A Reinforcement Learning Technique For Enhancing Human Behavior Models In A Context-based Architecture

2008

David Aihe
University of Central Florida

A REINFORCEMENT LEARNING TECHNIQUE FOR ENHANCING HUMAN BEHAVIOR MODELS IN A CONTEXT-BASED ARCHITECTURE

by

DAVID O. I. AIHE
B.Eng University of Benin, 1996
B.A. Metropolitan State University, 1998
M.S.Cp.E. University of Central Florida, 2001

A dissertation submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Computer Engineering
in the School of Electrical Engineering and Computer Science
in the College of Engineering and Computer Science
at the University of Central Florida
Orlando, Florida

Fall Term
2008

Major Professor: Avelino Gonzalez

ABSTRACT

A reinforcement-learning technique for enhancing human behavior models in a context-based learning architecture is presented. Prior to the introduction of this technique, human models built and developed in a Context-Based reasoning framework lacked learning capabilities. As such, their performance and quality of behavior was always limited by what the subject matter expert whose knowledge is modeled was able to articulate or demonstrate. Results from experiments performed show that subject matter experts are prone to making errors and at times they lack information on situations that are inherently necessary for the human models to behave appropriately and optimally in those situations. The benefits of the technique presented is two fold; 1) It shows how human models built in a context-based framework can be modified to correctly reflect the knowledge learnt in a simulator; and 2) It presents a way for subject matter experts to verify and validate the knowledge they share. The results obtained from this research show that behavior models built in a context-based framework can be enhanced by learning and reflecting the constraints in the environment. From the results obtained, it was shown that after the models are enhanced, the agents performed better based on the metrics evaluated. Furthermore, after learning, the agent was shown to recognize unknown situations and behave appropriately in previously unknown situations. The overall performance and quality of behavior of the agent improved significantly.

ACKNOWLEDGMENTS

First, I would like to thank the Almighty God for giving me the strength to accomplish this research. I would like to thank my family for the wonderful support and encouragement they provided during the course of writing this dissertation, my wife Susan, our kids, my mom, and my dad, you are all wonderful. Special thanks go to my advisor and committee chair Avelino J. Gonzalez for the support throughout the process of this dissertation. I would also like to thank the rest of my committee for their support.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABREVIATIONS

Definitions, Acronyms, and Abbreviations

The following is a list of acronym definitions that will be used throughout this document

- CxBR – Context Based Reasoning
- RL – Reinforcement Learning
- SME – Subject Matter Expert
- GFB – Global Fact Base
- HBR – Human Behavior Representation

CHAPTER 1: INTRODUCTION AND BACKGROUND

1.1 Overview

Human behaviour can be said to be largely dictated by decision-making and their resulting actions. These decision making processes determine how a person behaves in any given situation. Henninger [125] defines behaviour as "any observable action or reaction of a living organism." She notes that some psychologists extend this definition to "include conscious phenomena like perception, cognition, and judgements" [125]. The Oxford dictionary [1], defines "behaviour" as "the actions or reactions of a person or animal in response to external or internal stimuli". The external stimuli include touch, smell, sight, as well as others. How a person reacts to these stimuli dictates his or her actions at that point in time. Consider a driver faced with a choice of driving beyond the speed limit on a freeway and eventually arriving at his/her destination on time, or obeying the speed limit and arriving at his/her destination late. The decision and subsequent actions made by this driver ultimately constitute his/her behaviour on the freeway.

In some domains, for example aviation and military operations, the behaviour of a person is divided into *tactical* and *strategic* behaviours. Schutte [194] states that "*tactical behaviours* are generally considered to be near-term, dynamic activities" while *strategic behaviours* usually involve the decision-making process based on the overall mission of the person in the long run [194]. Tactical behaviours involve the immediate decision-making process of an individual, whereas strategic behaviours involve a planned out set of decisions by an individual. Latorella & Chamberlain [195] explain that the time pressure in any given situation differentiates whether the individual will operate in a tactical mode or a strategic mode. They [195] go further to present an example using pilots. When pilots attempt to solve known and anticipated tasks, they usually exhibit strategic behaviours, whereas when they are under a time-constraint, they act or react to the situation presented based on some predefined mental list, i.e., they exhibit tactical behaviours [195].

Modelling strategic behaviours are usually straight forward because they involve known and anticipated tasks. On the other hand, modelling tactical behaviours can be cumbersome. This is because the steps involved in solving tactical problems are usually dependent on the actions taken at any given moment. Modelling the decision-making process has been studied by several investigators [12, 14] among many others. A few human behaviour modelling and representation techniques exist. These techniques are discussed in detail in later sections of this chapter. After modelling the human behaviour, the task of representing the model in an efficient computational paradigm is not a trivial one.

The representation of human behaviour has been investigated by many researchers including [6, 36, 44, 4, 7, 9, 10, 2, 34, 87, 90, 40, 83, 88, 82, 84, 85], amongst many others. How to efficiently and effectively model and represent the way a person acts or reacts in a given situation has no definitive solution. Some human behaviour modelling and representation paradigms [2, 3, 4, 5] suggest ways to do this. However, these techniques all have the same deficiency; - the human expert is the source of knowledge for these models. Eliciting knowledge from experts has several limitations. These limitations include the lack of acquisition of implicit knowledge from experts and the reliance on expert interpretation of the real world, which is not always accurate. These limitations are discussed in subsequent sections of this chapter. Based on these limitations on the acquisition of knowledge, some modeling and representation techniques are either over-simplified by having too many assumptions (for example a person not being able to learn based on his / her experience or experiences of others), or only provide a limited view of a person's tactical behavior in a given situation (for example constraining the actions available to an agent in the model in tactical situations based on the actions known to only one expert). Furthermore, some of these methods do not address the ever-changing behaviour of humans in a given situation or in a new situation or during a change in situations, because they were designed for modeling strategic behaviors.

Many factors affect the way a human behaves in a given situation. For example, one would expect a person being held at gunpoint to cooperate with his or her captors. However, could this behavior in all certainty, represent every person? What about a martial arts expert who has an opportunity to overcome his/her captors? Would this person react similarly? This suggests that there may be multiple ways humans may behave in a particular situation. There are many variables involved in modeling the behavior of a person in any given tactical situation. Trying to address these variables by explicitly relying on the knowledge acquired from an expert may lead to unsolvable problems or representations that don't adequately fit the situation. For example, if the knowledge used in building the model of a soccer player is acquired from an expert who believes in only passing the ball to a teammate in front, when the model is placed in situations that necessitate otherwise, the behavior exhibited might not be optimal. With this in mind, several researchers [6, 7, 33, 101] have proposed representations to address some tactical situations.

Gonzalez & Ahlers [7] describe and implement a methodology called *Context-Based Reasoning* (CxBR) that models a humans' expected behavior in specific particular situations. CxBR is a paradigm that models human behaviors in terms of contexts. This method seeks to limit and reduce the complexity inherent in human decision-making by limiting the number of events available for the agent to think about in any given situation. Several successes have been achieved using this method, for example [4, 127]. While a generally effective method however, reliance on *Subject Matter Experts (SME)* [1]from whom knowledge is obtained can limit its effectiveness.

This research describes a method that eliminates one major limitation introduced to most human behavior modeling techniques during the knowledge acquisition process, i.e., the limitations of relying on SME knowledge. Like other paradigms, Context-Based Reasoning also suffers from this problem. The elimination of these limitations would lead to a more robust methodology that can be extendable to most human behavior representation techniques and most domains. Furthermore, apart from augmenting SME knowledge, it would minimize or eliminate other errors built into the system during the acquisition of knowledge such as the introduction of conflicting knowledge to a model.

To achieve this, we incorporate *Reinforcement learning* (RL) within Context-Based Reasoning. Reinforcement learning is a machine learning strategy that assigns rewards (positive or negative) as an agent (simulated or live) interacts with its

[1] Subject Matter Experts (SME) are people with experience in the domain being simulated. They typically provide knowledge that is used in the knowledge base of the simulation.

environment (immediate or distant). The synergistic combination of these methodologies promises to significantly enhance CxBR's ability to represent human tactical behavior.

1.2 Human Behavior Representation

As was previously defined, "human behaviours are the actions or reactions of a human in response to some external or internal stimuli" [1]. Some researchers have attempted to represent this response, for example [4, 5, 6, 52]. Pew & Mavor [52] note that the military simulation community defines human behaviour representation as "models of human behaviour or performance utilized in military simulations". Researchers in the field of human behavioural representation are faced with at least three issues [24].

- To efficiently and effectively represent the behaviour of a human
- To efficiently acquire these behaviours
- To validate the acquired behaviours

To efficiently and effectively represent human behaviour, the five components of human behaviour stated by Flournoy [108] must be taken into consideration. These components are: *sensing and perception, working memory, cognition, motor behaviour and long-term memory*.

Sensory and perception refer to the inputs received from the environment. These inputs can include rewards, punishments or some other form of response from the environment on an action or group of actions performed by the human.

Working memory refers to the part of memory reserved for processing these inputs alongside other variables. The dictionary [1] defines *Cognition* as "The mental process of knowing, including aspects such as awareness, perception, reasoning, and judgment - That which comes to be known, as through perception, reasoning, or intuition; knowledge.” The way a human recognizes events, i.e. the humans’ level of awareness, perception, reasoning and judgement of his environment should be represented. Some people have a high level of situational awareness and perception whereas others do not. Yet others learn about their environments and become fully aware over a period of time.

Motor behaviour refers to the learning and control of human movement. Gonzalez and Dankel [197] note that motor skill (behaviour) “…is physical rather than cognitive-oriented” [197]. They give examples of driving an automobile, riding a bicycle, etc. Knowing how to effectively represent a person’s motor behaviour is important in obtaining a good model of human behaviour, for example how many times will a person attempt to balance a bicycle before becoming successful?

Long term memory refers to the stored memory that can be retrieved anytime. As a person performs an activity, he/she learns from that activity. Some time in future, the person might need to retrieve information on some past event. The ability to represent this process is important for a robust model.

