Context-based Representation of Intelligent Behavior in Simulated Opponents

Avelino J. Gonzalez
Electrical and Computer Engineering Dept.
University of Central Florida
Orlando, FL 32816-2450
ajg@engr.ucf.edu

Robert Ahlers
Naval Air Warfare Center
Training Systems Division
Orlando, FL 32826

Abstract

This article describes and evaluates a concise, yet rich representation paradigm that could effectively and efficiently be used to model the intelligent behavior of opponents in a simulation-based tactical training system. This feature would be quite useful in the training process for two reasons: 1) the trainee would face a realistic enemy who is knowledgeable about tactics in the domain of interest and, 2) the instructor would not be burdened with playing the part of the enemy in those training systems where this is commonly done.

The representation paradigm proposed is based on the idea that applicable tactical knowledge is highly dependent upon the situation being faced by the decision maker (i.e., the context). A combination of script-like structures and pattern-matching rules in an object-oriented environment could serve to hold all knowledge pertinent to the context present at a specific time. This paradigm has been preliminarily tested in a prototype system that incorporates the knowledge of a submarine tactical officer on a patrol mission. Evaluation of the prototype shows that the context-based paradigm promises to meet the desired levels of conciseness and effectiveness required for the task.

1. Introduction

The use of autonomous and intelligent simulated adversaries in a training simulation can provide a more realistic experience to the trainees than would be presented otherwise. This would be especially true for tactical training simulators if an intelligent simulated adversary could be made to react and counter the trainee's tactics in a realistic fashion. Intelligent simulated adversaries will be henceforth referred to in this article as *Autonomous Intelligent Platforms* (or AIP's). They can be defined as instance representations of military platforms (i.e., submarine, destroyer, tank, helicopter, fighter aircraft) in a training simulation which behave as would a real platform in actual battle from a tactical standpoint.

Tactical knowledge is required in order to endow AIP's with the ability to act, not only intelligently, but also realistically, in light of a trainee's actions. In general, tactical knowledge can be said to address time-stressed tasks which require 1) assessment of the situation at hand, 2) selection of a plan to most properly address the present situation, and 3) execution of that plan [Thorndike, 1984]. It is how this knowledge is represented and manipulated that is the focus of this research project.

Most tactical tasks in the military consist of a pre-defined set of actions which are embarked upon after a certain situation has been recognized. The situation could be a mission, a set of orders, or merely a reflection of a specific set of battle conditions at the moment. The problem faced, therefore, is two-fold: 1) how to recognize the present situation (referred to as situational awareness), and 2) what to do when the situation is recognized (referred to as implementation of actionable information).

The basis of our approach to the problem of concisely and effectively representing AIPs is based on the following hypotheses:

1) Tactical experts are proficient at their task by recognizing and treating only the

key features of the situation, and abstracting these for use as the premises for the general knowledge. Thus, they only use a small, but important portion of the available inputs. An example of this is the tactical exercise of driving an automobile.

The driver of an automobile is generally bombarded with a multitude of inputs when driving: audio inputs such as engine noise, road noise, the radio, conversation with passengers, etc.; visual inputs such as the instruments, other automobiles, the surrounding scenery, pedestrians, etc.; tactile inputs such as vibrations of the car, the position of the steering wheel, the gear shifter, the clutch, etc.. These inputs are cognitively handled rather easily by the driver when they are all in the normal or expected range. However, if one of these should deviate from normal, such as the abnormal noise and vibrations resulting from a tire blowout, the driver will immediately focus on these inputs in order to recognize the present situation as a blowout, while ignoring all the other inputs being received.

2) There is only a limited number of things that can realistically take place in any situation. Using the example above, it would not be expected that a tire blowout take place while waiting at a stop light. This can be used to advantage to prune the search space of the problem, since there is no need to consider a blowout while waiting at a stoplight. Getting rear-ended, on the other hand, is a much more likely proposition.

3) The presence of a new situation will generally require alteration of the present course of action to some degree. For example, the recognition of a blowout at highway speeds will cause the driver to coast to a stop while maintaining a hard grip on the steering wheel, and directing the car towards the shoulder of the road. Thus, the context changed from one of normal driving, to one of blowout, with its attendant course of further action. This context remains in effect until the car comes to a complete stop, at which point another situation will be recognized and acted upon (i.e., get out of car, inspect tire, change tire etc.).

