be found in the agent's memory.

Contextual knowledge and reasoning about the context should be as separate as possible from domain-level reasoning. There are several reasons for this. First, by separating the two things, we may be able to make better use of processing resources, for example, by running the contextual reasoning during idle times or even on a separate processor. Second, it is good programming practice in general to separate a program's "concerns" into different modules or objects, and the same is true here. Contextual reasoning is likely to be different from domain-level problem-solving reasoning, and so it makes sense

to separate the two. This facilitates development and debugging, and it isolates changes in one concern's code from the others.

Third, since context is something that all agents need to be aware of and responsive to, we would like contextual reasoning to be applicable to as many different kinds of agents as possible, even if their reasoning style is different from what is needed for context assessment or needs to be opaque to developers working on contextual reasoning (e.g., if the agent's performance element's code is proprietary or secret). For example, in our own work we have used the same contextual reasoning mechanism for both plan-based and neural network-based agents. While the domain-level knowledge stored in the c-schemas in these agents was necessarily different, the knowledge about contexts, including the structures used to represent them (c-schemas), was the same.

When dealing with other agents, being able to reason and communicate about the shared context is important. Agents, except in rare cases, do not exist in isolation, but rather interact with other agents (human and/or artificial). Consequently, their context also includes the other agents, and in order to predict how the others will behave, it would make sense to have an idea about their contexts and beliefs.

The field of context-aware applications (e.g., [9, 28]) is concerned with this, but quite often the notion of context in that work is somewhat circumscribed (e.g., location, user intent, etc.) compared to that needed for, say, autonomous agents. Nevertheless, we can build on the foundation they have laid, as well as on work done in user modeling in other areas, to develop representations for contexts having to do with humans. Similarly, work in multiagent systems (and before that, in distributed AI) has looked at how agents can model the beliefs and so forth of other agents being cooperated with, and that can be incorporated into contextual reasoning as well.

We should go further, however. It is important for cooperation and coordination that participants agree on what the context is. Early research on this included Sycara's [34] "assumption surfacing" and argumentation in general to get both sides in a labor dispute to agree on a shared context (though those terms were not used) to facilitate negotiation, and work on partial global planning (PGP) [8] in distributed AI can be viewed as a way of getting multiple agents to negotiate to arrive at consensus about what their overall planning context should be (again, not in those terms). More recent work includes our own approach to distributed contextual reasoning in multiagent systems, which uses explicit representations of context as a basis for communication about and agreement on the context of multiagent interactions [45].

2. Origin of context-mediated behavior

Context-mediated behavior has its roots in case-based reasoning [20] and hierarchical reactive planning (e.g., [26, 13]). Many early case-based reasoners, such as Persuader [34], Mediator [21], and CAS [38], used a content-addressable *dynamic memory* [31] based on the CYRUS program [19]. As a by-product of indexing new cases in memory, structures called *memory organization packets* (MOPs) [31] are created by this type of memory. In previous work, the cases were primarily the focus for problem solving. However, it seemed to the author that many, if not most, kinds of reasoning proceed from generalized cases, with specific cases only being used when no such generalized problem-solving knowledge is available or to augment such general knowledge. In other words, generalized cases – the MOPs, which are created by similarity-based learning from cases as they are stored – were being created but not fully utilized.

This realization led to a reasoning style called *schema-based reasoning* (SBR) [39] in which the generalized cases stored in MOPs are used as schemas to drive reasoning. Three kinds of schemas were identified: procedural schemas (p-schemas), contextual schemas (c-schemas), and strategic schemas (s-schemas).

Procedural schemas are the procedural elements extracted from past cases and generalized cases of problem solving. They can represent anything from general goal-decomposition guidance, to hierarchical plans, to specific rules. They are applied in much the same way as plan schemas were the earlier NASL [26] and contemporary PRS [13] hierarchical reactive planners.

Contextual schemas are the generalized cases (and generalizations of generalized cases, and so on). In the original work on the MEDIC medical diagnostic program (e.g., [36, 39]) they represented generalized cases of diagnostic sessions and were called *diagnostic MOPs* (dxMOPs). As it became apparent that they were actually representing MEDIC being in a particular diagnostic context and were more general than just being diagnostic, they were renamed to reflect their role as general contextual knowledge structures.

Strategic schemas were meta-reasoning knowledge structures that governed the overall manner of reasoning. Whereas c-schemas were seen as tactical problem-solving structures, s-schemas were seen as more strategic. For example, the difference between how a novice diagnostician (e.g., a medical student) diagnoses a patient and how an expert does so was represented by an s-schema that modified how knowledge from p-schemas and c-schemas was applied.

While the ideas behind p-schemas and c-schemas continued to be pursued in later work on intelligent control of real-world physical systems (e.g., [46, 39]) and on to the present work being discussed in this article, s-schemas were not pursued. As we discuss later, however, it may be that the notion of strategic knowledge needs to be resurrected, either as a separate kind of knowledge structure or as "meta-c-schemas" representing such things as "being an expert", "being a novice", and so forth.

3. Overall CMB process

The process of CMB has changed over the years. Here, we will describe the current version, similar to what was reported in [43].

The overall CMB process is embodied in what we call a *context manager*, which we name ConMan. ConMan can be an integrated part of an agent's reasoning process (e.g., in Orca [40, 39]), or it can be an independent module that communicates with and provides contextual knowledge to the rest of the agent [43].

A sketch of ConMan's process is shown in Figure 1. When the agent is initialized, the context must first be assessed, as discussed below, by finding a set of c-schemas in its schema memory that characterize the current context. These are merged to form a structure, called the CoRe (Context Representation), that represents the current context.

The CoRe is a part of ConMan's situation representation (SitRep). Another part is the working memory, which holds information about the situation that has been provided by the other parts of the agent, for example, sensor data, messages from other agents, actions taken, and commitments to actions.

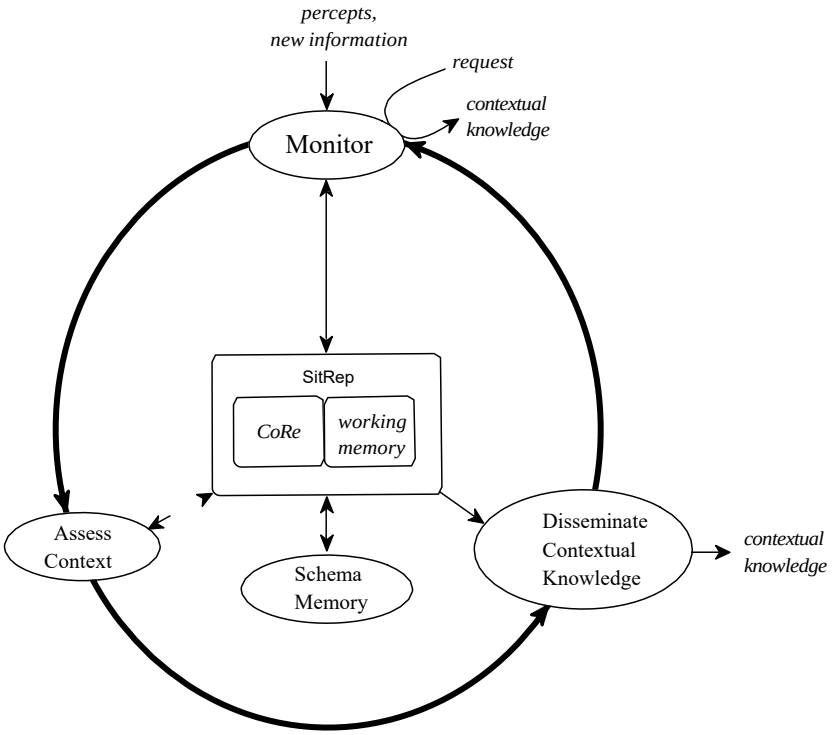

Figure 1. *Overview of CMB process*

Once the context has been assessed, ConMan then disseminates contextual knowledge to other parts of the agent that have requested it (e.g., when the agent and ConMan were configured). For example, if the agent has a module concerned with handling unanticipated events, then at start-up time, ConMan can be configured to automatically send contextual event-handling knowledge to that agent from the CoRe whenever the context changes.

After this, ConMan enters a monitoring mode. While in this mode, it services any requests from other parts of the agent for contextual knowledge, and it places in working memory any information received from the rest of the agent about the state of the world. If during monitoring ConMan determines that the situation has changed enough that the CoRe is no longer an adequate representation of the context, then it reassesses the situation.

One source of information about whether context assessment is again needed comes from the c-schemas comprising the CoRe. These can either explicitly state conditions under which they no longer would apply (cf. [48]), or predictions made by them may be violated to the extent that the monitor function realizes that there is a mismatch.