There are many ways to model and represent human behaviour. Researchers have postulated many paradigms and architectures that address aspects of this problem. Flournoy [108] groups these architectures into three classes:

1. Finite-state machines

2. Task network models

3. Pure cognitive models

Paw & Mavor [52] note that the finite-state machines paradigm is the most utilized behavioural representation used for military simulations. Inasmuch as there are a number of models that address human behaviour, some constraints to modelling human behaviour exist. Giordano [50] explores some of these constraints to modelling human behaviour and concludes that most times the requirements for HBR systems exceed the capabilities of technologies currently in use. He suggests that unless the emergence of 'disruptive technologies' [50] occur, certain HBR characteristics will continue to remain beyond our reach.

Most of the existing models focus on specific areas of human cognition and behaviour [12]. This is in line with Brooks' [84] argument that it is better to build simple creatures in complex worlds and then gradually increase their complexity than to build creatures in simple worlds and then gradually increase the complexity of the worlds. He asserts that "human behaviour is the external expression of a mass of independent behaviours that don't have any central control or representations of the world" [84]. Some researchers have tried to create more complex behaviours by combining existing models. This allows a more robust and flexible architecture for any given simulation. For example, Van Lent et al. [12] achieve a more realistic view of the situation in an urban combat mission by integrating three human behaviour representations (PMFServ, AI.Implant and Soar to be described later) into a single virtual environment. These three paradigms are necessary to achieve a realistic view because during a modern urban combat mission, in the context of counter insurgency, there are potentially three 'types' or groups of behaviours exhibited by humans: An attacking army (a group of humans), usually has combat doctrine and rules of engagement to which they adhere during these types of missions. The second group of humans are an opposing militia that fights the attacking army through unorthodox means. This militia have no combat doctrines or rules of engagement to uphold. The third group of people includes civilians caught in the cross fire. These civilians have only one goal when caught in this situation - to survive.

Behavioural variability which is defined by Wray and Laird [14] as the "differences in observed behaviour when entities (human or otherwise) are placed in essentially the same situations" [14] is typically overlooked by most modelling and representation paradigms. This variability exists when a persons' or agents' subsequent action cannot be completely predictable in the same situation. The context of situation is defined by [14] as "both the physical environment (e.g. buildings, terrains) and the strategic/tactical environment (e.g. mission rules of engagement, command structure)". According to Wray & Laird [14], it is wrong for researchers to assume that variability implies simple dichotomies as correct/incorrect or expert/novice. They list the sources of variability in human behaviour as mental and physical differences. The types of variability in human behaviour include within-subject and across-subject. Wray & Laird [14] define across-subject variability as when two different people act differently when faced with the same situation. Within-subject variability is a situation where the same person acts differently at different times when faced with the same situation. Sukthankar, et al. [13] describe a method for modelling physical variability in MOUT (military operations in urban terrain) soldiers. They note that the lack of variability in agents make them predictable and ineffective in a simulation with their human trainees. Sukthankar, et al. [13] acknowledge that both physical and cognitive differences contribute towards the overall behaviour of a person. Hence, both should be taken into account when modelling the behaviour of a human. An example is that during a combat mission, it will be easier for a person with small stature (shorter, smaller) to successfully hide in a small chamber than a person with big stature (big, fat or tall). Consequently, a persons' stature should be taken into consideration when modelling how the person would behave. Likewise, when being attacked by gunfire, it is easier for a sharpshooter to hit a bigger person than a smaller person. Sukthankar, et al. [13] showed how incorporating physical variability can induce different behaviours from agents.

Another factor commonly overlooked in HBR modelling is the stress levels of humans. How do you represent a human's 'stress level' in an agent's behaviour? Mental workload has been shown to affect the performance of individuals [15]. The effect of stress in modelling human behaviour isn't relevant to this research because the effects of stress are negligible in the cause of breaking the limitations of SME knowledge. As such no further discussions are made on this subject in this dissertation.

Another aspect of HBR is emotions. How does one represent emotions? How does one represent when a person is sad, happy, etc? Some agents can recognize emotions. These agents can interact with humans in question and answer sessions [31].

Davis [46, 47] analyses the relationship between emotions in agent systems and their computational requirements, and notes that 'emotion-like' states could cause the system to be dysfunctional. On the other hand, computational agents can identify these states and utilize them before the system becomes dysfunctional. He also suggests that such analysis be carried out during the development of a complex system. McCauley [48] proposes a way to have an internal emotional judge in agents that enable representation of a broad range of emotions. Kort and Reilly [49] propose analytical models that incorporate emotions into the design of cognitive machines. This is also not relevant to this research for the reasons given above.

Computer Generated Forces (CGF) are defined as “automated or semi-automated entities (such as tanks, aircrafts) in a battlefield simulation that are generated and controlled by a computer system perhaps assisted by a human operator” [196]. There are some drawbacks to using CGF’s. Archer & Lavine [32] point out that “using CGF in training and operations planning can be compromised when the CGF do not behave as realistically as actual soldiers”. They described work done in improving the realism of CGF entities in constructive simulations. One of the drawbacks of current CGF technology includes the predictability and relatively unrealistic nature of the CGF “with respect to the natural variability of human performance”[32]. This variability results from different levels of training, aptitude, fatigue and other environmental stressors to which humans are usually exposed in a battlefield. [32]

Although the work of Archer & Lavine [32] is a definite advancement in the study of CGF’s, it is limited in scope because it can only be used in the military domain. The authors didn’t offer ways of generalizing their method to other domains.

Another problem with CGF is the coordination of collaborative works among agents. Easterbrook [29] identified shared understanding and conflict as two key factors that affect collaboration. Easterbrook’s [29] work on developing a model of collaborative behavior based on the concepts of shared understanding, breakdown and conflict is relevant in the field of CGF.

Gore [30] notes the importance of considering the physical as well as the cognitive aspects of behavior on performance when investigating human errors. He notes that identifying the factors that lead of human performance errors will provide a better understanding of performance. According to Gore [30], "researchers in HOOTL[2] have not paid much attention to the impact the environment has on the behavioral predictions generated by the cognitive models and the link between the cognitive processes in any situation and the behaviors of the model." Gore [30] discusses the contextual control model (CoCoM)[3] developed by Hollnagel [122, 123], that addresses this issue by providing the link through its cognitive process module. The underlying principle of CoCoM is that it "believes that a person's comprehension and action depends on how context is perceived and interpreted" Gore [30].

While considering emotion, variability, stress, physical factors, collaborative tasks and other such factors that affect human behavior, the work described in this dissertation subsumes these factors and only treats them implicitly in some cases and neglects them in others. That is, only their effect on the actions is evaluated through the actions alone.

1.3 Acquisition of Knowledge for Modelling Human Behaviour

As noted previously, acquisition of knowledge is an important aspect of human behavioural representation. Gonzalez & Dankel [197] note that knowledge acquisition is composed of *knowledge elicitation* and *knowledge representation* within a tool. How and where does one obtain relevant knowledge for modelling human behaviour? Researchers have postulated various ways to do this. Knowledge acquisition techniques can be grouped in three categories: *Manual, Automated* and *Learning* techniques.

Schreiber et al. [109] lists five types of knowledge acquisition techniques:

[2] HOOTL is human-out-of-the-loop simulation. It is a type of CGF that utilizes computer models of human performance to create virtual human agents.

[3] CoCoM is discussed later.

- Interviewing: unstructured, semi-structured and structured
- Protocol analysis: an analysis of the expert is carried out while he/she is actually solving the problem. An observation of what the expert does is noted either by video, audiotape, etc. The modelling engineer then extracts meaningful structures and rules from the transcripts of these records
- Laddering: Graphical representations are made in terms of the relationships between the problem being solved and the domain. The expert and modelling engineer jointly construct these graphs.
- Concept sorting: "a useful technique used to uncover the different ways an expert sees relationships between a fixed set of concepts" [109]
- Repertory grids: experts are presented with samples of the problem domain. The experts are then asked to choose a pair that is similar and one that is different. The reasons given by the expert for the difference between the three chosen samples are noted and become known as a construct. An example [109] is when attempting to know an astronomer's understanding of the planets, "if we present him with a set of planets, and he chooses Mercury and Venus as the similar pair, and Jupiter as the different planet. We would ask the expert (astronomer) for his reason for choosing Jupiter as different from the other two planets. We would use his answer as a construct. In this example 'size' would be a suitable construct. The remaining elements in the domain are rated on this construct." [109]

According to Gonzalez and Dankel [197], manual techniques usually involve interview sessions (question and answer) between an expert and the knowledge engineer (KE), studying instruction manuals and books, observing the expert. Gonzalez and

Dankel [197] list different approaches to interview sessions and observational techniques used during knowledge elicitation.

Automated techniques present a method where the experts' knowledge is captured automatically. These techniques could range from a simple query session between and expert and an intelligent system to a more complex system that captures knowledge by observing the expert perform actions in his domain [198]. Some automated tools and techniques are presented below.

CITKA [16, 17, 21], developed for acquiring knowledge about military tactics, involves a query session between a subject matter expert and the CITKA system. The main advantage of this approach is in the reduced need for human effort in the acquisition of knowledge and implementation of the acquired tactical knowledge.

Kim and Gil [19] show how to use existing knowledge acquisition methods towards building human behaviour models. They also show how these models can be improved with the development of an acquisition dialogue tool.

Chen and Chan [111] use the inferential modelling technique (IMT) for knowledge analysis during the process of knowledge acquisition. IMT is a template that organizes the chunks of knowledge usually embedded in data obtained from experts. Simon [112] presents an acquisition method composed of three steps. The first step involves exploiting the structure of existing documents within the organization. The last two steps of their method are cyclic, and include interviews with experts, prototype creation and tests.