The work described here is based on the idea that by associating the possible situations and corresponding actions to specific contexts, the identification of a situation is simplified because only a subset of all possible situations are applicable under the active context. The work also addresses what course of action to follow when a situation is correctly recognized.

2. General Description Of Context-based Representation Paradigm

Our approach to implementing the concepts described above lies in the use of *Scripts*. A script is a knowledge representation paradigm developed by Roger Schank at Yale University [Schank, 1977] which attempts to capture the actions, objects, persons, and concepts that may be related within a given context. For example, a restaurant script will be composed of all the actions which are typically part of going to a restaurant, such as reading the menu, ordering the meal, eating it, paying the bill, etc. A restaurant script also contains props, objects which are typical to a restaurant scene from the customer's standpoint, (e.g., tables, chairs, menus, food, eating utensils, napkins, salad bars, etc.) as well as actors (i.e., waiters, hostesses, chefs, busboys, etc.). The actions involved are only those typical of the restaurant experience. It would not be normally expected, therefore, that the customer wash her car at the restaurant.

This concept can be easily extended to military tactics, where a script can be used to express the set of steps (at either a high or low level) that are necessary to carry out the action required by the present situation. Within the context of a mission, there is a limited number of things that are generally expected in terms of actions to carry out and the expectations in regards to the possible situations. It would be quite difficult to represent all this knowledge using rules alone. Thus, the basis for this research project is to use scripts (in addition to objects as well as a minimal number of rules) as the knowledge representation paradigm for a set of AIP's. For lack of a better name, this representation and reasoning paradigm will be referred to as *Context-based Reasoning* (CxBR).

The approach proposed here is based on the following assumptions:

1) Life for an AIP is a continuous and dynamic decision making process. Decisions are heavily influenced by a never-ending sequence of contexts, each of which, when active, regulates the behavior of the AIP as well as provide an expectation for the future. Active contexts change not only in response to external events or circumstances, but also as a result of actions taken by the decision maker (the AIP itself). A context can be likened to a situation that has been recognized, and which has a prescribed set of procedures that must be carried out, either sequentially, methodically, or arbitrarily. One example of a context would be driving an automobile on an interstate highway at normal cruising speeds. The behavior of an AIP in that situation is controlled by the context that is active for it at the time.

2) The active context may not be the same for all AIP's at the same time. This is reasonable to expect, since each may have a different mission, different sensor inputs, different capabilities, a different physical location, etc.

3) At least one specific context or set of contexts is always active. More than one context can be valid, but only one may be active. Furthermore, all valid contexts must be compatible. Thus, one context must be the central focus of attention. For example, using the case of traveling in an automobile, the driver may be cruising on an interstate normally at the speed limit, a situation which may be characterized as **normal-highway-driving**. Moreover, he/she may also be hungry which can be as a situation labeled **driver-hungry**. If he/she is more anxious to arrive at the destination than willing to satisfy the hunger, then the first situation will dictate the action being undertaken. Thus, the active context would be **normal-highway-driving**. Otherwise, a **driver-hungry** context will be the active one and the action will shift towards finding a place to eat.

4) Contexts are represented temporally as intervals of time rather than time points. Contexts can be considered to be transitions to reach a goal (look for a place to eat), or they can be a goal in themselves (eating).

5) Goals can be time points, but only to serve as transitions to other contexts. For example, arrival at a destination can be defined as a goal of **normal-highway-driving** and it can be represented as a time point, but it is not a context in its own right, only a transition to another context (e.g., **being-there**). This process may go on until the mission ends.

6) Only a limited number of things can take place in any single context. A situation, therefore, by its very nature, will limit the number of other situations that can realistically follow. Using as an example the domain of submarine warfare, it would not be expected that the submarine would be attacked in its own home port. This can be used to advantage to prune the search space of the problem, since there is no need to watch out for a torpedo attack while waiting to be resupplied at port. If unexpected situations do take place, that introduces the element of surprise into the AIP's behavior, which is highly consistent with the real world.

7) Certain cues exist which will indicate that a transition to another context is desirable. This makes use of the hypothesis that experts look for certain few specific cues which that will indicate a new situation (e.g., the vibration on the steering wheel upon a tire blowout.)

8) The presence of a new context will alter the present course of action and/or the applicable expectations to some degree. For example, the recognition of a blowout at highway speeds will cause the driver to attempt to coast to a stop while maintaining a firm grip on the steering wheel, and directing the car towards the shoulder of the road. Thus, the context changed from **normal-highway-driving**, to one of **tire-blowout**, with its attendant requisite action. This context remains in effect until the car comes to a complete and safe stop (the goal), at which point another context will be recognized and acted upon (e.g., **fix-tire**).