Another source of information comes from ConMan's schema memory. This is intended to be a semi-autonomous agent inside of ConMan that watches the working memory and, when there is a change, searches its memory structures for new c-schemas that the working memory "reminds" it of [19]. For example, if the depth is decreasing and there is traffic on the surface, an underwater vehicle's schema memory might be reminded of the context of being in a harbor. When schema memory places new c-schemas in the working memory, this should alert the monitor function that context reassessment may be necessary.

In the next few sections we first focus on how contexts and contextual knowledge is represented as contextual schemas, then we discuss the parts of the process in more detail.

4. Contextual schemas

The primary knowledge structure in context-mediated behavior is the contextual schema. A c-schema represents a particular context that the agent knows about and contains all of the agent's knowledge about that context: what it is, how to recognize it, what things mean it may observe while in the context, and how to behave in it.

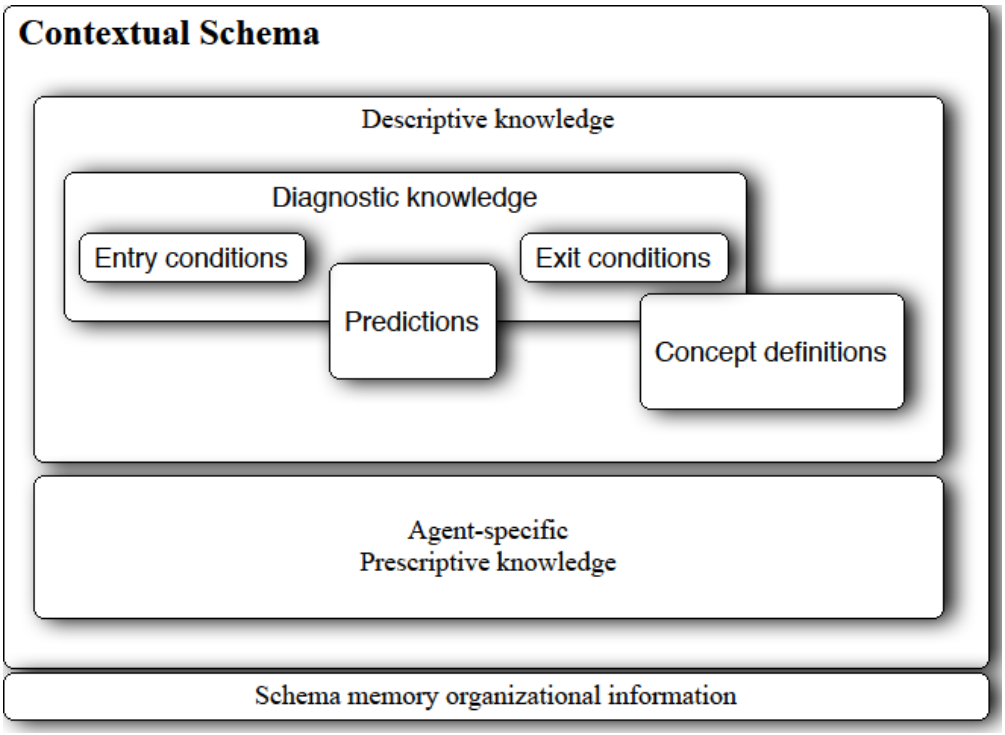

Figure 2. *General structure of a contexual schema*

Figure 2 shows a graphical representation of a c-schema for any CMB agent. There are two major parts, descriptive knowledge and prescriptive knowledge. In addition, depending on the schema memory used, there may be additional organizational/indexing knowledge associated with the c-schema to link it to other c-schemas.

Descriptive knowledge is knowledge about the context itself. The bulk of this knowledge is usually predictions about the features of situations that are instances of the represented context. For example, a c-schema representing the context "driving an automobile" would predict that the agent is in (or part of) a car, the car is on a roadway, there will be movement, and so forth. This knowledge can be used to flesh out the agent's understanding of the situation by providing default features to assume are present as well as to make predictions about what may happen in the future.

The predictions also overlap with the *diagnostic knowledge* in the scheme. This is knowledge that is used by ConMan to diagnose the current situation as being an instance of the context, as we discuss below. The satisfaction of some predictions may be diagnostic or partly diagnostic for the c-schema, for example. For instance a c-schema representing "being in a mined harbor" would predict diagnostic features such as the presence of multiple stationary sonar contacts and that the location is hostile territory. If these predicted features are present, then this would increase the context manager's confidence that this c-schema is a good representation of the current context. Some features might even be "pathognomonic", that is, so characteristic of the situation that diagnosis becomes trivial; e.g., if several redundant sensors all report a leak, then that is pathognomonic for the context of "leaking".

A special case of this type of diagnostic knowledge is an *entry condition*: something that must be true in order to conclude that the agent is actually in the context. For example, for an assistive agent to consider itself in the context "at the patient's home", the location must be consistent with being there.

Other diagnostic knowledge has to do with knowing when the c-schema no longer fits the situation, that is, when the situation has drifted enough to be considered another context (cf. [50]). This can help the context manager, while monitoring the evolving situation, to decide that context reassessment is in order.

A final kind of descriptive knowledge is context-specific *definitions of concepts* (see, e.g., [41]). Some concepts mean different things in different contexts. For example, "too warm" means something different to an assistive agent in the context "preparing food for patient" than in the context of "monitoring the home environment"; in the latter, a temperature of 30°C might be too warm, while in the former it might be 50°C.

Although the kinds of descriptive knowledge present in CMB's c-schemas are the same for all agents, the representation of and ontology for the knowledge will depend at least in part on the kind of agent. For example, when part of Orca, the system in which most of the work on CMB is done, the descriptive knowledge is represented in the agent's own frame-based knowledge representation, augmented with some CMB-specific, description logic-based knowledge about contexts themselves. When used in an agent that uses a different knowledge representation and/or ontology, ConMan would still use its own knowledge representation for some things (e.g., entry, and exit conditions), but it would need to either use the agent's for the rest or to have the ability to translate back and forth between the two representations. In the most extreme cases, for example, when working with a neural network-based agent, there may be little descriptive knowledge that the agent can use or would need; in that case, the knowledge would be limited to what ConMan itself needs to assess the context.

A c-schema's *prescriptive knowledge* is what tells the agent how to behave appropriately in the context. What is contained in this part of a c-schema very much depends on the kind of agent. As this differs substantially from agent to agent, we will discuss this in more detail below when discussing using CMB with different kinds of agents.

4.1. *Assessing the context*

Context assessment is a diagnostic process of determining the context based on a set of situational and agent features. While any diagnostic reasoning approach would work, CMB uses a differential diagnosis algorithm from the field of AI in medicine based on the work of Pople and Miller [27] and Feltovich and colleagues [10].

In this approach, context hypotheses are generated based on which c-schemas are *evoked* – brought to mind – by what the agent knows about the current situation. Schemas are evoked based on rather superficial matches between themselves and the current situation: the stove being on may remind a care-giving agent of the patient being in the context of cooking, of past patients being in the context of cooking, and of the patient's son cooking when he visits. This is similar to, in medical diagnosis, a finding of "cough" bringing to mind a host of things, including the common cold, the flu, bronchitis, and lung cancer.

Just as we would not wish for a doctor to conclude that we have lung cancer just on the basis of having a cough, we do not want a context manager to decide that the patient's son is cooking just because the stove is on. In both cases, more reasoning effort needs to be expended to determine, based on a deeper analysis of differential diagnosis, what the actual disease or context is.

Differential diagnosis is the process of comparing and contrasting hypotheses, based on available as well as acquired information, to distinguish between them. Hypotheses are first grouped into *logical competitor sets* (LCSs) [10] such that each LCS contains hypotheses competing to explain the same set of features.

Hypotheses are scored based on what they explain and do not explain and about failed predictions. The LCS with the top-ranked hypothesis is then selected, and the context manager attempts to "solve" it by winnowing its membership down to one hypothesis. This is done by either performing additional inferencing based on available knowledge and/or acquiring additional information. In either case, the context manager is guided by the available hypotheses: it seeks to separate the ranking of one versus the others based on what information one predicts that others do not.

For example, if an AUV, low on power, finds itself in shallow water and does not have a good idea of where it is (e.g., if it is prevented by the mission parameters from surfacing to obtain a GPS fix), it may have several hypotheses about its context, including: having low power; being in a harbor; being in an estuary; and being over a sandbar. Based on these evoked c-schemas, it would group them into two LCSs: one having the single hypothesis of having low power, and the other containing the hypotheses that all explain shallow water. Suppose that the latter is selected for solution. By comparing the context hypotheses, the context manager would notice some things that could distinguish between the c-schemas: the presence of a nearby shoreline on multiple sides would tend to support being in an estuary or harbor rather than being over a sandbar; the presence of surface traffic would support being in a harbor more than either of the other two; low salinity would tend to support being an estuary; and so forth.