ATTack (Acquisition Tool for Tactical Knowledge) developed by Henninger [126] based on the knowledge requirements of context-based reasoning, had a primary objective of reducing the time and decreasing the need for an expert in the acquisition of a knowledge base for CxBR representation [126]. ATTack made use of the Visual Interactive System for Task Analysis (VISTA) [4] to collect knowledge on objects of interest to the agent. ATTack is a pre-cursor of CITKA, having the same overall objectives.

Learning techniques are those that deduce knowledge for a given task from a variety of knowledge sources. Gonzalez and Dankel [197] define learning as "the improvement in the performance of a specific task (intellectual or physical) after previous exposure to that task or a related one" [97]. Typically, knowledge acquisition techniques involving learning usually come from examples (historical cases or hypothetical examples from experts). An example of a knowledge acquisition technique involving learning is *Redux*. Redux [18] is another automated approach for acquiring HBR knowledge. Its main focus is in the reduction of acquisition cost, validating and maintaining the knowledge used in HBR systems. This was achieved by allowing SMEs to use diagrams to specify behaviours in abstract scenarios [18]. The system analyzes and automatically generalizes from the scenarios presented by the expert.

Researchers at MIT [105] have developed a wearable platform that captures regular patterns in a persons' behaviour and forms a predictive model of his activities with them.

Sidani [58] captures expert behaviour by observing expert actions in a simulation. Sidani's method has the advantage of capturing both implicit and explicit knowledge. Gonzalez et al. [110] present a model called Template-based Interpretation (TBI) that

[4] VISTA was developed by Ahlers & Schnitzius and is a graphical tool to acquire knowledge for a CxBR system.

captures human behaviour by observing human actions in a simulation for the purpose of interpreting the person's intentions. Fernlund & Gonzalez [10, 138] also acquired expert knowledge by observation and were able to create tactical agents semi-automatically[5].

Fernandez-Breis, et al. [106] implement an approach combining natural language recognition techniques and knowledge acquisition that can extract knowledge from natural language texts. Kass & Finin [107] suggest techniques for implicitly acquiring knowledge about a user during a system's interaction with its user. They go further to postulate some rules governing the acquisition of this knowledge.

There are many more knowledge acquisition tools, for example, *Induction tool*, PLANET, ETS, and many others [197]. An underlying shortcoming of all these tools and techniques is their total dependence on expert knowledge. For manual approaches, the drawback is that the information presented by the expert or read in instruction manuals and books written by experts are thought to be excellent sources of knowledge. There are no known methods used in filling the 'gaps' left by experts or books and as such tactical models built from these acquired knowledge are always lacking. The same drawback applies to automated techniques. The SMEs' are limited in what they know and what they can do. Thus these automated techniques and tools have the same shortcomings. On the other hand some acquisition techniques that utilize learning attempt to break the SME knowledge limitation. Currently though, the learning techniques used always involve the presentation of examples. The same problem applies in cases where no examples exist or where the examples presented by the expert are faulty.

[5] Fernlund & Gonzalez achieved this by the use of a new methodology called GenCL which makes use of CxBR.

This research presents a method that attempts to eliminate the errors and inconsistencies that could exist in the knowledge acquired from SME. The limitations in SME knowledge are made unnoticeable by designing a model that learns from its actinos and mistakes. This is achieved by having the agent learn and enhance its behavior in a simulator, based on its experience. This experience is gained when the agent learns from its mistakes and successes when attempting to achieve its goals in a simulator.

1.4 A Brief Introduction to Agents

Models of human behaviors are best embodied in some form of simulated agents. These agents' sense and act within the environments in which they are situated. Russell and Norvig [25] discuss how agents should act as well as the different types of agents. Foner [26] describes an agent from the sociological point of view and introduces a prototype agent known as Julia that attempts to appear human. Franklin & Graesser [27] try to differentiate between an agent and a program. They furthermore tried to classify agents according to their properties. According to Russell and Norvig [25], an agent should be rational. When an agent does the right thing, it is said to be rational. However, a new question arises with this definition of rational - what is the right thing for an agent to do? There should be a way of evaluating an agents' performance either internally or externally. An agent should have some form of autonomy. In summarizing the structure of an agent, Russell and Norvig [25] note that an agent "is equal to the agents' architecture in addition to its program." A program is a "way of mapping the percepts of the agent to the actions it takes". The way the program implements this mapping is what brings about the different types of agents. Based on this, Russel and Norvig suggest four types of agent programs, namely: Simple reflex agents, Agents that keep track of the world, Goal-based agents, and Utility-based agents [25]. These are further described below.

1.4.1 Types of Agent Programs

1. A *simple reflex agent* is one that reacts to situations [25]. The program is written in a condition-action rule. The percepts are interpreted to represent the current state and a rule that matches this state is fired, thus producing some action associated with this rule. A CxBR agent can be said to at least have the qualities of a simple reflex agent.
2. *Agents that keep track of the world* have an internal state that is updated with percepts from the environment in a regular manner [25]. This allows the agents to be aware of and survive their often unpredictable environment. CxBR agents also can be said to have this quality.
3. *Goal-based agents* have some sort of goal information within their program [25]. For example, an agent with information about the current state of the "world" can make decisions on what actions to take. However, if a goal is included, the agent's decision could be based on how to reach that goal. In most cases the goal-based agent appears to be less efficient, but it is far more flexible. The *competing context concept*[6] *(CCC)* developed by Saeki & Gonzalez [28] and other CxBR models require goal-based agents [7].
4. The *utility-based agents* try to perform actions based on the value of that action in that state, towards the end goal. There is a direct mapping of a state to a number, and

[6] The competing context technique is an attempt to eliminate the need for hard-coding information in the contexts for the CxBR architecture. This technique is explained in section 1.4.5.5

this number describes how good or bad that state is. The value of a state relative to the goal is usually calculated based on some predefined functions.

In this section, an introduction to agents has been presented. As stated, most HBR agents, including CxBR's agents, have the qualities of all types of agents described above. Methods that utilize machine learning in the acquisition of knowledge also possess these qualities with the exception of the qualities of a utility-based agent. Lacking a utility-based attitude means that there are no mappings between 'rewards' or 'punishments' to actions in each state and thus no mappings between a goal state and rewards or punishments for being in that state. Agents that seek and acquire knowledge based on the utility of each state, including the goal state, are most desirable. This research seeks to establish a knowledge acquisition technique based on agents that include the properties of a utility-based agent using a CxBR framework. This would be achieved by the synergistic combination of CxBR with RL through the enhancement of a predefined human behavior model.

1.5 Introduction to Some Human Behavior Representation Paradigms

As noted in the previous sections, many modelling paradigms exist that attempt to optimally represent human behaviour. Each of these paradigms has its advantages and disadvantages. So far, there is no modelling paradigm that addresses all aspects of human behaviour. Some modelling paradigms are domain specific; for example, a paradigm that only models a soldier's behaviour in a battlefield situation, a captains' behaviour in a submarine, an automobile driver's behaviour on a freeway, and many others. In general, the modelling paradigms typically meet most of the requirements of a HBR model. Some researchers have defined models that combine existing modelling techniques to synergistically combine their strengths. Below is an introduction to the most common modelling techniques used in modelling human behaviour.

1.5.1 Cognitive Network of Tasks (COGNET)

COGNET is a framework developed by Zachary et al. [5, 113] that models human cognition and decision-making. COGNET meets the requirements of the cognitive analysis process as defined by [113]. According to Zachary et al. [113], the cognitive analysis process should represent the four main aspects of tactical decision making. These include, 1) real-time, 2) opportunistic, 3) multi-tasking and 4) situated in computer-based and verbal interactions. "It provides an integrated representation of knowledge, strategies, behavioural actions and problem solving skills that are used in specific domains to produce a powerful cognitive tool." [114]. The "COGNET framework is composed of a theoretical basis, its description language[7], its data collection, knowledge elicitation, analysis and the representation methods" [113].

COGNET is composed of a cognitive architecture and an internal knowledge base, (see Fig.1.1). The cognitive architecture has a specific structure with standardized operational principles. The internal knowledge is "a set of symbols on which it operates" and is organized in specific representational schemes [113]. Human information processing is "broken down into three parallel mechanisms – perception, cognition and motor activity". Perception includes sensation, which receives information from the

[7] The knowledge is usually represented using the COGNET description language

outside world and "stores it where it can be accessed by both the perceptual and cognitive mechanisms". This store is known as the extended working memory.

The cognitive process manipulates the information in the extended working memory by using some previously-acquired knowledge. The cognitive process doesn't operate directly on the perception of the outside world. It can modify the representation of the problem and also invoke actions via commands to the motor activity module. The motor activity module then manipulates the environment through physical instruments embedded in the system. The COGNET framework has been successfully applied to many domains including a vehicle tracking domain, here it was shown to have the ability to represent and predict attention-switching performance [113]. The framework was also applied to telephone operator services [114] and other complex domains. Fig. 1.1 illustrates the COGNET framework.

Although COGNET has been used to successfully model human behaviour, it suffers from the problem earlier identified in most human behaviour representation techniques, i.e. the over dependence on expert knowledge. The data collection and knowledge elicitation components of COGNET do not incorporate learning and as such the data collected is stale (not dynamic) in the sense that a model built with the COGNET framework, cannot be enhanced automatically.

