Context-Based Reasoning: A Revised Specification

Brian S. Stensrud, Gilbert C. Barrett, Viet C. Trinh, and Avelino J. Gonzalez

Intelligent Systems Laboratory
University of Central Florida
Orlando, Florida
{brian, gilbarrett, vtrinh, gonzalez}@isl.ucf.edu

Abstract

This paper is an extension to and revision of Gonzalez and Ahlers' [6] definition of the Context-Based Reasoning Paradigm. Included are rigorous definitions of all terms and components applicable to CxBR models along with a discussion on how and where these models store and execute tactical knowledge. In addition, new terms and concepts are introduced that justify the need for a revised specification and reflect the research done to and with CxBR over the past five years. Finally, this paper includes a description of a physically implemented CxBR model, with an emphasis on how the defined components play a role in its functionality.

Introduction

We re-introduce here the technique of Context-based Reasoning (CxBR) [6]. CxBR is a reasoning paradigm that allows for intelligent agents to be modeled for use in a variety of environments and scenarios where tactical expertise is necessary. After providing an overview of the paradigm and some related work in the field, each component of CxBR is introduced and defined in detail. An example CxBR model and interface is then given that ties together each component mentioned. Concluding the paper is a discussion on the intrinsic and extrinsic knowledge representation properties of the CxBR paradigm.

Motivation

The motivation for CxBR is the idea that people tend to use only a fraction of their knowledge at any one given time [6]. For instance, let us consider an auto mechanic on his way to work. While he needs to keep in mind rules of the road – following speed and caution signs, avoiding pedestrians and other obstacles, being mindful of the other drivers in the area – his knowledge of how to rebuild a car's transmission is irrelevant with respect to his need to maneuver the road. In creating a model for this mechanic's behavior while driving to work, the representation of his expertise in fixing cars can be omitted. On the other hand, such knowledge would be required for a CxBR representation of the mechanic's day-to-day activities. While driving, however, our mechanic will not likely need to tap his technical knowledge.

This idea lead to the concept of dividing the knowledge base into contexts is based on this idea. Given any behavior to model, contexts represent exclusive behavior classes relevant to that behavior. From that, the knowledge required to execute a specific behavior is confined to its representative context.

While this paradigm benefits from its apparent intuitiveness, there are other advantages that make CxBR a viable solution, especially within the realm of tactical behavior. First, de-composing a model's behavior space – or behavioral capabilities – into contexts enables the model to carry a very broad understanding of its task. While this understanding might, at times, be only on a general level, a context space representative of the entire domain in which the model is to operate, all but guarantees that it will operate on some level of intelligence at any point during its mission.

There are many times where a certain skill may be helpful in more than one situation. Furthermore, a certain behavior might be needed in a variety of tactical tasks. CxBR models, in this sense, are modular. Contexts, which may have been constructed for one specific task, can be extracted from its model and inserted into a model for a new task in which that context is relevant. Because of this feature, CxBR models greatly benefit from an object-oriented software engineering approach.

Overview of Context-Based Reasoning

CxBR is a reasoning paradigm from which autonomous agents can be modeled to execute a specifically defined task in either a simulated or real-world environment. The task assigned to the agent is encapsulated within a CxBR *mission*. This mission provides for the agent both a set of goals, which represent the criterion for completing the task, and a set of constraints specific to that task. Also present within a mission is a list of *contexts* that serve to partition the agent's task-related knowledge by the situations under which it applies.

A *context* represents a situation, based on environmental conditions and agent stimuli, which induces a certain agent behavior specific to that mission. When an agent is executing a mission within CxBR, its behavior is controlled by the current *active context*, a determination made by *context-transition logic*. At each time step, this transition logic examines the current stimuli on the agent and makes a determination of the active context for the subsequent time step. This logic is often in the form of *sentinel rules* that contain the conditions for a specific context-to-context transition; however the transition logic is not required to be rule-based.