By associating the potential contexts and corresponding actions to specific situations, the identification of a situation can be simplified because only a subset of all possible situations is applicable under the active context. This context-based approach also easily addresses what course of action to take when a situation is recognized.

This is in some ways similar to a system proposed by Thorndike [1984] called "Context Template-driven SAFOR". However, the latter appears to implement AIP's that are intelligent in a much higher level in that they can only respond to orders from higher command, or requests from subordinates. They do not appear to be able to perceive the environment independently. Moreover, the transition from one context to another appears to be significantly more rigid that what is being proposed here, which may lead to unintelligent decisions. Lastly, the system seems to not provide the capability to its AIP's to plan, a significant disadvantage.

Czejdo and Eick [Czejdo, 1993] present an environment in which addresses the problem of large-scale knowledge management. Called the Tanguy Knowledge Base Management System, it integrates rules, objects and database features in order to take advantage of their respective features.

Dreyfus and Dreyfus [Dreyfus, 1986] take exception to the idea of using contexts to simulate human intelligence in computers. They correctly point out that there can exist many contexts in the course of human life, and to attempt to account for all of them is a hopeless task. Nevertheless, their argument arises from the standpoint of refuting the claims that computers can be intelligent in the same way as humans, in all aspects of human intelligence. The objective of this work is less ambitious in scope, since the AIP's do not have to have the breadth of knowledge possessed by humans in order to appear intelligent in a training simulation. Rather, they only must appear to behave as a human enemy would in a very specific and narrow domain (i.e., submarine warfare, tank warfare). In fact, this may mean that they should not display optimal behavior, as that is not always typical under wartime stress. It is our belief that applying context-based reasoning as described below presents a highly effective and efficient methodology for imparting sufficient intelligence to AIP's so as to achieve their objective in a training simulator.

2.1 Representation of Contexts

In CxBR, contexts are the most important representational item. Much knowledge about how the AIP should behave, as well as to what other contexts it can transition is stored in the context objects themselves. There are three levels of contexts that can be represented, and they are ordered hierarchically. These are 1) the mission context, 2) the major contexts and 3) the sub-contexts. These will be described below.

2.1.1 Mission Contexts

A mission-context (simply referred to as a mission) is an overall definition of the objectives of the scenario. It defines the objectives as well as the constraints of the operation. The mission can also define the things to avoid during the mission. Examples of missions in the domain of submarine warfare are SEARCH-AND-DESTROY enemy submarines, MINING a harbor or choke point, GATE-KEEPING, BATTLE-GROUP-ESCORT, ANTI-SURFACE-OPERATION, SPECIAL-OPERATIONS, and others. A mission can define the types of lower level contexts which may be necessary during the execution of this mission. A mission context can also describe the political environment under which the mission is to be carried out. For example, if a hot war is in effect, then the rules of engagement will surely be different than if a cold war is in effect. No more than one mission will be active at any one time, and missions are mutually exclusive. So, a new mission would have to bump an existing mission from active status. In practice, however, there would be little need to change missions during the course of a training session.

The mission defines the constraints as well as the Major-Contexts therein. It is a class definition in an object-oriented environment and contains the following attributes:

Constraints: This attribute lists all the constraints that are imposed on the AIP during this mission. Some of these could be: withhold fire unless fired upon, any limitations placed upon the submarine's performance characteristics, etc.

Avoid: This attribute describes anything that must be avoided at all times throughout the training scenario. One obvious one is destruction-of-self, but there may be others such as avoid counter-detection at all costs, etc.

Major-Contexts: This attribute lists the Major-Contexts present in the mission. For example, for a patrol mission (called SEARCH-AND-TRACK), the major actions for the AIP will be getting to its assigned sector, searching the sector in an appropriate fashion, tracking an enemy contact if one is found, and breaking contact to return home when certain parameters are fulfilled.

2.1.2 Major-Contexts

Major-Contexts are the main focus of the Context-based reasoning and representational paradigm. They contain all the necessary information to operate the AIP, as well as to determine when the active context should be deactivated and another one put in its place.