We should mention that while using knowledge already available can often be enough to determine the context, sometimes actions need to be taken to gather additional knowledge, much as a doctor does when he or she orders laboratory tests. So far, we have tended to ignore this in our work, since seeking information by taking actions not only takes time, and so delays context assessment, but also requires collaboration between the context manager and the rest of the agent's reasoning processes to prevent interfering with the primary tasks of the agent. We will consider this problem more in future work.

Once information has been gathered, all of the hypotheses are re-scored and new LCSs are created (since inferences or new information may have uncovered new features to be explained) and the process continues until one hypothesis is sufficiently ahead of the others in terms of scoring. At that point, that hypothesis is considered confirmed (and its LCS solved) and its competitors are deleted from further consideration.

At this point, the context manager is committed to one c-schema as representing the current context. However, just as a person can have multiple diseases, an agent can be in multiple contexts simultaneously: e.g., low power and in a harbor. Consequently, the process continues until all LCSs have been solved, leaving a set of c-schemas that, together, characterize the current context.

Diagnosis is complete at this point. However, to be useful, the knowledge from all the c-schemas just found needs to be merged to create a coherent context representation, which we call the CoRe. We have looked at this problem in some detail [43]. For the present purposes, we only really need to note that it is a very difficult problem that may, itself, be (meta-)context-dependent. For example, suppose that the depth envelope suggested by one c-schema is 2–10 m, but by another is 15–20 m. Should the merged result be 2–15 m? Or one of 2–10 m or 15–20 m? Or the discontinuous envelop of 2–10 m or 15–20 m, but not 10–15 m? It depends in part on why the c-schemas suggest the envelopes they do, which may or may not be pertinent given the particular features of the current situation. And this is for the simplest case, numeric ranges. We have yet to solve the merger problem in a general way, though it is a focus of ongoing and future work.

5. CMB in different kinds of agents

In this section, we look at how CMB has been and can be used in agents employing different kinds of reasoning and at the kinds of prescriptive contextual knowledge important for each.

5.1. *Schema-based reactive planners*

A schema-based reasoner is one that represents all of its problem-solving knowledge in an explicit form as schemas, then retrieves and interprets those to control its behavior. As noted previously, schema-based reasoning grew out of research on case-based reasoning and on hierarchical planning and acting. We coined the term adaptive reasoning to refer to the particular kind of schema-based reasoner we are interested in, one that can change its behavior to fit the needs of the moment as well as over the long term, and that can do both in a context-sensitive fashion.

MEDIC [37, 39] was the first schema-based reasoner. As its name suggests, it was a medical diagnostic reasoner, working in the domain of pulmonology. The focus in the MEDIC project was not on creating a system capable of expert-level diagnosis, but rather on investigating how to capture in a computer program diagnostic reasoning of the sort seen in doctors. Consequently, more attention was focused on the process of diagnosis than on acquiring and using expert knowledge.

MEDIC gave rise to the first contextual schemas, then called "diagnostic MOPs" (dxMOP). These contained such things as goals for the consultation, particular disease hypotheses to consider, findings expected, suggestions for how to establish hypotheses in the context, and event-handling information, where events in that system were interruptions from the user, the creation of new findings (e.g., from discovering a new symptom), and so on.

Orca was the second schema-based reasoner. Its primary domain is intelligent mission-level control of autonomous underwater vehicles (AUVs) and other real-world agents. It has existed in several versions through the years, beginning with one in which all aspects of the reasoner were intertwined, if somewhat modularized, to the current (in progress) version, which is more of an agent-based system with ConMan as one of the agents.

Orca uses procedural schemas to achieve its goals. However, context-mediated behavior permeates all of its behavior and decision-making, including how it understands it world, handles unanticipated events, focuses its attention, and takes actions to achieve its goals.

Figure 3 shows the form of Orca's c-schemas. Prescriptive knowledge in Orca's c-schemas consists basically of four things. First is a kind of knowledge called *standing orders*. Part of this consists of behavioral parameter settings that not so much initiate behavior as modulate it. For example, when entering a harbor, the corresponding c-schema's standing orders may suggest tightening the AUV's depth envelope to keep it away from surface traffic and clutter on the bottom. Standing orders can also specify actions or goals to take when entering a context. For example, when Orca realizes it is in the context "low power", the corresponding c-schema's standing orders may activate the goal to return to base.

In recent work [51, 52], we have begun to examine adding "clean-up" actions and goals to the standing orders that should be taken when leaving a context represented by a c-schema. For example, if someone is in the context "cooking" and decides to leave the house, clean-up actions of the "cooking" c-schema should cause the person to turn off the stove.[2]

[2]We are not proposing this as a cognitive model of human behavior. The cited work focuses on modeling a cognitively-challenged individual for testing assistive technology and to incorporate into games and simulations.

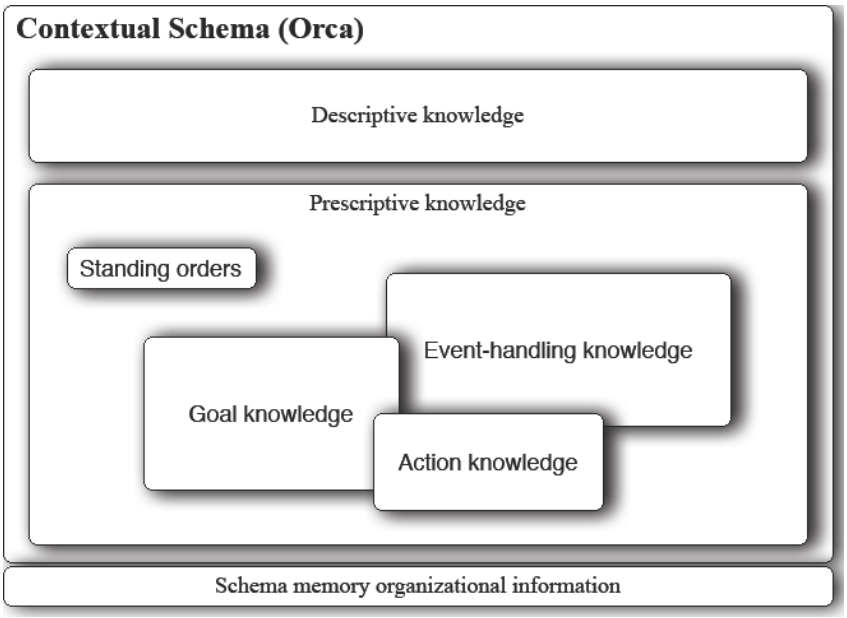

Figure 3. *Structure of a contexual schema for a reactive planner (e.g., Orca)*

A second important kind of knowledge in Orca's c-schemas is goal-related. A real-world agent may have many different goals: recharge batteries, rescue a human, take data samples, determine the current location, and so on. Which one(s) to work on at any given time and their priority is context-dependent. For example, during a rescue mission, taking data samples and recharging batteries are of secondary importance to the actual rescue-related goals.

Goals whose treatment differs from usual in the context are described in the context's c-schema. A goal can be automatically activated (i.e., as part of the standing orders) or deactivated for the duration of the context. Goal importance can also be specified, either as an absolute (fuzzy) importance (e.g., "very high") or relative to another goal ("more important than"). Thus goal-related knowledge focuses Orca's attention on the appropriate thing to work on, given the context, by telling it which goals should be pursued and the order of their importance.

A third kind of prescriptive knowledge helps the agent determine how to achieve its goals. For a schema-based reasoner such as Orca, this consists of descriptions of procedural schemas that are appropriate in the context. These can be linked to goals described in the c-schema, or they can just be available for when other goals arise. Although any p-schema Orca knows about can be found by searching its memory, some are listed in c-schemas for two reasons. First, a p-schema that can achieve a particular goal can serve as a starting point to search for the best way to achieve that goal, thus focusing the agent's attention on a part of memory and short-cutting perhaps a good deal of search. Second, and more important, listing a p-schema in a c-schema embodies the knowledge that the it (or one of its specializations) is the appropriate way to achieve the associated goal in the context.

A fourth type of Orca's prescriptive knowledge is about how to handle unanticipated events. For a real-world agent, surprises are a way of life: equipment fails, other agents do unexpected things, sensors give bad data. Elsewhere,[3] we have labeled this as being caused by CRUD: Complex missions and domains; Real physical systems; Uncertainty; and a Dynamic world. Consequently, there will be unanticipated events that occur during such an agent's mission that must be handled or the agent risks its mission and itself.