Related Work

Context Mediated Behavior (CMB) [14] is based on early work done in Context-sensitive Reasoning [15]. Contexts are captured and represented as context schemas (*c-schemas*). These c-schemas contain information regarding: important features to cause the c-schema to be active, standing orders, events, goals, and actions. In addition to c-schemas, procedural schemas, p-schemas, are used to define actions for agents. A context manager (CM) is used to reason about which c-schemas should be active for a given context.

Many of the concepts involved with CMB are parallel to that of CxBR. One notable exception is the way that multiple c-schemas can be active or merged to create a new c-schema. Also different is that c-schemas are used in much the same manner as cases in Case Based Reasoning (CBR). This differs from CxBR in that Contexts in the implementation of CxBR provide a more active functionality.

There are many parallels between the design of SART's [2] contextual reasoning and that of CMB and CxBR. Brézillon describes three major areas of knowledge representation: external knowledge, contextual knowledge, and procedural knowledge [3]. This is not different than the knowledge represented by CMB or CxBR except as a matter of semantics. Procedural context knowledge is where an agent's actions are defined. Contextual knowledge includes what is analogous to transition criteria in CxBR and any knowledge necessary to reason about what procedural context should be used. Brézillon [3] explains, "Proceduralized context defines what the focus of attention is, and contextual knowledge defines the context of the focus of attention." External knowledge is implied to be the knowledge known outside of contextual knowledge and procedural knowledge, such as particular elements of events or characteristics regarding the environment.

CxBR Components

The following subsections define each component modeled within CxBR, its function, and its relevance within the paradigm.

Missions

A mission, or mission context, is an abstraction defined within the model and assigned to a specific agent prior to run-time. Included within a mission is the goal, any imposed constraints, and the context topology that will dictate the high-level behavior of the agent.

The goal provides the agent with the criterion for mission termination – end-game conditions for the agent's behavior. For example, consider the assignment of a mission *X* in which the criterion for completing *X* would be to satisfy conditions *a, b,* and *c*. Obviously, that goal can be represented formally using a Boolean function (e.g. goal_x = ($a \cap b \cap c$)) and embedded within a CxBR model to indicate whether or not the agent has satisfied the requirements of *X*. While this expression can certainly encompass more exotic end-game criteria for a mission than the example above, the variables relevant to computing this goal state can certainly be reduced to a set of stimuli on and the current state of the agent performing *X*. Because of this, the mission goal can be formally defined as a Boolean function *g* of a set of environmental and physical conditions **E** and **P** that exist at the time of query.

$$goal = g(\mathbf{E}(t_0), \mathbf{P}(t_0))$$

In tactical missions, it is often the case where a 'goal' cannot be defined or is not applicable. More specifically, it is not uncommon to assign an AIP with the mission of performing a certain task or behavior for an indefinite amount of time. In this case, the goal can be construed as an end-game condition for the simulation or scenario. If, for example, an agent representing a scout plane is assigned the mission of performing general reconnaissance on a particular area, the 'goal condition' might be defined as the point where the agent has either been shot down or is ordered to discontinue the mission and return to base.

The constraints on the mission provide the AIP with a set of guidelines for operation. These constraints can be in the form of physical limitations placed on the sensing faculties of the agent, maximum and minimum counts for scenario-specific entities such as obstacles or enemies, or even map boundaries within which the AIP is required to operate. We can consider the constraints on the mission *M* to be the union of the set of physical, environmental, and logistical constraints (denoted $\mathbf{T}_p$, $\mathbf{T}_e$, and $\mathbf{T}_l$) placed on the agent as required by its mission. In this definition, a constraint *c* provides the AIP with either a constant value or a range of valid values for a certain variable within the simulation.

$$constraints = \{ \mathbf{T}_p, \mathbf{T}_e, \mathbf{T}_l \}$$

While the notion of a context will be formally introduced in the following section, it is important to mention it here, as it is an essential part of the mission. It was mentioned

earlier that to model a behavior with CxBR, that behavior must have the quality that it can be partitioned into sections representing all possible situations; these sections in sum represent completely that behavior. The reason for this requirement is that the behavior or task, as represented by any CxBR model, must be defined completely by the contexts that constitute it. It is because of this that the mission is also responsible for listing the contexts that are required to correctly execute the model's behavior in that mission. A default context is also listed within the mission, which is a behavior that the model can execute when it is unsure of a behavior to use for a certain situation. This context is also used as the initial context for the agent when it begins a scenario unless a more applicable context can be selected.