A major-context (or simply context for short) is a tactical operation undertaken as part of the mission in order to assist in achieving the goals set forth. One context is always in control of the AIP, and contexts are, also by definition, mutually exclusive of each other. Unlike the mission, however, contexts are normally activated and deactivated many times throughout the course of a training session. A context is activated by retracting from the fact base the fact that identifies the active status of the current context, and calling the initialization procedure of the newly activated context. The latter will assert into the fact base a new fact identifying the new context as the active one. This will allow the monitoring rule(s) that pertains to this context to become active and fire periodically. Furthermore, the initialization message will also modify any parameters that so require modification, such as possibly the AIP's heading, speed, depth, etc. In some missions, the sequence of some of the contexts will be known a-priori. Although some contexts may become invalid through the simple passage of time, this is not common. In most cases, the context will cease to be applicable due to either an action taking place or the completion of its task. Therefore, contexts are generally assumed to be active indefinitely, until bumped from active status by another context.

Each context is defined as a class in an object oriented environment, and possess the following attributes:

Initializer: References the name of the message-handler which is executed whenever the context/sub-context is first activated to initialize all required variables.

Objective: The objective slot puts a message as to what the objective of the context/sub-context is. The objective is in general terms and it references a frame that has some attributes that are the goal of this context/sub-context.

Compatible-next-major-context: This attribute lists those contexts to which transition from the current context is acceptable.

Compatible-sub-context: This attribute is a list of all sub-contexts which are compatible with the current context. For example, it would not be advisable to put an automobile in cruise-control when a blowout has taken place. Thus, the cruise-control context would not appear on that list.

Additionally, some contexts will have slots that are specific only to them, and deal with universal variables that need to be known throughout the entire simulation. For example, the attack context will have a slot defining the target of the attack and another defining the number of weapons used. Likewise, the under-attack context will have a slot defining the aggressor (source of the weapons bearing down on the AIP).

There may be other attributes added to the class definition for context/sub-contexts in the future. One particularly desirable would be a further refinement of the compatibility aspect by providing a numerical weight to each context to decide which one would be more desirable in the case where more than one would be acceptable given the current situation. This competing context concept is further described later in this article.

2.1.3 Sub-Contexts

Sub-contexts are lower level tactical procedures which are not critical in and of themselves to reaching the mission objectives. They are typically of temporally short duration. Sub-contexts are at this time mutually-exclusive with one another, but can be compatible, and thus co-exist, with the contexts. It is expected, however, that in the future, compatible sub-contexts may co-exist with one another on active status as long as they control different variables. In cases of incompatibility, a sub-context will not be activated when an incompatible context is active. Likewise, when a new context is activated while an incompatible sub-context is active, the sub-context is immediately "short-circuited". In any case, however, the contexts always take precedence. It is not necessary for one sub-context to be active at all times as is the case with contexts. When no sub-context is active, the sub-context is said to be "none".

The attributes of a sub-context objects are quite similar to those of a context, and thus will not be described further.

2.2 Situation Assessment and Transitioning Between Contexts

One of the foundations of the CxBR approach is that by knowing what the AIP is doing at any one time, it can know what to expect. This greatly facilitates the task of situational assessment. One example in the automobile driving domain is that when on an interstate highway, one does not have to be concerned with traffic crossing the roadway, as there are no intersections per se. When driving on city streets, however, one of the most dangerous situations is when the automobile approaches intersections, and thus, a driver has to be especially aware of them. The situational awareness function in the existing prototype is done by simply looking for parameter values that indicate that a change in context is warranted. This is a rather simple, yet quite effective means of doing situational assessment under CxBR.

The basic recognition of the situation is done through pattern-matching rules. While this might not seem to be a concise way of carrying this out, the use of the active-context and/or active-sub-context pattern in the rule premise will significantly limit the solution space of the search as was described in the previous section. Rules will have a pattern in their premises that indicates the active context to which they are applicable. Only when there is a fact in the factbase indicating the active status of the appropriate context will these rules be "active" and capable of being executed.

The transition among the various contexts is a critical issue in CxBR. This approach is based on the use of monitoring rules. Each major-context/sub-context will have at least one of these. These rules will fire continuously (every simulation cycle) as long as its parent context is active. In its right hand side, the rule will have a conditional statement(s) that will monitor the parameters which are relevant to the continuation of the context. Examples of these parameters are: whether the enemy contact has been detected, whether it is moving towards or away from the AIP, whether it is within firing range, etc. Once these parameters are satisfied and a change of context/sub-context is indicated, the rule will retract the fact that advertises the active context/sub-context. This will prohibit these monitoring rules from firing any longer. The rule will have the transition information embedded, so that it will call the initializing message for the new context/sub-context. The initializer will set the revised parameters on the AIP as may be required, and post into the factbase the new facts that announce the newly-activated context/sub-context. This will allow the monitoring rule for that context to begin firing.