[3] In a talk ("Intelligent mission planning and control of autonomous underwater vehicles") given by the author at the ICAPS Workshop on Planning Under Uncertainty for Autonomous Systems, 2005.

Orca's c-schemas provide context-specific knowledge about unanticipated events.[4] One part of this is knowledge allowing the agent to realize that an event has occurred. For example, if an AUV consistently is late arriving at waypoints, there may be a systemic problem, such as a loss of power or a severe current, that needs to be attended to. C-schemas can provide context-specific hypotheses about events that may have happened. For example, in a hostile area, detecting a sonar ping would be unexpected and potentially dangerous, since it could signal an enemy hunting for the AUV.

A second kind of knowledge has to do with diagnosing the event once it is noticed. For example, if an AUV stops its forward motion, it could be because of the loss of a propeller, loss of a thruster motor, being stuck in seaweed, caught in a current, etc. A c-schema can provide context-specific diagnostic information. In the example above, a c-schema representing being in a hostile area should provide diagnostic information that suggesting that the agent consider the hypotheses of an enemy hunting the AUV, an enemy operating sonar but unaware of the AUV's presence, and so forth; these would not be hypotheses considered, for example, upon hearing a sonar ping in the AUV's home harbor. In addition, the c-schema should provide context-specific ways of proceeding with diagnosis to determine the correct hypothesis (e.g., use passive rather than active sonar to identify the potential enemy, etc.).

Event-handling knowledge also includes information about the meaning of the event in the current situation. This can be as simple as an importance estimate or as complex as knowledge describing the likely outcome should the event be ignored. Based on this, the agent can determine whether it should attempt to handle the event or not.

Finally, event-handling knowledge contains suggestions about how best to handle the event in the current situation. Depending on the event and context, this could be as direct as specifying an action for the agent to take ("power failure" → "surface"), less direct, by specifying a specific goal to activate ("message requesting location" → "determine location and reply"), or very indirect, by specifying a very general goal ("sonar ping" → "determine source of sonar ping").

We should stress at this point that all of the information in a c-schema should properly be viewed as heuristic, or default, knowledge about how to behave in a context. It may be augmented by the agent's "default" knowledge about the world, other c-schemas, or the agent reasoning about the world.

5.2. *Neural network-based reasoners*

Great strides have recently been made in the use of neural networks for a wide range of problems in AI, in large part due to better hardware (particularly the widespread availability of high-performance GPUs), new kinds of networks (e.g., convolutional networks), and new and better learning algorithms. Some years ago, we augmented a simple neural network-based reasoner with context-mediated behavior [3], and we are currently looking at the problem again with an eye toward making use of the advances that have occurred in the intervening years.

There are several important benefits to using CMB with a neural network-based agent. The first is the possibility to have the same agent behave in very different ways in different contexts. This is very difficult with neural networks, given that they learn broadly based on their training sets and so may overgeneralize to the extent that their behavior is not as appropriate to a specific kind of situations as one may like. CMB, however, can give a neural network-based agent the ability to have different, highly-specific weights and structures for different classes of situations – or even completely different kinds of neural networks in different situations. For example, in a situation calling for interpreting images from a camera, a convolutional network could be specified, while in a situation requiring

[4]We use the term "unanticipated event" here not to mean completely unexpected, but rather in the sense that someone stepping in front of a car while one is driving is unexpected: it may be known to be possible, but it is not possible to specify when or even if it will happen in a particular driving situation.

prediction based on behavior over time, a long short-term memory (LSTM) [18] network might instead be used.

A second, related benefit is that by partitioning the weights by context, the agent gains the ability to learn in context, then store those weights upon leaving the context for future use. Only those weights having to do with the context are changed. Not only that, but training time should be much shorter, since the network for a particular context is likely to be much simpler than for a network that learns about all contexts at once.

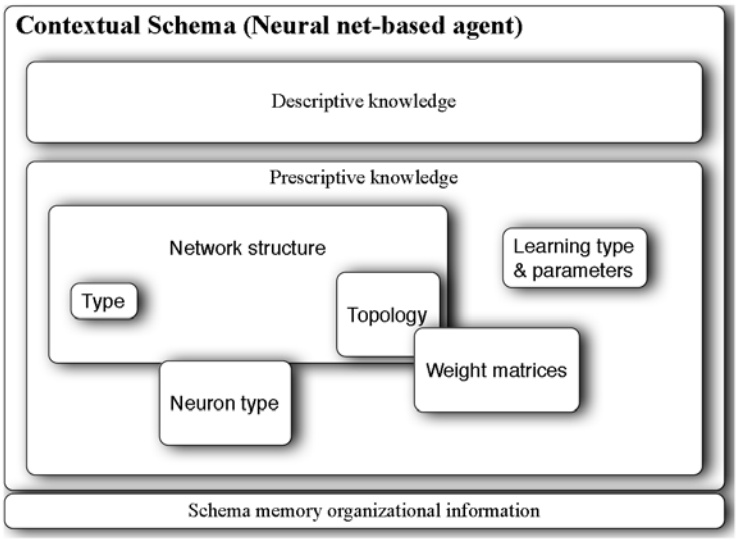

Figure 4. *Structure of a contexual schema for a neural network-based agent*

Figure 4 shows a c-schema for a neural network-based agent. Here, we do not mean an agent that is governed by a single neural network with a static structure, but instead an agent that is capable of executing a given neural network based on a description of the network, much as is done when using such neural network software packages as Keras, TensorFlow, or Theano. ConMan would provide such an agent with knowledge about context-specific neural networks. In the worst case, a "default" c-schema would provide the agent with a general-purpose network trained to do whatever the designers of the agent determine is appropriate when in no known, specific context.

The prescriptive part of a c-schema for this kind of agent needs to provide a specification for a neural network that is appropriate for the context. One kind of knowledge needed is about the network's structure. This includes the type of network (densely connected feedforward network, recurrent network, convolutional network, etc.) as well as the topology appropriate for the situation (e.g., how many layers, how many neurons in each).

A second kind of information would be the kind of neuron to use, for example, the transfer function for a simple neuron (e.g., logistic, ReLu, tanh, etc.) or a more complicated kind of neuron (e.g., an LSTM node) and its parameters.

A very important kind of knowledge is the set of weight matrices containing the network's learned "knowledge". These, along with information about the topology, etc., can be given to the network interpreter to implement the neural network in the situation.

A final kind of information that a c-schema can store for neural networks is the learning type and any learning parameters. For many kinds of agents, this will be rather simple: just backpropagation and the learning rate. For others, however, it may be much more complex. For example, if an agent uses a

neuroevolutionary algorithm, such as NEAT [33], then the learning information could include a population of genomes for the genetic algorithm portion of the agent's learning algorithm.

5.3. *Rule-based reasoners*

Rule-based systems (RBSs) have been and continue to be useful in many areas. They provide a simple formalism for representing knowledge, and, depending on the direction of reasoning (forward or backward), they can solve synthesis problems (forward) or analysis problems (backward), or both. Fuzzy rule-based systems (FRBSs) extend RBSs with rules and concepts making use of fuzzy logic and fuzzy set membership.

Although we have not incorporated CMB into an agent that is completely based on a fuzzy rule-based system, we have gained some relevant experience in a project that entailed incorporating fuzzy rule-based reasoning in Orca's event-handling module. Much of what we learned there could be carried over to any effort to add contextual reasoning to a FRBS or RBS.

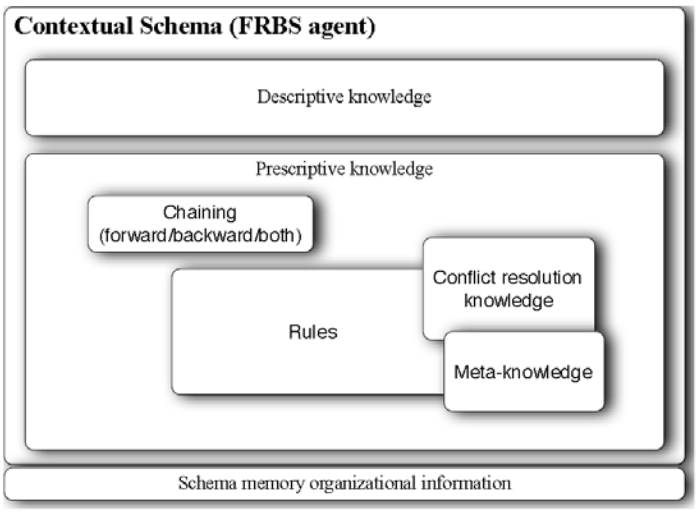

Figure 5. *Structure of a contexual schema for a (fuzzy) rule-based reasoner*

Figure 5 shows what a c-schema would like in such a system. Here, we see agent-specific predictive knowledge to influence this style of reasoning. In particular, the c-schema provides context-specific rules for the agent to use while in the context represented. This essentially partitions the agent's rulebase into context-appropriate packets. Not only does this increase the specificity of the agent's rules (and, hence, its behavior), it also increases the agent's efficiency, since it reduces the size of the rulebase to be searched. The rules could exist as explicit rules, or they could be stored as a pre-compiled structure, such as a Rete network [11]. Other useful knowledge could also be provided, such as knowledge about how to resolve rule conflicts and other "meta-knowledge", as well as the direction of chaining to use when applying the rules in a particular context.