The mission defines the high-level behavior of the agent by assigning it both a set of contexts and *context-transition pairs*, which indicate the specific context switches that will be allowed during the scenario. For example, consider the following two sets. The set $\mathbf{C_x}$ represents a set of five distinct major contexts present in a mission M_x, while set $\mathbf{T_x}$ includes all possible context-transition pairs applicable while executing M_x.

$$\mathbf{C_x} = \{c_1, c_2, c_3, c_4, c_5\}$$
$$\mathbf{T_x} = \{<c_1,c_4>,<c_2,c_3>,<c_3,c_1>,<c_4,c_2>,<c4,c5>,<c5,c1>\}$$

Since the context-transition pair $<c_1, c_4>$ is a member of $\mathbf{T_x}$, context c_4 is an applicable transition from context c_1. In other words, if the agent is currently operating in context c_1, it is possible to switch contexts at a given time-step t_0 to context c_4, if certain conditions exist at t_0. The logic used to trigger these pairs is known as *context-transition logic*, and will be defined in the next section.

A CxBR model's *context topology* CT_x consist of a set of contexts $\mathbf{C_x}$, along with the set of context-transition pairs $\mathbf{T_x}$, the Default Context (c_{DX}), and the scenario's *universal transition criteria* $\mathbf{UTC_x}$. CT_x, along with the goal conditions and constraints, comprises mission M_x.

$$CT_x = <\mathbf{C_x}, \mathbf{T_x}, c_{DX}, \mathbf{UTC_x}>$$
$$M_x = <goal_x, constraints_x, CT_x>$$

Contexts

A *context* is a set of environmental and physical conditions that may suggest a specific behavior or action [6]. Within a CxBR model, however, a context is a functional state induced as a result of these conditions. Contexts are inserted within a mission to represent all possible conditions that can arise during the course of that mission. This ensures that a model can exhibit intelligent behavior no matter what occurs during mission execution.

CxBR models are constructed such that a single context is *active* at any one point during a scenario. It is said that a context within the model is 'active' if the conditions implying its validity exist and the agent is using its included knowledge to make decisions within a scenario. That context is then denoted the current *active context*.

The knowledge engineer responsible for creating the model is in charge of defining and creating each context. Because of this, contexts themselves are often intuitive subsets of the behavior to be modeled. When encoding the knowledge for these contexts, the idea is to achieve a model that can take the same actions that an expert might take when in the same situation. Consider a mission M with context set $\mathbf{C} = \{c_1, c_2 \ldots c_n\}$. While the division of knowledge represented by these contexts is in the extreme case arbitrary, the knowledge engineer responsible for constructing the model will likely partition each context in a manner consistent with his understanding of the mission. Furthermore, the context-space might also be partitioned so that each context is coupled with a specific task or behavior that is necessary for the mission. This technique is often used for tactical models in which the sequence of activities and behavior is well known and bounded, and also where the mission itself entails the execution of a series of sub-tasks. It is important to note here, however, that the context-space must partitioned in order to represent all possible *situations* that may exist for the agent during a scenario – *not* simply to divide all possible actions that the agent might take. This is to ensure that the behavior space of the agent is completely spanned by the set of contexts and no situation within a scenario leaves the agent without an appropriate contextual response.