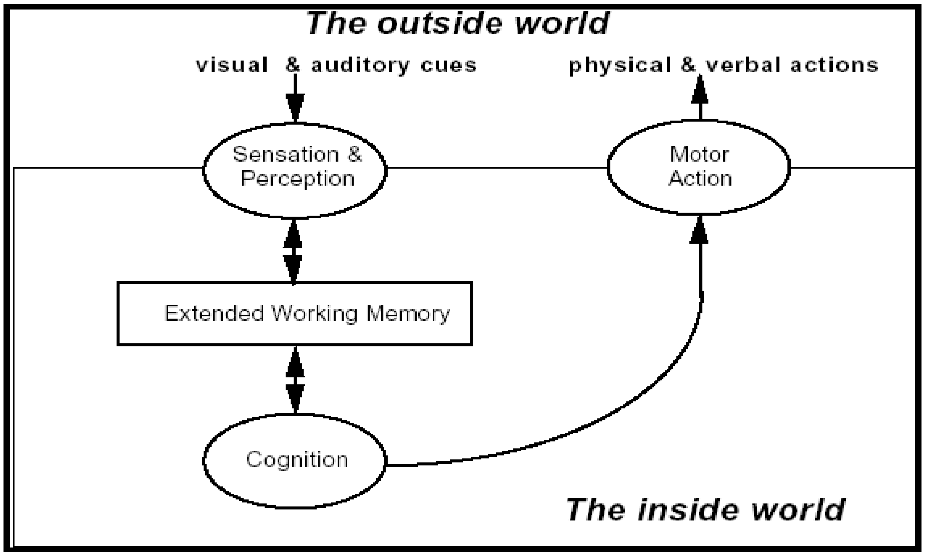


Figure 1.1 Conceptual View of COGNET Cognitive Architecture (Reproduced from Zachary et al. [113] without permission)

1.5.2 Atomic Components of Thought or Adaptive Character of Thought (ACT-R)

ACT-R is an Adaptive Character of Thought – Rational theory conceived by Anderson [101]. This theory is based on the premise that "complex cognition comes from the interaction of procedural and declarative knowledge" [101]. Procedural knowledge, represented by production rules, arise from the "simple encodings of transformations in the environment [101]". On the other hand, "declarative knowledge is represented by units called chunks" and it arises from "simple encodings of objects in the environment" [101].

ACT-R bases its foundations on the workings of the human cognitive process. This process has a large database of knowledge units (chunks and productions rules). In any given context, "the appropriate units are selected by an activation process based on the statistical information gathered on the environment." The ACT-R theory states that “the power of human cognition depends on the amount of knowledge encoded and the effective deployment of the encoded knowledge” [101].

The goal of the ACT-R theory is to provide the details to a claim made by Anderson [101] on the formation of intelligence. Anderson claims that “all that there is to intelligence is the simple accrual and tuning of many small units of knowledge that in total produce complex cognition. The whole is no more that the sum of its parts, but it has a lot of parts.”

There are three questions that address how the details of the claim are provided: 1) How do we represent the units of knowledge? 2) How do we acquire the units of knowledge? 3) And how do these units of knowledge get deployed in a cognitive system?

Declarative knowledge is represented in chunks. Chunks are “schema-like structures” [101] that have pointers that specify their category and encode their contents. “Procedural knowledge is represented by production rules”. These production rules usually act towards achieving a goal and sometimes create sub-goals in the process. These sub-goals establish “an abstract hierarchical structure on behavior” [101].

The second question revolves around how the units of knowledge are acquired. We need to know the origin of both the chunks and the production rules. The actions of production rules usually create chunks. The encoding of chunks are the origins of production rules. Anderson [101] notes that these definitions of the origins would cause circularity in the theory. Thus, he also suggests the creation of chunks from an independent source. The independent source is the encoding from the environment. During knowledge acquisition in ACT-R, chunks from the environment are encoded and inferences about the rules for their transformation involving examples of the problem are made.

The final question is, how do these knowledge units organize themselves in such a way that the right unit is chosen in a particular context? How do you identify the relevant knowledge for a particular situation quickly? Anderson [101] notes that this is a major problem that has dogged artificial intelligence (AI) systems, and in particular, expert systems. He notes that the power of expert systems lies in their knowledge base, i.e. the more knowledge available to an expert system, the more power it should have. The problem, however, is that with the growth of an expert systems knowledge base, the slower it executes, up to a point that it is no longer effective to use.

Anderson [101] developed a solution for this problem using his rational analysis method[8]. There are two parts to this solution. The first is an identification of the knowledge structures (chunks and productions) that most likely fit the current context. According to Anderson [101], there is a track record of general usefulness maintained by the mind and this is combined with "contextual appropriateness" for some inference about the knowledge to use in the current context. The second part is the identified knowledge structures determining the performance.

In summary, Anderson [101] states that "ACT-R implies that declarative knowledge is a direct mapping of things in our environment" while procedural knowledge is the direct mapping of the observed transformations. These two types of knowledge are combined and applied based on the statistical knowledge of the environment.

[8] According to Anderson [101], rational analysis theory states that "knowledge is made available according to its odds of being used in a particular context." These odds are calculated by an implicit performance of a Bayesian inference during the activation process.

There are many versions of the ACT theory as well as implementation of them. Lebiere [130] provides a tutorial on ACT-R version 5.0 and provides a history of the ACT theory.

Anderson et al. [102] present some additions to the ACT-R architecture to enhance the integration of various modules into a single problem solving unit. They showed how this integration performed better than previously.

1.5.3 State Operator And Result (SOAR)

In SOAR, goals and sub-goals are generated and plans are created and implemented on how to reach these goals. SOAR was developed by Laird et al. [33]. Until the goals are attained, the plans for achieving those goals remain active. When a new situation arises, new goals are generated and new plans on achieving these new goals are created and implemented. The cycle continues until there are no more goals to achieve. The goals and plans are implemented using the rule-based paradigm. According to Laird et al. [103] "the design of soar is based on the hypothesis that all deliberate goal-oriented behavior can be cast as the selection and application of operators to a state".

There are two types of memories available in SOAR, the working memory that holds the current situation, the results from intermediate inference, active goals and active operators, and the long-term memory that describes how to respond to the various situations in the working memory [103]. In solving a problem, the steps involved include: proposal of candidate operators, the comparison of candidate operators, the selection of a single operator from the list of proposed candidate operators, and the application of the selected candidate operator. This is shown in Figure 1.2. Figure 1.3 shows the SOAR architecture.

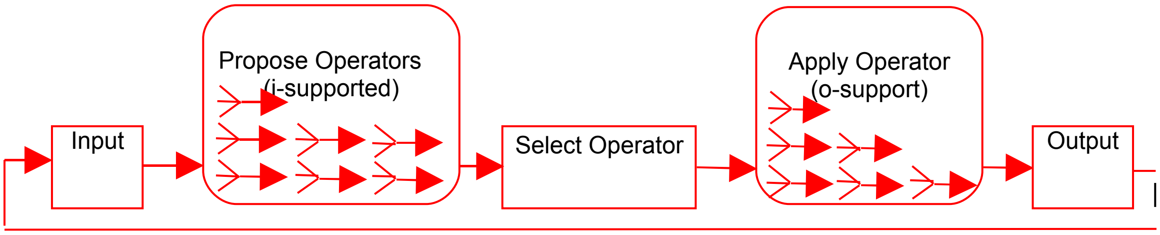


Figure 1.2 SOAR Decision Cycle, (Reproduced from [166] without permission)

In SOAR, learning is achieved through a mechanism known as *chunking*. Chunking occurs when an operator impasse is resolved and SOAR summarizes and generalizes the processing that led to that sub-state. According to Ritter et al. [115], the development of SOAR was based on the combination of three main elements, "the heuristic search approach of knowledge-lean and difficult tasks", "the procedural view of routine problem solving", and "a symbolic theory of bottom-up learning designed to produce the power law of learning." Ritter et al. [115] compare the similarities and differences between SOAR and ACT-R architectures for modeling behavior. They note that the "limitations on SOAR's theoretical assumptions originate from the general characteristics of intelligent agents, rather than from a detailed behavioral representation"[115]. Therefore, "SOAR is more biased towards performance than ACT-R because its background is AI-based, while ACT-R is based upon cognitive psychology"[115].

Young and Lewis [116] examine the contributions made by SOAR's approach to the issues pertinent with the working memory. They show how a cognitive system can handle complex tasks that require large quantities of information by utilizing long-term memory in conjunction with the external environment if there is some constraint on the capacity of the working memory.

Several researchers have successfully combined the SOAR approach with other cognitive models or other learning paradigms amongst which include the EPIC-SOAR model for a simplified enroute air traffic control task [117] and SOAR-RL [118]. SOAR-RL [118] is a modification to the SOAR architecture that provides learning opportunities for an agent from statistics of its successes and failures in the selection of an operator. It is reinforcement learning (RL) embedded in the SOAR architecture.

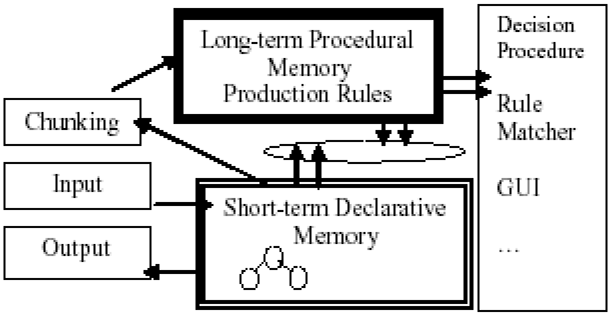

Long-term Procedural Memory Production Rules
Chunking
Input
Output
Short-term Declarative Memory
Decision Procedure
Rule Matcher
GUI
...

Figure 1.3 SOAR Architecture (Nason and Laird [118] reprinted without permission)

1.6 Organization of Dissertation

This dissertation is organized in the following order. Chapter 1 provides a summary of the research done and the background information of some human behavior representation paradigms. Chapter 2 describes some techniques that use contexts to represent human behavior. Chapter 3 clearly defines the problem being addressed.

Chapter 4 contains some machine learning techniques that could be used in solving the problem. In Chapter 5, the conceptual approach to a new methodology is described that integrates CxBR and RL. Chapter 6 describes the design of a prototype and the various experiments while chapter 7 contains the results from the experiments and a comprehensive evaluation of these results. Chapter 8 is a conclusion of the research work with a summary of the work, recommendations and future research work that could be an offspring from this work.

1.7 Summary

This chapter introduces the reader to the problem being investigated. Background information on various human behavioral modeling techniques is presented. Also presented in this chapter are the ways in which knowledge is acquired and represented by the various modeling techniques. The limitations of the various techniques used to represent human behavior are highlighted with an emphasis on the most prevalent limitation, i.e., the total dependence on SME knowledge. Finally the outline of the dissertation and chapter summary were presented.