5.4. *Other reasoning styles*

Any agent that operates in multiple different contexts needs a way to tailor its behavior to the context it is in at any given time. We believe that virtually any reasoning style can benefit from incorporating or being assisted by context-mediated behavior.

We intend to investigate applying CMB to several other reasoning styles in the future. For example, a recent project we were involved with dealt with using learning classifier systems (LCSs) [10] to model agents in fisheries, a coupled human–biological system [53]. Many of the same advantages seen for rule-based and neural systems are likely to be realized using CMB with LCSs, as well; in this case,

a c-schema would contain context-specific populations of rules that have been evolved for the context, learning parameters, etc.

There are other possibilities, too. Bayesian networks could benefit from context-mediated behavior by utilizing context-specific probability distributions. Constraint satisfaction would benefit from CMB by, e.g., having c-schemas suggest appropriate heuristics based on the context, where "context" here would have to do with the features of the CSP being solved (cf. textures in constrained heuristic search [12]). CMB could benefit state-space search by suggesting heuristics as well as search styles and strategies given properties of the problem. Adversarial search – game-playing – could benefit from the same kinds of things as well as by changing behavior based on the context of whom the agent is playing (e.g., by using static evaluation functions specific to opponents' playing styles).

5.5. *Multiagent systems*

In the past few years, we have begun to look at how context-mediated behavior can be extended to multiagent systems. An agent in a multiagent system (MAS) needs to consider its own local problem-solving context, but its context also includes the presence of other agents with whom it is interacting. This extends its range of possible actions (e.g., it can ask others to take on some of its goals), but also constrains its behavior (e.g., it may have commitments to other agents that must be honored) in complex ways.

This implies that the descriptive and prescriptive knowledge in an agent's c-schemas needs to be extended to include knowledge about other agents with whom it may be interacting. This includes knowledge of how they are expected to behave, how it can and should communicate with them, and knowledge of any credibility issues or any authority relationships that may exist.

We have looked at context-mediated behavior in the case of an agent involved in a MAS in some of our projects. An early project looked at extending Orca for use in a two-AUV imaging system called MAVIS [44]. This project added information about communication to Orca to guide an agent in how it should communicate with its peer.

We have also looked at extending Orca to function as an AUV controller in an autonomous oceanographic sampling network (AOSN) [7] comprised of many AUVs and other instrument platforms of varying levels of intelligence. The CoDA (Cooperative Distributed AOSN controller) project [47] focuses on organizing and reorganizing AOSNs as the situation changes. Part of the project involves using various organization protocols to self-organize a set of instrument platforms based on the situation and, when the situation changes, to reorganize appropriately. Although this does not necessarily require context-sensitive reasoning, it can benefit from it.

A recent project looked at making context-sensitive trust decisions in a multiagent setting [48, 49]. Here, contextual schemas were used to recognize the context in a competitive multiagent system in which there could be untrustworthy ("miscreant") agents, then used contextual knowledge to choose the best strategy (where "strategy" here means a mapping from situations to actions).

An entire MAS, no less than a single agent, must behave appropriately for its context. Its organization, the assignment of agents to roles, how agents communicate with each other, how it responds to unanticipated events – all of these are context-sensitive, where the context is the entire system's rather than any particular agent's.

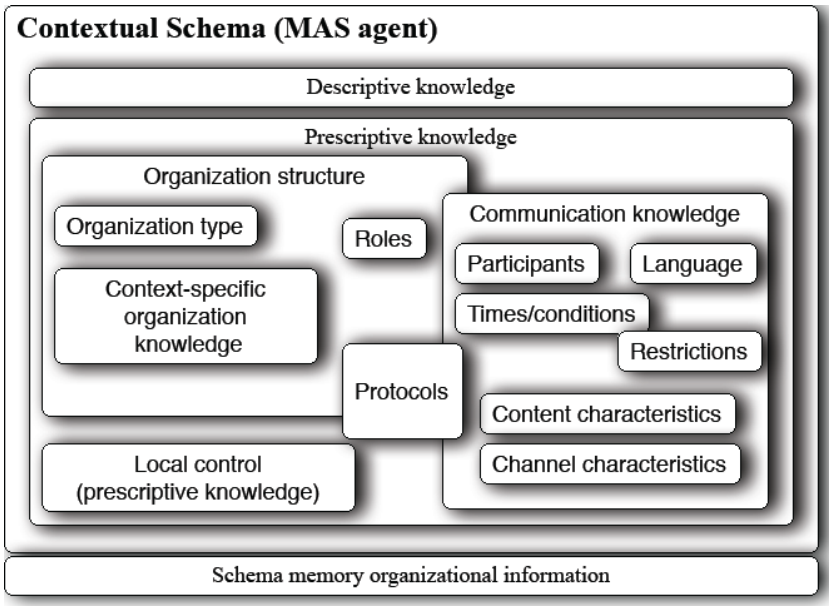

Figure 6. *Structure of a contextual schema for an agent participating in a multiagent system. Note that it would also include prescriptive knowledge pertinent to the reasoning style of the agent for local control.*

In the CoDA project, we have also examined agents can recognize, based on the context, which kinds of organizations should be used in a situation. Figure 6 shows what a c-schema used for this would look like. The prescriptive knowledge in this case still contains information about the actions the agent should take to achieve its own goals, how to handle unanticipated events, and so forth. But now that is only a subset of the prescriptive knowledge encoded in a c-schema.

One important kind of contextual knowledge here is information about the appropriate organizational structure for the situation [45]. This includes the suggested type of organization (e.g., a hierarchy, a bidding system, a team, etc.), as well as parameters for the organization that are specific to the context, for example, the number of layers in a hierarchy, the roles and authority relationships, etc.

Knowledge about communication would also be provided by c-schemas. Here, the knowledge includes such things as the agent language to use, whom to talk to, when or under what conditions to communicate, any restrictions on communication (e.g., “radio silence” in hostile situations), the kinds of things to communicate, and any characteristics of communication channels that may impact communication (e.g., the very low bandwidth of underwater acoustic communication channels).

Finally the protocols the agent uses to participate in the MAS are also context-dependent and should be provided by the agent’s c-schemas.

Going beyond this work, we have very recently begun to look at distributing the process of context assessment and contextual reasoning in general across the agents in a MAS [45, 30]. This is in addition to the agents’ own context-mediated behavior, but it follows a similar process. Agents communicate about the context explicitly and share information about the context that they may have that other agents may not. For example, during context assessment, each agent may be reminded of contexts they each know about, and they share this information with each other prior to the (distributed) differential diagnostic process.