Within a CxBR model, individual contexts are nothing more than conduits between the current set of stimuli facing the agent and the behavior that will be executed in response. When a CxBR context is declared active, it references the appropriate behavior modules and fact-bases, which in turn provide it with the correct course of action. The command for that action is then passed from the context to the agent's interface for execution. The context will continue to repeat these steps until a different context is denoted as active.

An active context controls the agent by referencing various knowledge and action modules. These modules are not restricted to a specific form – inference engines, neural networks, and expert systems are all valid modules. Using these modules along with a local fact base present within the agent interface, the active context derives an appropriate action. Restated, the context logic for a context is composed of the control functions, knowledge and action rules that constitute the AIP's 'behavior' within that context. We define $\mathbf{F_{MC}}$ as the set of functions that control the AIP under a specific active context, such that

$$\mathbf{CF_{MC}} = \{cf_1, cf_2, c_3, \ldots, cf_n\}$$

Furthermore, we define the set of *action rules* for a specific context as $\mathbf{AR_{MC}}$. Action rules are general purpose

productions used for among other things, Sub-Context activation. They can use facts located in the agent's local fact base, or local variables in the functions that form part of $\mathbf{F_{MC}}$. Some implementations of CxBR may additionally contain a global fact base upon which facts accessible to all models may reside. Action rules may also use facts on the global fact base as antecedents. Thus, we can define $\mathbf{AR_{MC}}$ as:

$$\mathbf{AR_{MC}} = \{ar_1, ar_2, ar_3, ar_4 \ldots ar_k\}$$

Lastly, we define the knowledge contained by the Major Context as a set of frames or classes whose attributes and methods/daemons are essential elements of the tactical knowledge required to successfully navigate the current situation. We refer to this knowledge, for lack of a better name, as *Knowledge Frames* or $\mathbf{KF_{MC}}$.

Therefore, the Context-logic which controls the actions of the AIP while under the control of a Major Context is formally defined as:

$$\text{Context-logic} = < \mathbf{CF_{MC}}, \mathbf{AR_{MC}}, \mathbf{KF_{MC}} >$$

Sub-Contexts. CxBR supports the use of context-like structures, known as *Sub-Contexts*, which encompass a small functional section of a context not directly critical to the mission objectives. These structures share logical similarities to contexts, but lack many of their attributes. A Sub-Context is called upon, like a function, to perform a subtask deemed necessary in the logic by a context. Unlike contexts, however, one Sub-Context does not need to be active at any given moment. Furthermore, when a Sub-Context has finished executing, it is immediately deactivated and control shifts back to the Major Context that called it. In terms of its role, it is more convenient to think of Sub-Contexts as user-defined functions that are slightly more complex and specific to the model's mission. However, unlike user-defined functions - whose scope is typically the context that uses it – Sub-Contexts can be used by any context present within the model. This enhances re-usability of components in the model. Nevertheless, we can represent the Sub-Context by a vector function - whose input is an *action rule* of the calling context.

$$\text{theSubContext}_0 = f_0(\mathbf{AR_{MC}}_i)$$

Context Moderators. A *context moderator* is an abstract operator that can be applied as a parameter to either a CxBR model's context transition logic or to the functions and actions called upon by the active context selected. As a result, these moderators have the ability to either affect decision-making after an active context has been selected (*functional moderators*) or to affect the context-transition logic itself (*context-transition moderators*). These moderators are often embodiments of stimuli not directly associated with the assigned mission but nevertheless have an affect on an agent's behavior. An example of such moderators are human moods and emotions, which were integrated into a CxBR model as both functional and context-transition moderators in [10].

Context-Transition Logic

The selection of an active context during a scenario is controlled by the context-transition logic. Knowing the active context and the recent stimuli on the AIP, the context-transition logic selects the appropriate context transition amongst the pairs listed by the mission.

Context-transition logic is permitted to take any form within a CxBR model, so long as a context is chosen at each time step. The most popular representation of context-transition logic is through the use of sentinel rules and universal sentinel rules.