CHAPTER 2: HUMAN BEHAVIOR REPRESENTATION THROUGH CONTEXTS

2.1 Contexts

Humans employ contextual reasoning in everyday decision-making. However, one may wonder what exactly is a context? In simple terms it means, “That which surrounds, and gives meaning to, something else” [1]. Nevertheless, many researchers in the field of contexts have different ideas and meanings of what contexts are. Sowa [37] states that “the word context has been used with a variety of conflicting meanings in linguistics”, Sowa [37] goes on to list two major perceptions of the word contexts as derived from the dictionary. There are:

- The basic meaning of Context is some text that surrounds a word being used in a sentence or some phrase of interest. Sowa suggests this to be a section of linguistic text.
- The derived meaning of Context is in a non-linguistic situation, it includes some topic of interest.

Sowa, goes on to say “Context may refer to the text, to the information contained in the text, to the thing that the information is about or to the possible uses of the text, the information, or the thing itself” Sowa [37].

Kokinov [34] states that contexts can be viewed as “a set of internal or mental representations and operations” or “a set of environmental elements”. Kokinov [93] goes on further to define context as “… the set of all entities that influence human (or system’s) cognitive behavior on a particular occasion”. According to Kokinov [34], the way AI researchers’ model contexts are different from the way contexts are modeled and viewed by researchers in psychology. Usually, “the AI approach to contextual reasoning can be viewed as navigation between and within context boxes”[34]. The boxes are predefined and the only issues for the AI developer are how to represent the individual box, how to recognize that the box has to be changed and how a new box is chosen amongst the various boxes [34]. Kokinov [34] states that the issue of constructing a new context on the fly is never addressed in AI researches. One of the focuses of this research is to address this limitation – creation of context on the fly.

On the other hand, according to Kokinov [34], “when psychologists study context effects, they do not think of changing the goals or beliefs of the subject”. Kokinov [34] mentions intentionality, controllability, awareness and efficiency as independent aspects of the automaticity of any cognitive process. What Kokinov concluded from the “review of the psychological studies on context is that context usually has an unconscious and unintended influence on people’s behavior and that this happens all the time and is triggered by all sorts of incidental elements of the environment but also by the previous memory states”[34]. Kokinov notes that “it is very important that the previous memory state produce context effects, since the context effect maintains the continuity of the cognitive processes and prevents human thoughts from continuously running in leaps”[34]. The previous memory states also ensure efficiency because “they restrict the set of all possible interpretations, inferences, searches, etc., to the set of relevant ones” [34]. Kokinov & Yoveva [87] note the effects of contexts on problem solving. They show the effect of ‘near’ and ‘far’ contexts in experiments that contain illustrations of the problem being solved, illustrations relating to the problem being solved and also illustrations not relating to the problem being solved. Kokinov [2, 34, 90, 91, 93] went on further to suggest a way of dynamically changing contexts. Kokinov proposes in [93] “a dynamic theory of context that considers contexts as the complete set of entities that can influence a human’s cognitive behavior in any given occasion”[93]. Contexts are “thought of as the dynamic fuzzy set of all associatively relevant memory elements (mental representations or mental operations) at a particular instant of time” [93]. The main principles of Kokinovs’ dynamic theory of contexts are [34]:

- “Context only refers to the state of mind of an entity and not to the environment in which that entity exists”
- At any given time, the specific distributions of priorities of all mental representations and operations correspond to the context
- The associative relevance of mental elements measure the priorities
- “associative relevance is graded and computed automatically and in parallel to the reasoning process”
- “There are no clear cut boundaries between the set of priority elements because context is dynamic”

Kokinov [34, 90, 93] presents the DUAL[9] architecture that utilizes the dynamic theory of contexts.

Zibetti [35] discusses the role of contexts in interpreting and understanding perceived events as actions carried out by other people. A definition of context is based on the state of the system at the processed time and also the temporal definition. Some interesting examples where given by Zibetti [35] to portray what action is being perceived by onlookers in different scenarios and finally suggests a method called

[9] The DUAL architecture is described in detail in section 2.

C.A.D.S [96] (Categorization et Assignation Dynamique de Signification) [96]. C.A.D.S is a model that attributes meaning to situations through the process of categorization.

As stated earlier, there are many definitions of contexts. Brezillon [40, 83] defines contexts “as a collection of relevant conditions and surrounding influences that make a situation unique and comprehensible”. Three types of contexts as proposed by Brezillon and Pomerol, are enumerated in [83] as external knowledge, contextual knowledge and proceduralized context. The external knowledge is the part of context that isn’t relevant to a step in the decision-making process of a task. The “contextual knowledge is the part of the context that is directly relevant to the step of the decision making process in the task”[83]. Brezillon [40] breaks the contextual knowledge further into the proceduralized context, which is prevalent at any given step of the decision-making process of a task. Based on his definition of a context, Brezillon conceived the notion of Contextual Graphs, “that allow any given problem for operational processes to be represented in contexts by taking into account the working environment” [83].[10]

Turner defines contexts as “any identifiable configuration of environmental, mission-related, and agent-related features that has predictive power for behavior” [36]. Turner [36] conceived the Context-mediated behavior (CMB) paradigm which ensures an agent behaves appropriately in any given context. More on this later.

2.2 Representing Human Behavior through Contexts

Successful attempts have been made to model tactical behaviours through the use of contexts [7, 36]. Based on the flexibility of contexts, and the many definitions of it, several researchers [36, 38, 39, 40] have proposed methods that rely on contexts to build

[10] A more detailed explanation on contextual graphs can be found on section 1.5.4.3 & in [83]

agents that exhibit tactical behaviours. Although there is no universally-accepted best "modelling paradigm", each modelling paradigm proposed by the various researchers have advantages and disadvantages. These are briefly described below.

2.2.1 Context-Mediated Behavior

Context-mediated behavior for intelligent agents was conceived by Turner [36, 44]. It is based on the premise that an intelligent agent should effortlessly recognize the contexts in which it is in and act appropriately in accordance to the explicit knowledge about the context available to it. He notes that there is "no such thing as context-free appropriate behavior" and that an intelligent agent should take context into account automatically as humans and animals do.

Turner [36] lists four desirable properties to his approach on context-sensitive behavior, the first is to make sure it is efficient, a change in context should be immediately recognized and the appropriate context for the new situation immediately activated. Secondly, it should be automatic, i.e. after the new situation is recognized, the change in context should occur automatically. Thirdly, CMB helps in the agents' perception and understanding of the environment. Finally, explicitly representing contexts provide an opening for the contextual knowledge of agents to be adjusted from their experience.

Turner [36] identifies the aspects of an agent's behavior that can be affected by context as:

1. Understanding the situation: before any decision is made by the agent on how to behave, it should understand the current context. Its knowledge about its current

context would aid in answering some questions pertinent to the behavior it exhibits when in that context. These questions include

   a. What predicted features of the context that are not yet visible in the current situation?
   b. What are the unusual features of the current situation?
   c. Are there multiple meanings to known concepts in the current context?
   d. How would the interpretation of sensor data be achieved in this context?

2. The behavior should be automatically modulated to fit the context: This is an operation that is implicit in humans; an example provided by Turner [36] is that when a person enters a library, the person automatically reduces the tone of his / her voice; when a movie ends a person automatically starts leaving the theater, this becomes his immediate goal. Turner [36] states that once a context is recognized, an appropriate behavior for that context should be exhibited by the agent without any reasoning effort on the agents' part.
3. Handling of unanticipated events: the handling of unanticipated vents should occur effortlessly as soon as the context is recognized. Knowledge about how to handle events should be available, as this knowledge would help the agent:
   a. Detect the event
   b. Evaluate the event
   c. Respond to the event
4. Deciding on what to focus attention: the decision on the goal to focus attention is context-dependent. Attention should be focused on the goals of the current context, for example if a person is hungry and going to buy some food, his

attention should be on purchasing the food and eating. However, if on his way to the food store his car experiences a flat tire; his immediate goal would change to address this context. The focus of attention would switch to fixing the car.

5. Selection of actions for achieving goals: as soon as a decision is made about what goal to focus attention, the actions to achieve this goal must be selected. This selection process is context-dependent, for example if a person remembers that a bill is due immediately and has to pay that bill immediately, knowing (s)he could pay by phone or the internet, if the person is driving, (s)he would select the action to call and pay by phone, if (s)he is at home, (s)he could choose to either pay over the internet or pay by phone. Turner [36] states that the knowledge of the current context available to the agent should allow an effortless selection of the actions based on the context.
6. Selection of strategies for problem-solving: various strategies exist for solving the same problem, depending on the context. An example provided by Turner [36] is the difference in the way a medical student and an experienced doctor perform physical examinations on a patient. A medical student would follow a step by step, pre-set procedure, whereas an experienced physician can skip parts of the process he knows from his experience are not necessary for the particular case.

With context having an effect on these aspects of behavior, Turner postulated a Context-mediated behavior (CMB) as “a mechanism for ensuring that an agent behaves appropriately for its context.” There is an explicit representation of an agent’s knowledge for each context and CMB makes sure for each context, the right knowledge is provided. Figure 2.1 illustrates the Context-Mediated Behavior process.