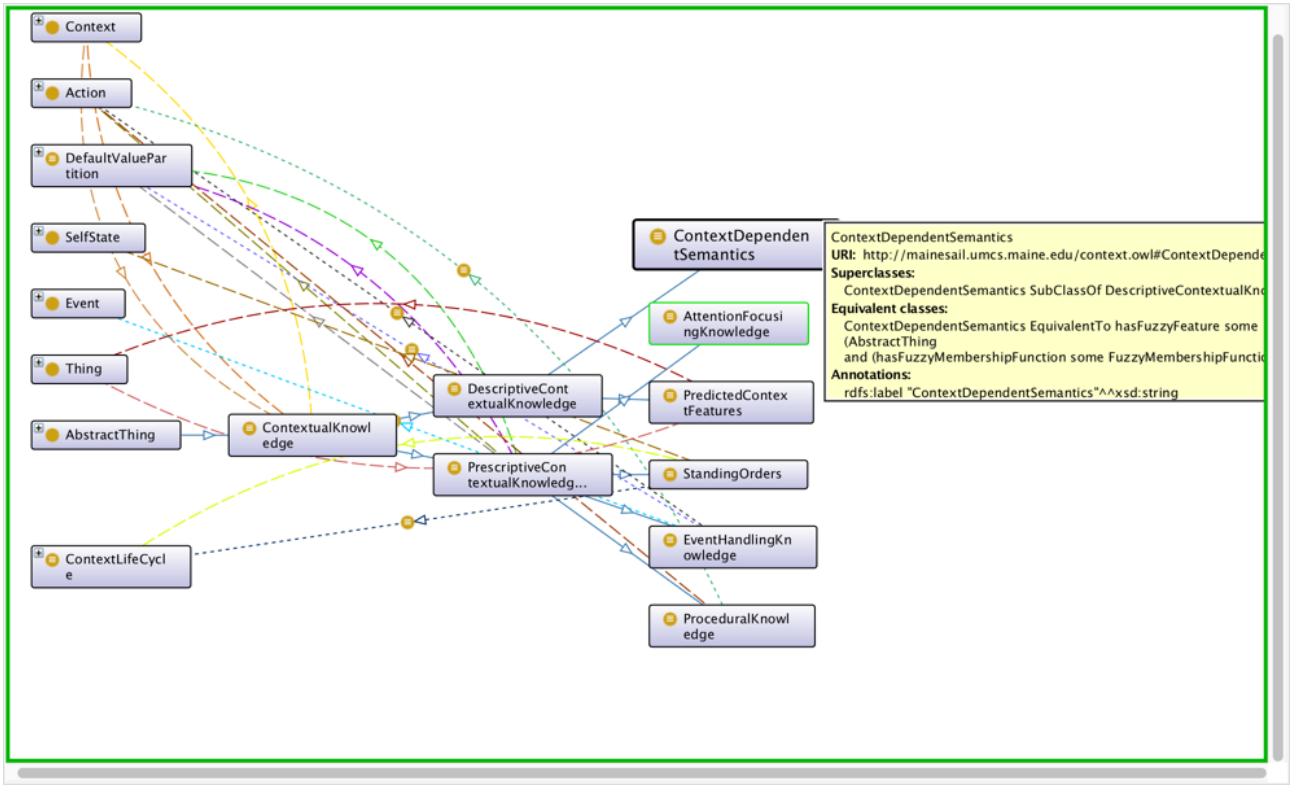

Figure 7. *Portion of an ontology for contexts and contextual knowledge.*

This work is still in progress and at an early stage. However, one useful outcome already has been an ontology for contextual knowledge (see Figure 7), represented in description logic, that can help facilitate and formalize distributed contextual reasoning [30]. In addition, mechanisms have been developed to allow agents to exchange information about even idiosyncratic contextual schemas without sending the entire schemas by making reference to their shared contextual knowledge [30] (cf. work on low-bandwidth communication by E. Turner [35] that shares some properties).

5.6. *Acquiring and learning c-schemas*

Work on CMB has been focused mostly on establishing how to represent and use contexts and contextual knowledge, with less attention being paid to acquiring and learning c-schemas. We believe that acquiring knowledge from humans will be facilitated by our approach, however. Humans seem to think in terms of contexts and cases, with such statements as “the last time I saw something like this” and “usually in cases like this” being common in, for example, medical problem solving. Consequently, it stands to reason that it will be easier to gather knowledge from humans when they can organize their thoughts around the contexts in which that knowledge is important. Thus, one could start by asking an expert to think about and list the kinds of situations that are important in his or her field of expertise; these would become c-schemas. Then, one could use those c-schemas to organize the acquisition of knowledge about how an artificial agent should behave in the corresponding contexts.

Although knowledge acquisition from humans is a first step, it is not the whole solution to acquiring an agent’s contextual knowledge. The agent needs to learn c-schemas for new contexts and tailor its existing c-schemas and their contextual knowledge based on its own experience.

Our work has generally relied on a memory for c-schemas based on a kind of dynamic, conceptual memory common in early case-based reasoners [19, 23]. This kind of memory addresses the learning problem by incorporating *similarity-based learning*, a form of inductive learning.

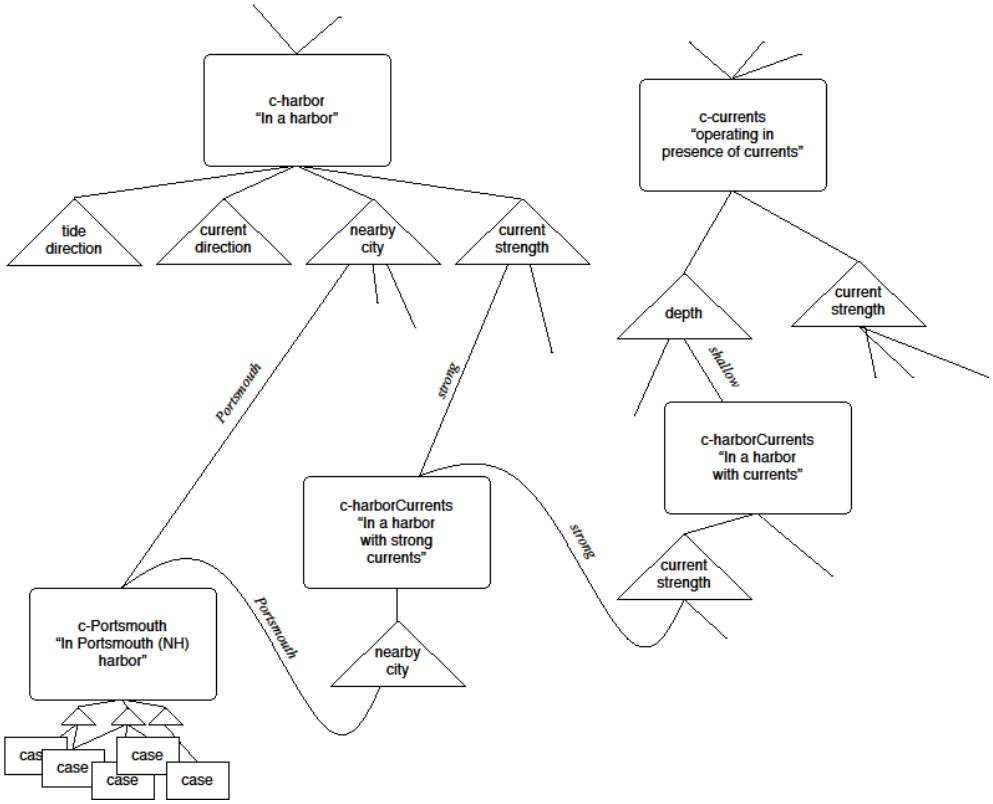

Figure 8. *A portion of a dynamic memory for c-schemas.*

A dynamic memory essentially organizes cases and schemas in multiple discrimination networks. Each schema is associated with a set of indices that link it to other, more specific schemas. Each index has two parts: the feature or features it relates to in the parent schema, and a value for that feature that occurs in the schema being indexed, which differs from the parent's value. For example, in Figure 8, the triangles represent the feature being indexed and the labeled arcs represent the values, so the c-schema representing operating in Portsmouth harbor can be found from the c-schema representing "in a harbor" by following the index "nearby city/Portsmouth", and the c-schema for "in a harbor with strong currents" can be found by following the index "current strength/strong" from one representing "in a harbor with strong currents".

The organization of a dynamic memory is idiosyncratic, since the memory itself learns the organization as cases or schemas are stored. When something is stored, the indices of the top-level c-schema are traversed using values of features in the new thing compared to the existing c-schema. Matching feature/value pairs lead to c-schemas or cases. If a c-schema, then the new thing is indexed under it in a recursive manner. If a case, then if the new thing is a schema, the case is replaced and the case is indexed under it; if the new thing is a case, then a collision has occurred, and similarity-based generalization occurs to create a new c-schema, which is then indexed there, with the cases indexed under it. If, however, no feature/value pair exists for an important feature when the new thing is being stored under a c-schema, then such a pair is created as a new index, and the thing is indexed there.[5] Thus as new episodes of problem solving (i.e., cases) are stored, new c-schemas may be created automatically by the memory.

[5]The process is much more complex than this, including index elaboration, generalizing indices as much as possible, setting thresholds numbers of cases involved in a collision before c-schema creation occurs, etc. For details, see [19] or [39]. In addition, we have augmented the basic dynamic memory to provide evoking strengths for the c-schemas found [23], which are then used during context assessment.

Recently, we have begun to look at augmenting this memory-based learning mechanism with more robust machine learning. For example, in the near future, we will be examining how neural network-based learning can be used both to create new c-schemas as well as to maintain the organization of the c-schema memory.

6. Current and future directions

In this section, we look briefly at the current work going on in our lab on context-mediated behavior and at future directions that are planned.

Completing an agent-agnostic context manager. We are currently working on creating a version of ConMan that can work with most agent reasoning styles with little or no modification. This problem requires understanding what parts of the context assessment problem are agent-independent and also determining how to effectively integrate agent-specific knowledge with the agent-independent portions of c-schemas. The result will be a context manager that can be with any agent.