Sentinel Rules. With this implementation, the knowledge containing conditions under which a context transition is required are called *sentinel rules*, or *transition sentinel rules*. Sentinel rules indicate when the appropriate conditions for each applicable transition (each context-transition pair provided by the Mission) hold true. If, for instance, the mission provides a context-transition pair for context c_1 to c_3, a sentinel rule will be present within c_1 that monitors for the conditions warranting a transition from c_1 to c_3. If that condition arises, the transition sentinel rule corresponding to that pair will fire, and a transition will be instantiated.

Sentinel rule antecedents may include the fact-base of the current context and the current status of the agent (e.g. inputs, physical state and location). While often there are fixed conditions for transitioning to a given context, sentinel rules are unique to the context where they exist. This feature allows the agent to function in more complex tactical domains where transitions to a context might be a consequence of two entirely different motivations.

When sentinel rules are implemented within a mission M_x, the CxBR model provides a set $\mathbf{S_x}$ of transition criteria that represent the conditions necessary for each transition listed in $\mathbf{T_x}$ (the set of legal context-transitions). Representing the rule defining the transition criteria from context c_i to c_j as s_{ij}, we can define the set of sentinel rules $\mathbf{S}_x$ as the combination of all s_{ij} where $<i, j>$ is a member of $\mathbf{T_x}$ (i.e. if $<i,j>$ is a valid transition within mission M_x).

$$\mathbf{S}_x = \bigcup_{i,j=1}^{i,j \ni <i,j> \in M_x} \mathbf{S}_{ij}$$

In many tactical scenarios, there exist conditions that require the agent to perform a certain task or behavior

without considering its current context. To account for such conditions, *universal sentinel criteria* are encoded within the mission. These criteria dictate whether the agent should shift to a certain context regardless of its current active context, and hold precedence over all other transition criteria. When sentinel rules are used to represent a model's transition logic, universal sentinel criteria are encoded by a set of *universal sentinel rules*. Universal sentinel rules define conditions under which the model must transition to a certain context *irrespective* of the current active context.

$$\mathbf{US}_j = \bigcup_j \mathrm{usr}_{xj}$$

Transitional Fuzzy-ART Templates. While sentinel rules are the most popular form of context-transition logic within CxBR models, there are other ways to achieve a similar functionality. Stensrud [8] describes a learning algorithm by which context-transition logic can be constructed automatically by observing an expert's behavior in a simulation. More specifically, the actions and stimuli of an expert are fed into a Fuzzy ARTMAP neural network in order to create a mapping between the stimuli on the agent and the specific high-level decision made by the expert in response. CxBR contexts are created *a priori* and are designed to correspond to a set of actions that may be necessary to complete a task. The decision of the expert to change his action is identified as a possible context transition and fed into the neural network along with the inputs that prompted the expert's decision. After the completion of the observational phase, the set of Fuzzy ARTMAP templates created within the neural network represent the context-transition logic used by the expert. That logic is then plugged-in to the CxBR model (a model designed to imitate the observed expert) along with the other pre-existing contexts. When this model is executed, context transitions can be determined simply by supplying applicable inputs to the neural network.

Agent Interface

CxBR models represent low-level functional intelligence through an agent interface that serves as a medium between the model and the physical or simulated agent. This low-level intelligence represents the physical capabilities of the agent it models – moving, turning, stopping, firing, etc. While these functions are used to carry out the assigned mission, they are not considered tactical intelligence and are therefore stored externally to the model.

When an autonomous agent is executing a scenario, its controlling CxBR model is continuously determining an appropriate course of action based on its current active context and relative stimuli. The actions selected by the model are represented in terms of the low-level commands represented in the interface.

Also present within the agent interface is the raw data representative of the states of both the agent and the surrounding environment. For example, for a mission defined to drive a vehicle to work, data representing the agent's speed, location, and distance to work would be stored in the interface module as well as any information on the current traffic environment, nearby pedestrians, and the like.

A CxBR Model. Figure 1 is a block diagram of a generic CxBR model that can be generated using the current framework created by Norlander [11]. This framework serves as both an engine for CxBR models as well as a foundation on which they are constructed.