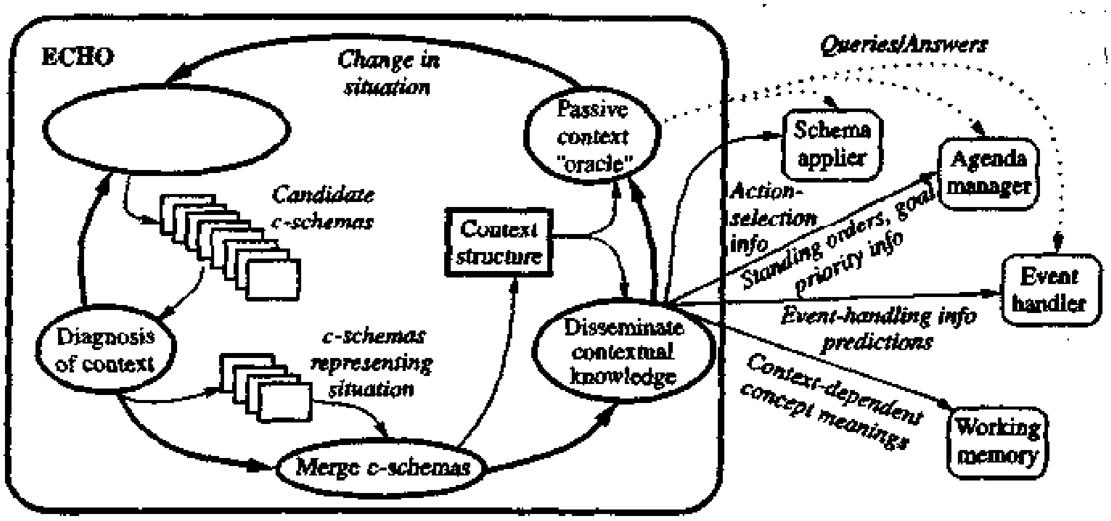


Figure 2.1 The Context-Mediated Behavior process (reprinted without permission from [36])

The CMB process is made up of interactions between ECHO (embedded context-handling object) and other modules. The long-term memory is searched for contextual schemas (c-schemas) that could potentially identify the new situation. Turner [36, 44] calls this "*evocation*". This process of evocation identifies a few candidate c-schemas. A diagnosis of the situation as a representation of one or more contexts is carried out by ECHO, resulting in the creation of a "context structure". This context structure represents the current context. In some situations, multiple c-schemas are needed to identify the situation correctly. These c-schemas are merged and the knowledge in them is sent to the reasoning modules of the agent.

In CMB, contextual knowledge is stored in c-schemas. The knowledge acquisition process is achieved by interactions with domain experts. This interaction could follow any of the methods described in the section on knowledge acquisition. Turner [36] hoped that the agent would eventually learn c-schemas from its experience but this was never implemented. Contexts are identified through diagnosis, the attributes of the current situation, knowledge about known situation and relationships between them. These provide the information for diagnosing the current situation as represented by one context or a combination of contexts. Each c-schema contains the contextual knowledge about the situation as well as the relationship between it and other c-schema. When a situation that has never been experienced before occurs, “a c-schema representing a similar context would be merged to form a correct representation of the new context” [36].

The approach used by Turner [36] in transitioning between contexts, is to provide information that would trigger a context change within the context definition. Turner [36] notes that learning the events that trigger a context change may be difficult. He suggests it as a future research topic.

Some successful applications of the CMB technique include the control of an autonomous underwater vehicle (AUV). Although not related to Turner’s [36] CMB, a pedestrian flow model (PEDFLOW) has been developed using a context-mediated behavior technique by Kukla, et al. [131]

A drawback of the CMB technique is the lack of agent learning. A model built with the CMB technique cannot be enhanced automatically during the models interaction with its environment. This is because the knowledge acquired is ‘static’. Knowledge acquired for a CMB model is totally dependent on a SME.

2.2.2 Contextual Graphs

Brezillon [83, 132] describes a contextual graph (CxG) as “an acyclic graph with a single source, a single output (sink), and a serial-parallel organization of nodes connected by oriented arcs”. The nodes in the graph represent the actions, the “contextual and recombination nodes”, the sub-graphs (activities) and a parallel grouping. There is always an end to the algorithm in a contextual graph. Contextual graphs present the reasoning process understood by operators because the modeling of the operators’ activities is possible through the sub-graphs. “An action is an executable method; an activity is a complex action with different elements.” [83] An ordered sequence of the elements of a contextual graph from the input to the output is known as a path. A practice is the sequence of actions in a path.

Brezillon [132] notes that “a proceduralized context is an ordered sequence of contextual-knowledge pieces and their values.” From this definition, the context of any action is defined by its proceduralized context and the contextual knowledge.

A great attribute of CxGs’ is the ease of introducing new practices. The generation of a new practice consists of the application of a few changes to an existing practice or contextual nodes. The knowledge of an existing practice used by an operator and the possibility of acquiring it when needed are attributes of CxG systems.

The building of the proceduralized context from contextual knowledge is usually based on communication between members in a community of practice, irrespective of their domain of origin. Usually when people interact, a piece of knowledge, i.e. the focus of attention of each person, is combined to create an interaction context. The piece of knowledge provided by each person is taken from their contextual knowledge. The people combine and structure this knowledge into a shared segment of knowledge. Additions to this shared knowledge are possible based on the request of other(s) in the group. The addition of knowledge to this shared knowledge is refered to as the progressive building of proceduralized context. If this shared knowledge is finally accepted by all parties interacting, it is integrated into a knowledge structure agreed upon by all parties (*proceduralized context)*. This proceduralized context is then moved to the shared contextual knowledge of everyone when it is no longer the focus of attention. “The proceduralized context, therefore, contains all the pieces of knowledge assembled and accepted by all persons that interacted”[132]. It represents a “functional knowledge or causal and consequential reasoning. This newly-created contextual knowledge (previously proceduralized knowledge) can be utilized later as either a whole or part of another contextual knowledge to be integrated into a new proceduralized context”[132].

Brezillon [132] argues that this is why the more experience a person has, the more structured the knowledge available to the person is.

Each action in a CxG is usually associated with some fixed and static context. The dynamic nature of context is achieved at the practice level. The evolution of the contextual knowledge and procedural knowledge during the application of a practice account for the dynamic nature of contexts.

There are some successful applications of CxGs, among which include an incident management for a subway line Brezillon et al. [133]. A prototype software that exploits the concepts of a CxG has also been developed [83].

Although Brezillon claims CxGs can learn, it is the opinion of this author that the ‘learning’ mechanism in CxGs is not fully developed and as such learning doesn’t actually occur. The lack of learning is a short coming of CxGs. A model built using CxG cannot be enhanced automatically during the models interaction with its environment. The knowledge used in building CxG models is also totally dependent on an expert.

2.2.3 Context-Based Reasoning

Context-Based Reasoning (CxBR) by Gonzalez and Ahlers [7, 39], provides intelligence to an agent by controlling its actions in a real or simulated environment. It is based on some guiding principles that pertain to the way humans think in their everyday activities. The basic ideas from which CxBR was created are [41]:

- “In any given situation, tactical experts are proficient at a task by identifying and dealing with only the key features of that situation.” [41] An example would be if an automobile is taken to an auto mechanic with water leaking from underneath the radiator, an expert auto mechanic wouldn’t bother examining its battery or ignition system. He/she automatically recognises the key feature of the situation - water leaking from the radiator and proceeds directly to the radiator.
- “The numbers of things that can realistically happen in any given situation are limited” [9]. A popular example given by Gonzalez et al [9] is that it is highly unlikely for a tire blow-out to occur while a car is waiting in a traffic light. As such, an agent wouldn’t consider a tire blow out event when in a traffic light situation.
- “When faced with a new situation, the present course of action would be altered accordingly to deal with the present situation” [9]. An example would be when driving to work from home; the usual plan of action could be to go from ones’ driveway to a suburban street, to a freeway to a city street and then to the parking lot of ones’ office. If a new situation occurs, for example a tire blow out or if the road is blocked by construction or an accident, a new course of action would be taken to achieve the overall goal of getting to the office.

CxBR is composed of a hierarchy of contexts. At the top level is the *mission context* that defines the overall goals and mission of the agent. Then there are one or more *major contexts*, and below that, *sub-contexts*, *sub-sub-contexts*, etc. Controlling the agent to achieve its goals is the main objective of CxBR. Figure 2.2 shows a block diagram of a context.

CxBR is action-based rather than goal-based, sub-goals are considered within a context but only implicitly.

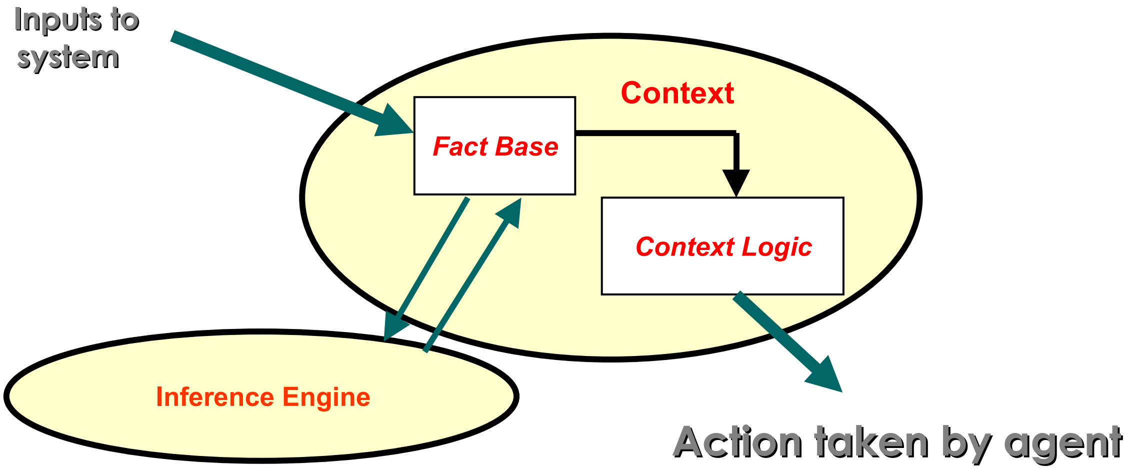


Figure 2.2 Block Diagram of a Context (Reprinted without permission from Stensrud et al. [89])

The mission context defines the objectives and the constraints of the agent. The goals the agent has to achieve are listed, as are the constraints imposed on the mission. For example, in a mission to drive to work, a goal can be getting to work on time and a constraint on this mission could be to avoid getting a speeding ticket, hitting a pedestrian, etc. At times, the constraints placed on a mission call for the agent to act intelligently and smartly to circumvent or satisfy the constraints. In the example given, if the agent has to get to work on time, and it leaves home late and goes through a freeway, because it has a constraint of not getting speeding tickets, it has to maintain the speed limit specified for that freeway and as such might not achieve its goal. The agent must then intelligently manoeuvre its way across different available shorter routes, taking into consideration its overall goal of getting to work on time.