Applying CMB to other kinds of agents. Related to the previous work, we are actively considering how to apply CMB to agents with reasoning styles that differ from schema-based reasoners. Of particular near-term interest is how to best apply CMB to neural networks, especially deep learning networks, to enhance them with context-sensitive behavior, as we discussed above. We will be looking at how to represent networks in c-schemas, not only for simple feedforward deep networks, but also for recurrent networks, convolutional networks, long short-term memory-based networks, etc. We are also very interested in applying CMB to neuroevolutionary approaches such as NEAT, so that network topologies as well as weights can be learned in a context-dependent manner. We also intend to look at applying CMB to learning classifier systems to enhance their ability to reason effectively in a wide range of contexts.

Continuing to formalize the knowledge representation for contextual knowledge. As mentioned, some work has been done on creating an ontology for the contextual knowledge contained in c-schemas, with the knowledge being represented in a description logic. Much work remains to be done, however, including incorporating the DL-based representation into ConMan and application c-schemas, as well as choosing and/or implementing a DL-based reasoner for ConMan's use.

Context merger. We have done some work on context merger, e.g., for numeric features and for features than can be generalized using an isa hierarchy or more formal ontology. The problem is far from solved, however. We will continue this work in the future, looking for general ways to merge different kinds of knowledge. We have the suspicion that the merger process itself is likely context-dependent, and so this ties into the need to consider "meta-contextual" knowledge discussed below.

MAS/distributed context assessment. Work continues on applying CMB to multiagent systems. This includes incorporating organizational knowledge into c-schemas for use in system self-organization and reorganization in the CoDA project. It also includes looking at how to distribute all parts of the CMB process across multiple agents, so that the MAS as a whole reasons about and is aware of the system-wide context. Although we have a rough roadmap for how to do this, as well as some work that has laid the foundation, much work remains to be done in the future.

Active vs. reactive CMB. In past work, we can think of CMB as being more reactive than active: context assessment happens in response to situational change. Recently, one of our students (C. Wilson) has begun work on a more active approach, in which an agent can reason about future contexts it can potentially be in as well as take actions to establish a desired context [51, 51, 52]. This has significant potential benefits for an agent, including helping it determine what to do to retain important features of the current context and how to cleanly exit a context (e.g., turning off the stove when transitioning from the "cooking" context to the "getting the mail" context). It also offers the possibility

for the agent to plan to be in a particular desired context, or not to be in undesired ones, based on overall properties of the contexts rather than simple goal achievement. This potentially can offer a solution to the pesky problem of dealing with so-called "preservation goals", such as maintain health, etc.: the agent, instead of planning to preserve such a goal, can instead plan to transition only to contexts in which the goal is still satisfied.

Learning contexts and contextual knowledge. As noted, relatively little has been done on this issue, though it is very important in the long run. We have begun thinking about ways that modern machine learning techniques, such as deep learning, can be incorporated into CMB to help it learn new contexts and to learn about its existing ones. For example, a long short-term memory network could be used to help detect and learn the consequences of trends in features of a context over time, or a feedforward network could be used to learn the applicability of a c-schema for a particular situation. This dovetails with our work on supporting alternate reasoning styles, since the topology and weights of a network that, say, predicts the usefulness of a c-schema could be stored in the c-schema itself and itself be subject to learning. Machine learning could also be used in the c-schema memory, both to augment or replace current similarity-based learning and to learn and maintain the memory's organization. We are at a very early stage of this work, but we are excited by the tremendous advances being made in machine learning currently, and we are optimistic about their usefulness for CMB.

Fielding applications. Most of the work done on CMB has been done in simulation, since we have been more focused on elucidating the mechanisms and knowledge representation needed than on applications. However, we anticipate applying CMB in a range of applications, some in the near future. We have mobile robots in our laboratory that we can field our agents on, and we have begun discussions with other researchers at our institution about using their instrumented underwater and surface vehicles, both as testbeds for CMB as well as a means to control these assets while collecting important oceanographic data. Since there are multiple vehicles, we may also be able to use them to test our multiagent systems work.

We also intend to continue collaborating with our former student (L. Whitsel) who is interested in trust and cybersecurity issues. This collaboration is particularly interesting, as his ideas on context representation and use are somewhat different than "traditional" CMB, and so should be a rich source of comparisons and discussion.

Fielding real agents, especially in support of other goals, will also allow us to address our goal of validating the context-mediated behavior approach by gathering data on how it performs in a variety of domains and applications.

Simulation and modeling applications. A recent project focuses on simulating the behavior of cognitively-impaired humans using CMB to represent their thought processes, based on the hypothesis that some inappropriate actions may be the result of errors either in assessing their context or in applying their contextual knowledge [52]. Although fascinating in its own right, the motivation for this project is to provide realistic subjects with which to test assistive agents for the cognitively-impaired. This work also has application to video games, since it could provide a means to implement non-player characters (NPCs) that behave more realistically in the "fog of war" or when being cognitively overloaded in general. Applications exist for so-called "serious games" as well, such as those used in battlefield simulation.

This work also ties into our long-term goal of developing a "cognitive prosthesis" to assist cognitively-impaired individuals, for example, those suffering from dementia. We anticipate using context-mediated behavior to control the agent, but we can also make use of this work to allow the agent to better understand its patient and to predict what he or she may do in a given situation.

Integration with other approaches to context management. CMB is not the only approach to context-sensitive reasoning, of course, and we would like to investigate not only how our approach

compares to others, but also how we might combine the best features of ours and others' approaches. Two very promising approaches to context-sensitive reasoning in particular are Brézillon's contextual graphs (e.g., [4]) and Gonzalez's CxBR [16]. Some work has been done by those authors to compare their work [15], and we intend to investigate how our work compares as well. We also will look for ways to incorporate some of their ideas, where useful, in our work and to point out where parts of CMB might benefit their work.

Meta-contextual knowledge. As we mentioned early in the paper, the first schema-based reasoner, MEDIC, not only had contextual and procedural schemas, but also a third type, *strategic schemas*. These had less to do prescribing what to do as how to do it, and they were used in concert with c-schemas. It has become clear to us over the years that these were really a representation of a kind of "meta-context", in particular representing the context of applying contextual information contained in c-schemas. Although s-schemas were dropped from CMB early on, we plan to investigate whether bringing them, or something very much like them, back will benefit CMB. We have alluded to one area in which some meta-knowledge would be beneficial: merging contextual knowledge from different c-schemas, which we said may itself be context-dependent. We can also see that some c-schemas that we have considered, such as one representing "clandestine operation" have much of the character of both c-schemas and s-schemas, since they represent both particular things to do/not do (e.g., standing orders for radio/acoustic silence) as well as things that cut across and are orthogonal to other c-schemas (e.g., causing an agent to ignore the need for GPS fixes, decreasing the importance of survival goals if preserving them would risk capture, etc.). We can even envision meta-c-schemas representing such things as "learning context" or "performance context" that would affect how the agent makes use of and changes its other knowledge, including c-schemas.

7. Conclusion

In this article, we have attempted to provide the reader with a review of the more important parts of one approach to modeling and using context, context-mediated behavior. Since work on this has been done over almost 30 years and reported in 50+ publications, we cannot hope to have reviewed everything. However, we hope we have given a reasonable overview of the approach and in particular the way we have addressed the principles set forth in an earlier section.

Although we have done much research on CMB, there remains much to be done, both by us and in cooperation with our colleagues in the context community. We look forward to many more years of fruitful research on the intriguing and important problem of how to create an agent that always behaves appropriately for its context.

8. Acknowledgments

The author would like to thank the many students and colleagues associated with his MaineSAIL laboratory who worked over the years on various projects referred to in this paper, especially Erik Albert, Robert Arritt, Richard Blidberg, Steve Chappell, David Gagnon, James Lawton, Prabha Ramakrishnan, Sonia Rode, Larry Whitsel, and Chris Wilson. A particular debt of gratitude is owed the late Prof. Elise H. Turner, with whom the author collaborated so productively for so many years on contextual reasoning and a host of other projects; she is very much missed, both professionally and personally.

References

[1] Aikins, J. S. *Prototypes and Production Rules: A Knowledge Representation for Computer Consultations*. PhD thesis, Stanford University, 1980.

[2] Aikins, J. S. A representation scheme using both frames and rules. *In Rule-Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project*. Addison– Wesley Publishing Company, Reading, Massachusetts, 1984, pp. 424–440.

[3] Arritt, R. P., and Turner, R. M. Context-specific weights for a neural network. *In Proceedings of the Fourth International and Interdisciplinary Conference on Modeling and Using Context (CONTEXT'03)* (Stanford, CA, 2003), Springer, New York, pp. 29–39.