The agent interface module stores any sensor data that is read-in by the agent, and includes any necessary low-level functionality needed to implement the actions indicated by a context. When a model is run, this module is instantiated and assigned a mission. The CxBR model controls the agent by calling for actions in terms of the functions defined within this interface.

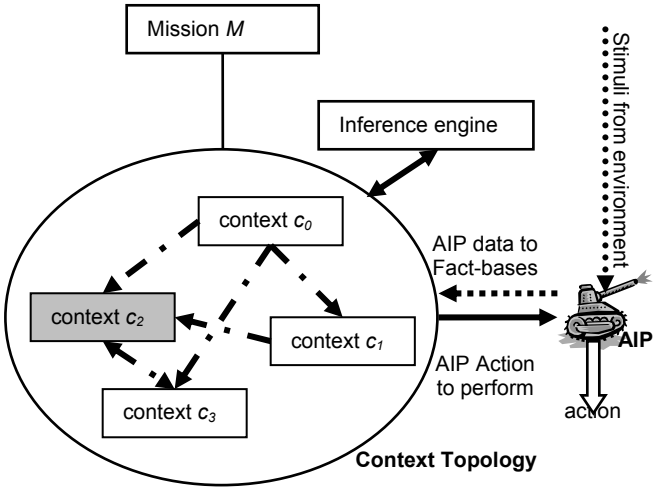

Figure 1 - Block diagram of a CxBR model

As illustrated, CxBR missions define a context topology for the model as well as valid context-transition pairs (illustrated by the dashed lines); agent constraints, universal sentinel rules, and mission objectives (goals). They are also responsible for identifying the Default context, which is the context that the agent will operate in at the start of the scenario. If no sentinel rules fire within the current context and it is also found that the current context is not valid, the model will revert to this default context.

As an example of a CxBR model, we present the iRobot Scenario developed in [14]. This scenario was an exercise in implementing a CxBR model on a physical platform. In this scenario, the mission is to maneuver an iRobot around

an open area looking for a single enemy entity. Upon detection, determine the hostility level of the enemy. If it approaches, consider it hostile and retreat. If the enemy retreats, follow it at a close distance. If the enemy is not responsive (i.e., stationary), execute an end of mission signal and retreat to the original starting position. The context topology for this scenario is provided as Figure 2. The agent interface connects the CxBR model to the iRobot and defines its low-level functions (move, turn, activate sonar).

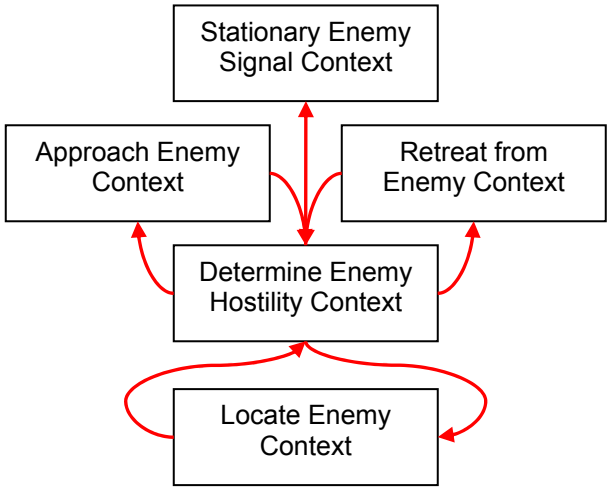

Figure 2 – Context Topology for iRobot Scenario

Knowledge Representation in CxBR

As alluded-to in the previous sections, the CxBR paradigm itself provides a way of representing knowledge through the use of the agent, mission, context, and context moderator objects.