Beneath the mission context level is the *major context*. A major context contains actions performed by the agent while in that context. These actions are based on the feedback received from the environment as to the agents position in the "world" (it's environment). An example of a major context could be driving on a freeway. While driving on a freeway, the actions performed by an agent would be different from those performed by the same agent when driving in a city context. At any point in time, there must be one and only one major context in control of the agent. This major context is referred to as the *active major context*. Major contexts are mutually exclusive of each other. *Sub-Contexts* are used to represent actions not directly critical to reaching the mission's objectives; "they are usually of short durations and are called by one or more major contexts"[41]. Figure 2.3 shows the hierarchical structure of CxBR.

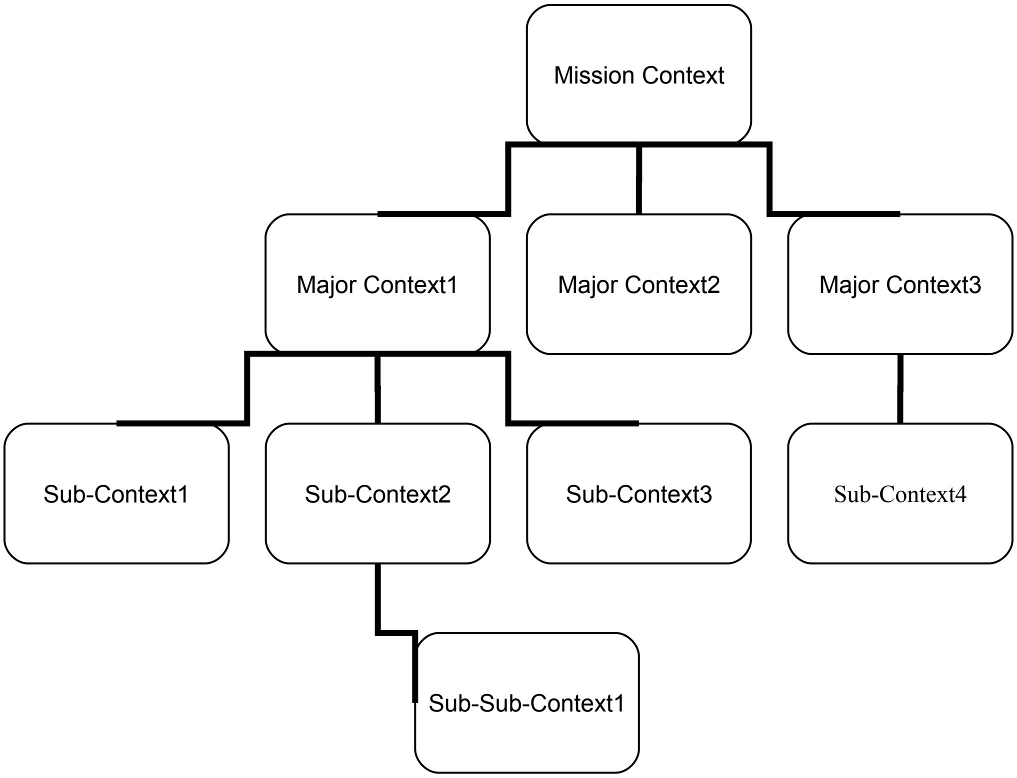


Figure 2.3 Hierarchical Structure of CxBR

As the agent performs actions on the environment, the environment changes and a search through the transition rules within contexts is done to recognise any evolving situation. Once a situation change is recognised, the context that addresses the new situation becomes the active context and takes control of the agent until a change in situation occurs again. This process of situational awareness, action on the environment and context transition occurs continually until the agent achieves its goal or fails to do so. Figure 2.4 shows the diagram of a CxBR model.

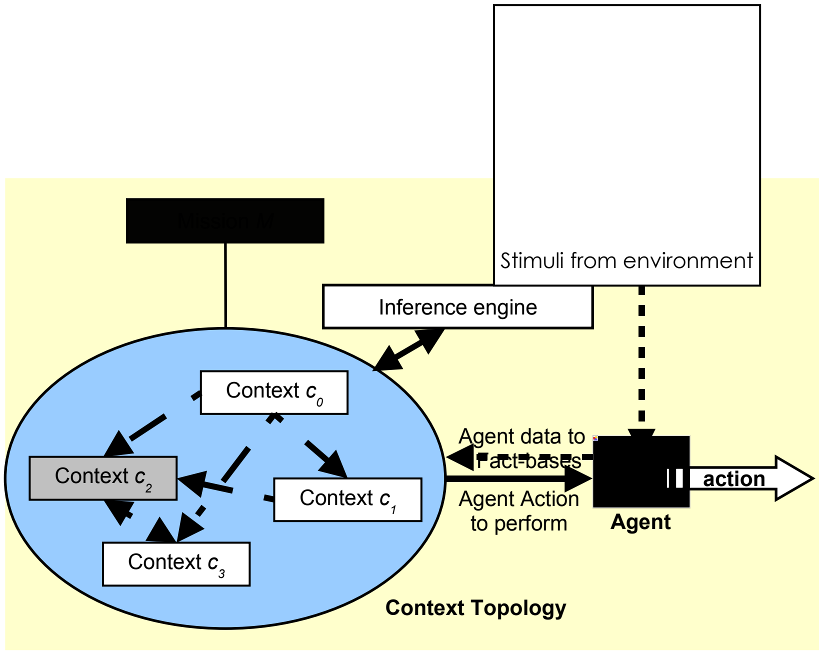


Figure 2.4 Diagram of a CxBR model. The dashed lines represent valid context-transition pairs while the solid lines indicate either inputs or commands. C_2 is currently the active Major Context (Reprinted without permission from Stensrud et al. [89])

Before using the CxBR paradigm, a detailed knowledge of the environment must be available. Also, there must be subject matter experts (SME) in the domain of interest before model development can be done. The knowledge provided by the SME is acquired by some means and modelled appropriately for the problem at hand. The modelling of the problem usually involves defining and creating *context boundaries*[11]. After the definition and creation of the context boundaries, the actions prescribed by the SME are hard-coded to each context. The manner in which the context transition should occur is also hard-coded within a context when a new situation arises. Norlander [124] built a framework for implementing CxBR agents in simulations.

[11] Context boundaries are the definitions of the identification of a situation and all allowable actions in that situation.

The contexts are "hard-coded" based on the knowledge supplied by an expert. This leaves little room for flexibility and learning. As such, the agent may act irrationally when faced with unknown situations. In general, the agents' actions or knowledge are constrained by those expert that provide the knowledge for the contexts, and the agent doesn't learn.

Attempts have been made to introduce flexibility to the CxBR paradigm with respect to the context transition process. Gonzalez and Saeki [42, 43, 28] introduced the *competing context concept* in which the context transitions defined in the contexts are not hard-coded, but rather allow eligible contexts to compete amongst themselves for the right to become activated. A *time-warp simulation* is carried out to determine the context to make active. It selects a context at random when no clear 'winner' exists between the competing contexts. While the competing context achieved its purpose, it doesn't learn. If a wrong context or action is chosen, the agent doesn't learn to not choose it again.

Recently, Fernlund and Gonzalez [10] developed an approach that automatically builds contexts by observing human actions. Although their approach achieved its purpose of learning through observation, it lacks the capabilities of experiential learning. If the observed expert behaves badly, so will the agent, and this might affect achieving the agents' goal. Learning from observation improves knowledge acquisition, but does not break the SME limitations, as one must still be observed.

The Context-Mediated Behaviour paradigm described in the previous section is conceptually similar to CxBR with some differences: 1) Instead of having the compatible contexts listed within a context and competing amongst contexts listed as compatible[12], in Context-Mediated Behaviour (CMB), a diagnosis is carried out on the contexts. 2) CMB

[12] This occurs in the competing context concept extension of CxBR

centralizes the management of contexts whereas CxBR distributes the management to the various contexts. 3) In CMB, contexts are merged to form new contexts, a feature that isn't available in CxBR. 4) CMB does not use a fact base, whereas CxBR uses the global fact base and the local fact base as working memories.

A comparison of Contextual Graphs and CxBR was carried out by Lorins et al. [92] to highlight the similarities and differences between both paradigms in terms of context representation, contextual change / movement, knowledge acquisition, etc. He concludes that more exploration on the advancements and their implementation is needed to have a complete comparison based on the above metrics.

2.2.3.1 Components of CxBR

The following components are an integral part of the CxBR architecture:

1. Contexts
2. Sentinel / Transition rules
3. Local fact base
4. Global fact base
5. Environment
6. Inference Engine
7. Agents

I Contexts

Contexts can direct the actions of the agent. The required responses to environmental stimuli are stored in contexts. They hold the transition rules as well as all actions to be undertaken by the agent. A typical mission would contain many contexts. The relationship between a context and the actions of a context is one-to-many, meaning one context can contain many actions. In the current CxBR architecture, the actions defined in a context are pre-programmed.

Another part of the context is the transition rules. The transition rules contain all transition definitions between the existing context and other compatible contexts. In the current CxBR architecture these rules are neither learnt nor updated during the course of the simulation as new information is introduced to the agent. The coding of the transition rules is pre-programmed.

II Sentinel or Transition Rules

This is the part of the system that alerts the agent when a change in situation occurs. It also initiates a transition to a new context based on the defined rules. The sentinel or transition rules are embedded in a context and are activated periodically or every simulation cycle. As these rules fire (are activated), information about the current state of the agent is obtained from the calculations, inferences and deductions that occur within the inference engine.