In some cases, the transition to a new context/sub-context would be a result of an "external" message (e.g., a communication from fleet command). Examples of such would be a message to return home, or go to periscope depth to receive or transmit a more detailed communication. This external message is represented as a fact posted on the fact base which has a "prompt" indication (i.e., (communication prompt)). In such circumstances, an additional rule is required, which fires only once, to retract the current context/sub-context fact (as well as the prompt fact) and to invoke the initializing message that activates a new context/sub-context.

Of course, there are also universal monitoring rules which are not tied into any one context/sub-context. These rules search for situations which could occur under any context, such as being fired upon by an aggressor, or the detection of an enemy contact.

A more thorough discussion of CxBR is included in [Gonzalez, 1993; 1994].

3. Implementation and Evaluation of the CXBR Approach

In order to properly evaluate the ideas set forth in this article, two prototypes were developed and tested. Prototype #1 was developed in the submarine warfare domain, while prototype #2 was in automobile driving. Both of these will be discussed in this article, but the first prototype is more comprehensive in nature and will thus be discussed more thoroughly.

The two prototypes were implemented in CLIPS 5.1. This environment proved to be a good framework for the task, but certainly not ideal. Memory limitations of the DOS-based CLIPS version, its inability to match facts in the factbase with patterns anywhere but within rule premises, and its basic inadequacy as a simulation tool often made it cumbersome to use.

3.1 Submarine Warfare Prototype

While the implementation of submarine tactics was not per se an objective of the current investigation, it became clear that to design an appropriate architecture and develop a prototype that verified the use of that architecture, a limited set of submarine warfare tactics had to be defined. The first prototype was thus built to evaluate the behavior of the AIP in a SEARCH-AND-TRACK mission, in the presence of one enemy submarine (called *ownsub*). In order to make it self-contained, the prototype incorporated its own simulation of the submarine warfare environment, which took up considerable computing resources.

All missions, major-contexts and sub-contexts were represented as classes in the CLIPS Object Oriented Language (COOL), as were the SUBMARINE classes and the weapon classes. Monitoring rules, of course, were implemented as CLIPS rules. Initialization messages were implemented as message-handlers in COOL. The central manager was composed of the main loop which contained all the procedures that were to be repeated every simulation cycle, and a number of other functions which carried out calculations such as distances, bearings, etc., when required. The output was in the form of a text report which, every five seconds, listed the x and y positions as well as the depth, of all submarines and weapons involved in the scenario.

Interruptions could be made to the simulation to introduce control of the AIP by the instructor. Interactions permitted with the AIP through this interruption mechanism included orders to attack, communication prompts, baffle-clearing prompts, and return home orders. Interactions with other simulated submarines (called *ownsub*, and driven by the student being trained) included changing its heading, depth, speed, and firing its weapons.

Upon satisfaction that the full Prototype #1 operated correctly, a version of it was built that could be interfaced with an external graphical simulation. This would allow the investigators to evaluate the feasibility of incorporating this technique within existing simulator training systems, a critical step in verifying its usefulness. The simulation employed for this purpose was the *Intelligent Platform Modeling System* (IPMS), a testbed being developed at the Naval Air Warfare Center, Training Systems Division in Orlando, FL.

The externally-interfaced version of Prototype #1 was developed by stripping off all the code from the full prototype which served to support its built-in simulation. A networked interface was used as the means of communication between the CLIPS-based AIP model and the DOS-based IPMS simulation. In order to resolve the memory limitations of the DOS-based CLIPS 5.1, a UNIX-based version was employed, running on a Silicon Graphics workstation.

The AIP (also called *opsub*) implemented in Prototype #1 was found to behave tactically correctly from a qualitative point of view when subjected to several different situations. The situations to which opsub was subjected included transiting to its designated sector using a sprint-and-drift tactic, searching the sector for enemy activity, maneuvering into position to track enemy contacts, tracking such contacts, clearing its baffles, getting into position to attack the enemy attacking the enemy when externally ordered, (in the spirit of a reconnaissance mission), and evading enemy attacks. Opsub was placed in these situations through the control of the location, bearing, speed, depth and weapons of ownsub.