At some level, knowledge is contained in all CxBR components. Some of this contained knowledge is directly responsible for the action of the agent, such as the high-level behavioral knowledge represented within Contexts. Other knowledge contained in these CxBR Objects is concerned with the dynamics of the paradigm itself, such as the context topology contained in the Mission Object. Regardless of whether the knowledge is used for directly controlling the agent or the dynamics of the paradigm, CxBR does not constrain nor specify the use of any particular type of knowledge representation paradigm.

The importance of not demanding a specific knowledge-representation paradigm is in the flexibility offered to the modeler. Any knowledge or associated reasoning mechanisms employed must be determined by the knowledge engineer responsible for model construction. For simple systems, a rule-based structure may prove to be the most efficient. However, if learning is to be incorporated or the details of decision-making are not easily classified in terms of rules, a structure such as a neural network may be employed. Both of these paradigms have been successfully integrated with CxBR models in the past (see [7], [11], [12]). Again, the CxBR paradigm itself does not limit the type or types of knowledge representation used; rather it is a decision to be made based on the requirements of the model being constructed.

Agent's Extrinsic Knowledge

Each agent is aware of its current Mission at any given time. Missions, contexts, and context moderators are objects in CxBR that support the autonomous behavior of an agent. Their interrelated nature and interdependencies are depicted in Figure 1. As a brief oversimplification, a mission is composed of a set of contexts, which themselves can be modified by one or more sets of context moderators.

As shown in Figure 3, a Mission contains the following knowledge: the agent's high-level goal, mission constraints, and the context topology. Contexts contain high-level behavior representation, sub-goals, context transition topology, context transition criteria, and context moderator affects. Context moderators are new to CxBR and provide an optional way of expanding the richness of agent behavior through influencing context transition logic and intra-context behavior.

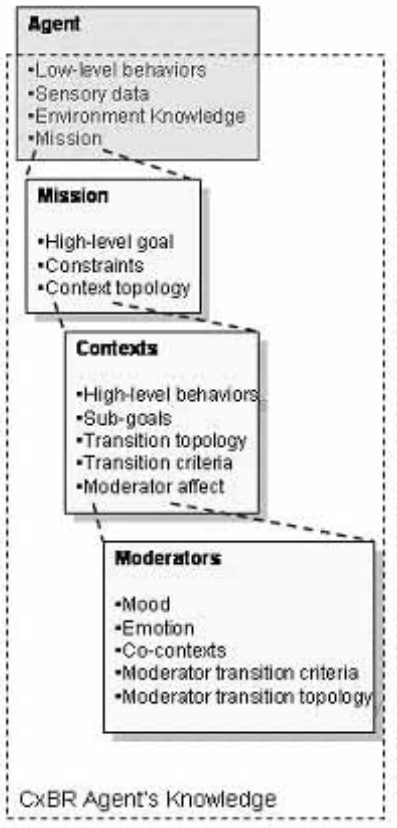

Figure 3 – Knowledge Schema for a CxBR Agent

Agent's Intrinsic Knowledge

We define low-level behaviors as behaviors closely related to dynamic physical and behavioral characteristic of the agent. Such behaviors may include motor skills, sensory data, what the agent perceives about its world, environmental knowledge, or even what the agent remembers with regard to its historical perception of the world These low-level behaviors are fundamental in defining the agent. This is true in the sense that the agent is defined by the low-level behaviors of which it is capable and, also, in the sense that the constraints of the behaviors themselves further define the agent. Consider a behavior such as movement and a corresponding function move() to represent this behavior. Different agent types should be characterized in distinctly different ways by how move() defines them. For example, move() to a helicopter allows for three dimensional movement through space, but there are certain constraints that must be adhered to regarding maximum velocity, maximum altitude, attitude of the aircraft, etc. A fish would also have a low-level behavior defined by move(). However, the maximum velocity or maximum altitude of a fish will obviously differ from that of a helicopter.