[4] Brézillon, P. Representation of procedures and practices in contextual graphs. *The Knowledge Engineering Review 18*, 02 (2003), 147–174.

[5] Brézillon, P., and Cavalcanti, M. Modeling and using context: Report on the first international and interdisciplinary conference context-97. *The Knowledge Engineering Review 12*, 4 (1997), 1–10.

[6] Buvač, S., and Mason, I. A. Propositional logic of context. In *AAAI* (1993), pp. 412–419.

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

[8] Decker, K. S., and Lesser, V. R. Generalizing the Partial Global Planning algorithm. *International Journal on Intelligent Cooperative Information Systems 1*, 2 (1992), 312–346.

[9] Dey, A. K., Abowd, G. D., and Salber, D. A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. *Human-computer interaction 16*, 2 (2001), 97–166.

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

[11] Forgy, C. L. Rete: A fast algorithm for the many pattern/many object pattern match problem. *Artificial intelligence 19*, 1 (1982), 17–37.

[12] Fox, M. S., Sadeh, N., and Baykan, C. Constrained heuristic search. In *Proceedings of the 11th international joint conference on Artificial intelligence - Volume 1* (San Francisco, CA, USA, 1989), Morgan Kaufmann Publishers Inc., pp. 309–315.

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

[14] Gomez, F., and Chandrasekaran, B. Knowledge organization and distribution for medical diagnosis. In *Readings in Medical Artificial Intelligence*, W. J. Clancey and E. H. Shortliffe, Eds. Addison–Wesley Publishing Company, Reading, Massachusetts, 1982, pp. 320–338. (Originally published in IEEE *Transactions on Systems, Man, and Cybernetics*, vol. SMC–11, no. 1, pp. 34–42, 1982.).

[15] Gonzalez, A. J., and Brézillon, P. Comparing two context-driven approaches for representation of human tactical behavior. *Knowledge Engineering Review 23*, 3 (2008), 295.

[16] Gonzalez, A. J., Stensrud, B. S., and Barrett, G. Formalizing context-based reasoning: A modeling paradigm for representing tactical human behavior. *International Journal of Intelligent Systems 23*, 7 (2008), 822–847.

[17] Guha, R. V. *Contexts: a formalization and some applications*, vol. 101. Stanford University Stanford, CA, 1991.

[18] Hochreiter, S., and Schmidhuber, J. Long short-term memory. *Neural computation 9*, 8 (1997), 1735–1780.

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

[20] Kolodner, J. L. *Case-Based Reasoning*. Morgan Kaufman, San Mateo, CA, 1993.

[21] Kolodner, J. L., and Simpson, R. The mediator: Analysis of an early case-based reasoner. *Cognitive Science 13* (1989), 507–550.

[22] Kolodner, J. L., Simpson, R. L., and Sycara-Cyranski, K. A process model of case-based reasoning in problem-solving. In *Proceedings of the International Joint Conference on Artificial Intelligence* (Los Angeles, California, 1985).

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

[24] Lenat, D., and Guha, R. V. Cyc: A midterm report. *AI magazine 11*, 3 (1990), 32.

[25] McCarthy, J. Generality in artificial intelligence. *Communications of the ACM 30*, 12 (1987), 1030–1035.

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

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

[28] Padovitz, A., Loke, S. W., and Zaslavsky, A. Multiple-agent perspectives in reasoning about situations for context-aware pervasive computing systems. *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and HumansIEEE Transactions on 38*, 4 (2008), 729–742.

[29] Reggia, J. A. Abductive inference. In *IEEE Proceedings of the Expert Systems in Government Symposium* (Washington, DC, 1985), IEEE Computer Society Press.

[30] Rode, S., and Turner, R. M. Representing and communicating context in multiagent systems. In *Modeling and Using Context (Proceedings of the Ninth International and Inter-disciplinary Conference on Modeling and Using Context)* (Larnaca, Cyrpus, November 2015), H. Christiansen, I. Stojanovic, and G. A. Papadopoulos, Eds., Lecture Notes in Artificial Intelligence, Springer.

[31] Schank, R. C. *Dynamic Memory*. Cambridge University Press, New York, 1982.

[32] Sowa, J. F. Building large knowledge-based systems: Representation and inference in the cyc project: Db lenat and rv guha, 1993.

[33] Stanley, K. O., and Miikkulainen, R. Evolving neural networks through augmenting topologies. *Evolutionary computation 10*, 2 (2002), 99–127.

[34] Sycara, K. Arguments of persuasion in labour mediation. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI-85) (Los Angeles, 1995)*, pp. 294–296.

[35] Turner, E. H., Chappell, S. G., Valcourt, S. A., and Dempsey, M. J. Cola: A language to support communication between multiple cooperating vehicles. In *Proceedings of the 1994 Symposium on Autonomous Underwater Vehicle Technology AUV'94* (1994), IEEE, pp. 309–316.

[36] Turner, R. M. *Organizing and using schematic knowledge for medical diagnosis. In Proceedings of the DARPA Case-Based Reasoning Workshop* (Clearwater Beach, Florida, 1988), pp. 436–446.

[37] Turner, R. M. Using schemata for diagnosis. In *Proceedings of the Twelfth Annual Symposium on Computer Applications in Medical Care* (Washington, DC, 1988).

[38] Turner, R. M. Case-based and schema-based reasoning for problem solving. In *Proceedings of the DARPA Case-Based Reasoning* Workshop (Pennsacola,, 1989).

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

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

[41] Turner, R. M. 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).

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

[43] Turner, R. M. Context-mediated behavior. In *Context in Computing: A Cross-Disciplinary Approach for Modeling the Real World Through Contextua Reasoning*, P. Brézillon and A. Gonzalez, Eds. Springer, 2014, ch. 32, pp. 523–540.

[44] Turner, R. M., Fox, J. S., Turner, E. H., and Blidberg, D. R. Multiple autonomous vehicle imaging system (MAVIS). In *Proceedings of the 7th International Symposium on Unmanned Untethered Submersible Technology (AUV '91)* (Durham, NH, 1991), pp. 526–536.

[45] Turner, R. M., Rode, S., and Gagne, D. Distributed, context-based organization and reorganization of multi-AUV systems. *Journal of Unmanned System Technology (JUST) 2*, 1 (2013), 1–9.

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

[47] Turner, R. M., and Turner, E. H. A two-level, protocol-based approach to controlling autonomous oceanographic sampling networks. *IEEE Journal of Oceanic Engineering 26*, 4 (October 2001).

[48] Whitsel, L., and Turner, R. M. A context-based approach to detecting miscreant behavior and collusion in open multiagent systems. In *Proceedings of the Seventh International and Interdisciplinary Conference on Modeling and Using Context (CONTEXT'11)* (Karlruhe, Germany, September 2011).

[49] Whitsel, L., and Turner, R. M. Using contextual knowledge for trust strategy selection. In *Modeling and Using Context (Proceedings of the Ninth International and Interdisciplinary Conference on Modeling and Using Context)* (Larnaca, Cyrpus, November 2015), H. Christiansen, I. Stojanovic, and G. A. Papadopoulos, Eds., Lecture Notes in Artificial Intelligence, Springer.

[50] Whitsel, L. T. A *Context-Based Approach to Detecting Miscreant Agent Behavior in Open Multagent Systems*. PhD thesis, School of Computing and Information Science, University of Maine, University of Maine, Orono, ME 04469 USA, December 2013.

[51] Wilson, C., and Turner, R. M. Modeling erroneous human behavior: A context-driven approach. In *Modeling and Using Context (Proceedings of the Ninth International and Inter-disciplinary Conference on Modeling and Using Context)* (Larnaca, Cyrpus, November 2015), H. Christiansen, I. Stojanovic, and G. A. Papadopoulos, Eds., Lecture Notes in Artificial Intelligence, Springer.

[52] Wilson, C., and Turner, R. M. Context and the virtual human. In *Modeling and Using Context (Proceedings of the Tenth International and Interdisciplinary Conference on Modeling and Using Context)* (Paris, France, June to appear), P. Brézillon, R. M. Turner, and C. Penco, Eds., Lecture Notes in Artificial Intelligence, Springer.

[53] Wilson, J., Hill, J., Kersula, M., Wilson, C., Whitsel, L., Yan, L., Acheson, J., Chen, Y., Cleaver, C., Congdon, C., Hayden, A., Hayes, P., Johnson, T., More-head, G., Steneck, R., Turner, R., Vadas, R., and Wilson, C. Costly information and the evolution of self-organization in a small, complex economy. *Journal of Economic Behavior and Organization 90* (June 2013), S76–S93.