The externally-interfaced prototype was able to transit to the sector using a sprint-and-drift maneuver, and carry out a baffle-clearing tactic while doing a search of the sector. It was additionally capable of searching the sector, detecting the enemy (ownsub), maneuvering into position to track ownsub, and breaking contact to return home when told to do so. The prototype also performed an approach to fire its weapons. However, due to the inability of the IPMS to model weapons in the water, evaluation of attacking and evading attacks was not possible.

The qualitative evaluations of the two versions of Prototype #1 described in this sections allows us to conclude that: 1) the CxBR paradigm can be used to accurately represent the tactical behavior of an AIP from a qualitative standpoint, and 2) the paradigm has been shown to be compatible with a distributed simulation environment.

Conclusion #1 is an essential one, since inability to be used to represent tactical behavior would invalidate the CxBR paradigm without the need for any further evaluation. Conclusion #2 is significant from a usefulness standpoint if AIP's are to be retrofitted to existing simulators. Moreover, it is also important in light of the U. S. Army's interest in distributed interactive simulations.

3.2 Automobile Driving Prototype

Qualitative success in the performance of the AIP prototype does not provide a complete picture of the viability of the CxBR technique. Furthermore, conciseness can only be unequivocally judged by comparing a CxBR prototype with an equivalent purely rule-based implementation of the knowledge and capabilities exhibited by a CxBR prototype. This was accomplished and the evaluation is described in the section that follows.

The scope of the automobile driving prototype was more modest than that of Prototype #1. This prototype used an automobile simulator system [Klee, 1991] and implemented a short scenario where the AIP automobile (labelled *student car*) is cruising on a two-lane road and approaching a curve near an intersection where another car (labelled *simulation car*) is waiting to turn left into the road ahead of it. To complicate matters, a small truck (*simulator van*) is coming around the bend in the opposite direction. Figure 1 graphically depicts a bird's eye view of the scenario faced by the AIP. The AIP is tasked with avoiding a collision with both the car and the van, as the car attempts to cut in between the AIP and the approaching van. The scenario was varied by running various tests with different "release distances" for the car and the van. This meant that the distance available for the AIP to maneuver ranged from one where no real danger was present, to one where a collision was physically inevitable. The courses of action available to the AIP were to: 1) slow down (possibly through the application of brakes) in order to maintain a distance between itself and the simulation car; 2) brake and swerve to the left if there is sufficient distance to avoid hitting the oncoming van; and 3) swerve to the right (off the road) if there isn't sufficient distance.

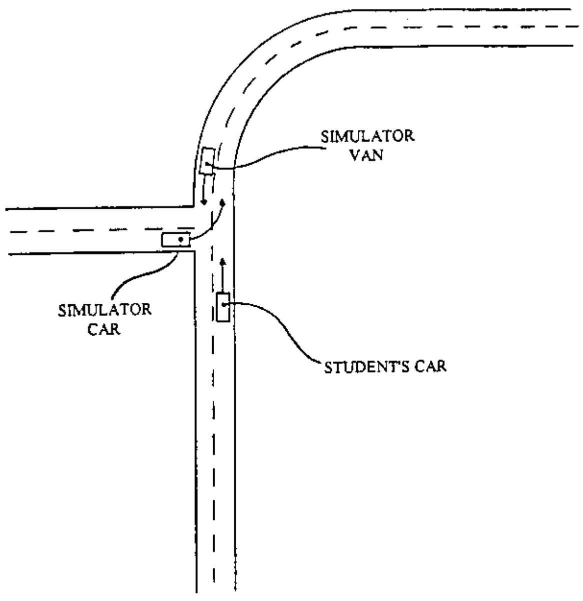

Figure 1 - Bird's Eye View of Simulated Scenario

Criteria of comparison	CxBR	Rules-only
Execution time avg. (sec)	124.1	126.91
CLIPS elements	53	43

Table 1 - Summary of Quantitative Evaluation of CxBR

The tactics used by the AIP to accomplish this were represented through contexts as described earlier in this article. At the same time, a rules-only representation was also implemented for the purpose of comparison. Both implementations were done in

CLIPS 5.1 to make the comparison as straightforward as possible. The parameters compared were 1) the number of CLIPS elements used by each implementation and, 2) the time of execution required by each implementation. The results are shown in Table 1 above. A full description of the evaluation procedure is found in [Brown, 1994].