In addition to low-level behaviors, in CxBR each agent has some perception of and knowledge about its surrounding world. What is of particular importance here, as in the other areas of knowledge representation employed by CxBR agents, is the flexibility the modeler is permitted in choosing knowledge representation paradigms. The method in which memory is implemented for a model is not constrained by the CxBR paradigm. A set of data structures stored in memory could be used to allow fast retrieval of information. Alternatively, a database could be interfaced with the model to allow storage and retrieval of large quantities of data.

Conclusion

Context-Based Reasoning is an intuitive and effective means by which to model tactical behavior in either simulated or real-life scenarios. Introduced in this paper is a formal definition of the CxBR paradigm as it applies to creating CxBR models for use on autonomous agents. Both the paradigm and the current modeling framework have proven to be compatible with expansion and change, which will allow CxBR to remain a viable and effective human-behavioral modeling paradigm throughout the years to come.

References

[1] Barrett, G.B. and Gonzalez, A.J., "Modeling Collaborative Behaviors in Context-Based Reasoning," *Proceedings of the Swedish-American Workshop on Modeling and Simulation (SAWMAS)*, 2002.

[2] Brézillon, P., "Context in Artificial Intelligence", *Computer and Artificial Intelligence*, Vol 18, Number 4, 1999, pp. 321-340.

[3] Brézillon, P., "Modeling and Using context in Applications", *International Journal on Human-Computer Studies*, vol. 48 (3), 1998.

[4] Fernlund, H.K. and Gonzalez, A.J., "An Approach towards Building Human Behavior Models Automatically by Observation," *Technical Report*, Intelligent Systems Laboratory, Orlando, FL, 2002.

[5] Gonzalez, A. J., and Dankel, D. D., The Engineering of Knowledge-Based Systems: Theory and Practice, Englewood Cliffs, N. J.: Prentice Hall, 1993.

[6] Gonzalez, A. J. and Ahlers, R. H., "Context-Based Representation of Intelligent Behavior in Training Simulations", *Naval Air Warfare Center Training Systems Division Conference*, 1998.

[7] Henninger, A.E., "Neural Network Based Movement Models to Improve the Predictive Utility of Entity State Synchronization Methods for Distributed Simulations", Doctoral Dissertation, University of Central Florida, Orlando, FL, 2001.

[8] Norlander, Lars, "A Framework for Efficient Implementation of Context-Based Reasoning in Intelligent Simulation", Master's Thesis, ECE Dept., University of Central Florida, Orlando, FL, 1999.

[9] Saeki, S. and Gonzalez, A. J., "Soft-Coding the Transitions Between Contexts in CGF's: The Competing Context Concept", *Proceedings of Computer Generated Forces and Behavior Representation Conference*, 2000.

[10] Stensrud, B.S., Barrett, G.B., Lisetti, C.L. and Gonzalez, A.J., "Modeling Affect in Context-Based Reasoning", *Proceedings of the Swedish-American Workshop on Modeling and Simulation (SAWMAS)*, 2002.

[11] Stensrud, B.S., "An Algorithm for Learning Context-Transition Logic from Observation," *Technical Report*, Intelligent Systems Laboratory, Orlando, FL, 2003.

[12] Stensrud, B.S., Barrett, G.B., and Gerber, M., "A Functional Comparison of Context-Based Reasoning and Native Behaviors Within OneSaf Test Bed," *Technical Report*, Intelligent Systems Laboratory, Orlando, FL, 2003.

[13] Turner, R.M., Context-Mediated Behavior for Intelligent Agents, *International Journal of Human-Computer Studies: "Using context in Applications"*, Volume 48, Number 3, March, 1998, pp. 307-330.

[14] Turner, R.M., "A Model of Explicit Context Representation and Use for Intelligent Agents", *Proceedings of the Second International and Interdisciplinary Conference on Artificial Intelligence (CONTEXT'99)*, Trento, Italy, September 9-11, 1999.

[15] Trinh, V.C., Stensrud, B.S. and Gonzalez, A.J., "Implementation and Analysis of a Context-Based Reasoning Model on a Physical Platform," *Technical Report*, Intelligent Systems Laboratory, Orlando, FL, 2003