The execution times for the CxBR implementation were 2.26% better than those for the pure rule-based alternative. Although this difference is nearly insignificant, it does demonstrate that the CxBr is more efficient. This difference is expected to become more pronounced as the size and scope of the domain expands.

In regards to the number of CLIPS elements, the rule-based alternative was actually more concise. Once again, the limited scope of the prototype skews the results, since the CxBR alternative requires a "fixed overhead" in CLIPS elements in order to adequately represent the tactical knowledge. This overhead is in the form of the classes for each context or sub-context defined for the tactic, and the message handlers involved with each class instance. As the situation becomes more complex, this overhead becomes a smaller part of the total knowledge, while the rules required for pure rule-based reasoning become more numerous to account for all the possibilities.

4. Summary and Conclusions

The use of contexts to represent and reason about tactical knowledge has the advantage of encapsulating all facets of such knowledge as it applies to a small slice of the entire domain knowledge. By modularizing the knowledge in such a way, and by explicitly expressing the relationships between the various contexts such that the number of possible transitions between contexts are inherently limited, efficiencies can be implemented. These efficiencies are in terms of economy of knowledge as well as in efficiency of execution of the system.

This article describes the concept of Context-based Reasoning as well as two prototypic implementations of these concepts in order to evaluate its effectiveness in achieving the efficiencies expected. The prototypes were successful in achieving the objectives of the investigation. Nevertheless, areas of further work were discovered where the present system is deficient, namely in how to deal with time, either as historical information or as in planning the next move. Basically, the prototypes are reactionary in nature, as their planning capabilities are rather limited. Nevertheless, from a conceptual standpoint, planning is quite consistent with the general CxBR approach, and such a capability will be featured in future versions of the prototypes.

5. Bibliography

[Brown, 1994] Brown, J. C., "Application and Evaluation of the Context-based Reasoning Paradigm", Master's Thesis, Department of Electrical and Computer Engineering, University of Central Florida, Orlando, FL, July, 1994.

[Czejdo, 1993] Czejdo, B. and Eick, C. F., "Integrating Sets, Rules, and Data in an Object-Oriented Environment", *IEEE Expert*, February, 1993, pp. 59-66.

[Dreyfus, 1986] Dreyfus, H. L., and Dreyfus, S. E., *Mind over Machine*, New York: MacMillan/ The Free Press, 1986.

[Gonzalez, 1993] Gonzalez, A. J. and R. H. Ahlers, "A Context-based Representation of Tactical Knowledge for Use in Simulation-based Autonomous Intelligent Platforms", Proceedings of the 15th Annual Interservice/Industry Training Systems and Education Conference, Orlando, FL, November, 1993, pp. 543-552.

[Gonzalez, 1994] Gonzalez, A. J. and R. H. Ahlers, "A Novel Paradigm for Representing Tactical Knowledge in Intelligent Simulated Opponents" Proceedings of the 7th International Conference in Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, Austin, TX, May 1994, pp. 515-523.

[Klee, 1991] Klee, H. I., "Development of a Low Cost Driving Simulator", *Technical Report*, Department of Computer Engineering, University of Central Florida, 1991.

[Schank, 1977] Schank, R. C., and Abelson, R. P., 1977, *Scripts, Plans, Goals and Understanding*, Erlbaum, Hillsdale, NJ, 1977.

[Thorndike, 1984] Thorndike, P. W., and Wescourt, K. T., "Modeling Time-stressed Situation Assessment and Planning for Intelligent Opponent Simulation," *Final Technical Report* PPAFTR-1124-84-1, sponsored by the Office of Naval research, July, 1984.

6. Author's Biographies

Avelino J. Gonzalez is an Associate Professor at the Electrical and Computer Engineering Department at the University of Central Florida. His area of research is in intelligent simulations, specifically, automated intelligent platforms in a training simulation. He has a PhD in Electrical Engineering from the University of Pittsburgh, and a Bachelor's and Master's degrees also in Electrical Engineering from the University of Miami.

Robert H. Ahlers is a Research Psychologist with the Human/Systems Integration Division of the Naval Air Warfare Center, Training Systems Division. He has managed research projects concerned with the application of knowledge-based modeling to the simulation of intelligent agents within a training environment. He graduated from the University of Virginia with B.A. and M.A. degrees in experimental psychology and from North Carolina State with a Ph.D. in Human Factors.