III Local Fact Base

The local fact base is part of the agent architecture. It stores information about the immediate environment, actions available to the agent (as defined in contexts). The local fact base acts as a working memory. This information isn't shared, and as such can be accessed only by the agent. Typically, after the inference engine identifies the appropriate context for the current situation, it passes this information to the agent, and this is stored in the local fact base. The local fact base reflects things about the environment that are known only by that agent.

IV Global Fact Base

The global fact base contains all information on the environment that is known to all agents in the environment, for example time of day, weather, etc. It can also be said to be working memory to all agents. The global fact base has a direct link with the environment and the agent. Norlander [124] places the global fact base in the agent (autonomous intelligent platform). For purposes of this research, the global fact base was an entity of its own, detached from the agent. As soon as the state of the environment changes, it is reported to the global fact base along with related information on why a change occurred (i.e. what caused the change – the action executed). Information from the agents' action is passed on to the global fact base.

V Environment

The environment is a representation of the world and all that will affect the agents' behavior in that world. The agents' actions in the environment influence the events that occur in the world. Some events in the environment occur irrespective of the action taken by the agent, for example a traffic light turning red or an antelope running across the road. As most events occur [13]it is the duty of the agent to learn how to identify these events and their characteristics. For each event, the agent has a choice of carrying out an

[13] Events that occur randomly to some people might be argued to occur at a particular frequency by others. Finding the frequency of occurrence or patterns / characteristics of these events or states preceding them is usually difficult.

action or not. Information on the various states of the agent and events in the environment are constantly being updated in the global fact base. Events that are related to the simulation environment of the agent are updated in the global fact base. Some of these events might not be visible to the agent at all times.

VI Inference Engine

The inference engine as defined by Norlander [124] is used for pattern matching – to match patterns with facts in the various fact-bases. It is also used to assert and retract facts as the simulation progresses. During the course of a typical simulation, the environment sends information to the global fact base, patterns within this information are processed (matched) with the transition rules in the defined contexts by the inference engine. As soon as a match is found, the actions within the context are performed by the agent and these are asserted in the fact bases. This cycle continues until the mission goal is achieved or it is otherwise determined that it cannot be achieved (agent killed, for example).

VII Agent

The agents in a CxBR model are usually unintelligent because they possess no knowledge of what to do in their environment without the direction and control of contexts. The contexts make the agents intelligent. They are the object of attention in a CxBR simulation, because they perform actions, effect a change of state and are dynamic. The agent contains the local fact base, the mission goal, an inference engine, a clock and default context. The local fact base contains the information essential to the agent as it tries to achieve its goal. It is akin to the working memory of the agent. Information is constantly being updated when something new about the environment is perceived by the agent.

2.3.2.1 Formalization of CxBR

Stensrud et. al. [89] formalized CxBR. This is reproduced verbatim and presented below without permission:

The mission goal is a Boolean function g of a set of environmental **E** and physical **P** conditions at any given instant.

$$\text{Goal} = g(\mathbf{E}(t_0), \mathbf{P}(t_0)) \qquad \text{(i)}$$

Constraints on a mission M, is the union of the set of physical, environmental and scenario-specific constraints ($\mathbf{co}_p$, $\mathbf{co}_e$, $\mathbf{co}_s$) placed on the agent.

$$\text{Constraints} = \{\mathbf{co}_p, \mathbf{co}_e, \mathbf{co}_s\} \qquad \text{(ii)}$$

The mission assigns a set of contexts **C** and *context-transition pairs* that pick specific context switches allowed during the scenario. This combination, defines the high-level behavior of the agent. An example presented by [89] is to consider the set of contexts:

$$\mathbf{C} = \{\mathbf{C}_1, \mathbf{C}_2, \mathbf{C}_3, \ldots\ldots., \mathbf{C}_n\} \qquad \text{(iii)}$$

if the Mission M, has a context-transition pair $<\mathbf{C}_1, \mathbf{C}_4>$ assigned to the Agent **A**, it means it is possible to transition to context $\mathbf{C}_4$ from context $\mathbf{C}_1$ at a given time-step t_k, when the agent is operating in context $\mathbf{C}_1$

The context topology of mission M is made up of the set of contexts **C**, the set of context-transition pairs **T** the Default Major Context (**DMC**) and the *Universal Sentinel Rules* for the scenario [89].

$$\text{Context-Topology} = <\mathbf{C}, \mathbf{T}, \mathbf{DMC}, \text{Universal-sentinel-rules}> \qquad \text{(iv)}$$

Combining equations i – iv, the definition of a mission is:

$$M = <\text{Goal}_{M}, \mathit{Constraints}_{M}, \mathit{Context\text{-}Topology}_{M}> \qquad \text{(v)}$$

The context logic for a major context is made up of the control functions (cf’s), knowledge and action rules [89]. The set of functions that control an agent in any given context $\mathbf{CF}_{\mathbf{MC}}$ is defined as follows [89]:

$$\mathbf{CF}_{\mathbf{MC}} = \{cf_1, cf_2, \ldots., cf_n\} \qquad \text{(vi)}$$

The set of *action rules* (ar’s) for any given context is $\mathbf{AR}_{\mathbf{MC}}$. Typically, action rules can activate Sub-Contexts, can utilize facts in the fact bases to carry out actions, etc. $\mathbf{AR}_{\mathbf{MC}}$ are defined as follows:

$$\mathbf{AR}_{\mathbf{MC}} = \{ar_1, ar_2, ar_3, \ldots\ldots., ar_k\} \qquad \text{(vii)}$$

“The knowledge in a Major Context is the set of frames or classes whose attributes and methods are essential elements of the tactical knowledge required to successfully navigate the current situation”[89]. Stensrud et. al. [89] refers to this as *Knowledge Frames* ($\mathbf{KF}_{\mathbf{MC}}$)

The context-logic that controls the actions of an agent in any given context is:

$$\text{Context-logic} = < \mathbf{CF}_{\mathbf{MC}}, \mathbf{AR}_{\mathbf{MC}}, \mathbf{KF}_{\mathbf{MC}}> \qquad \text{(viii)}$$

Sub-Contexts are called upon by Major Contexts. They are activated when the calling *action rule ar* is fired. Their inputs are the *action rules*, and their output is the achievement of their sub-goal. Control mustn’t be returned to the calling Major Context and any Major Context can call any sub-context.

$$\text{(sub-goal)SubContext}_{m} = f_0\, (\mathbf{AR}_{\mathbf{MCi}})$$

Transition Sentinel Rules are defined as "the rules that contain the conditions under which a Major Context transition is required" [89]. For example, according to [89], if a mission has a context-transition pair of Major Context C_k to C_n, C_k will have a sentinel rule that constantly monitors the environment for the satisfaction of conditions needed to transition to C_n. Each Context C_i has a set **S** of transition criteria. For a given context pair, there can be multiple transition rules, let $\mathbf{S}_{ij}$ represent the set of sentinel rules for transitioning from Major Context *i* to Major Context *j.* The set of sentinel rules for any given Context C_i is $\mathbf{S}_i$ which is the combination of all $\mathbf{S}_{\mathbf{ij}}$ where <*i, j*> is a valid transition within mission *M*.

$$S_i = \bigcup_{j:<i,j>\in M} S_{ij} \qquad \text{(ix)}$$

The actions of an agent when operating under the control of a Major Context $C_{i,}$ are determined by the Major Context C_i

$$C_i = <\mathbf{S}_\mathbf{i}, \mathbf{FB}_\mathbf{i}, \textit{Context-logic}_i> \qquad \text{(x)}$$

Where $\mathbf{FB}_\mathbf{i}$ is the local fact base.

2.2.4 Competing Context Paradigm

In some complex tactical situations, a CxBR agent might be faced with more that one choice of contexts to transition to. The current CxBR architecture does not explicitly address this issue. The competing context approach was conceived by Saeki & Gonzalez [28, 42, 43] to address such situations. An example given by Saeki & Gonzalez [43] is an agent that seeks to reach a meeting at an appointed time. If this agent encounters a tire blowout while en route, it is faced with making a decision on whether to fix the tire and continue with the car, or abandon the car and walk to the meeting. The decision on what to do is based on the agent’s most important goal as well as its current location relative to the meeting. If the agent is close enough to the meeting location, and its most important goal is to get to the meeting on time, the agents’ best decision would be to abandon the car because it would take longer to fix the tire. However, if its most important goal is to get to the meeting with the car, the agent must fix the tire. Saeki & Gonzalez [43] note that in such cases, “it is beneficial to define the current situation as a set of needs to be addressed by the agent in order to accomplish its mission.” The eligible contexts to which the agent can potentially transition should meet some or all of these needs. The contexts then ‘compete’ with each other for control of the agent.

To accomplish context competition, Saeki & Gonzalez [43] implemented a constraint-based system that integrates the ability of an opportunistic agent with the matching of these constraints. There are four processes to their approach. 1) The generation of situation interpretation metrics (SIMs); 2) The selection of the contexts that satisfy the generated metrics; 3) The matching of the attributes of the various contexts in the selected context group; and 4) The time-warp simulation. The last is optional, based on whether the outcome of the third step is ambiguous.

- The situation interpretation metrics (SIM) is the relevant information about the current situation as it relates to the most important goal. An example of the SIM for the example presented earlier is the distance between the meeting place and where the flat tire occurs is generated. The time it would take to fix the tire is computed, as is, the time it would take to walk to the meeting based on the current location.

- The relevant context group selection is the process where a set of potential contexts are selected based on the currently active context and the most important goal. As soon as a decision is made on the most important goal, some of the attributes on all contexts are compared against this goal. The contexts with those attributes matching the goal are selected as *candidate contexts*, while the others will no longer be considered. Saeki & Gonzalez [43] note that “this has an effect of reducing the search space of potential best contexts”.
- The context attributes matching “is the process where contexts match all their attributes to SIMs.” At this stage, the contexts that have less attributes matched to the SIMs are eliminated. If more that one context is picked at this stage, the process moves on to the time-warp simulation.
- The time-warp simulation is the “process that executes a super real-time simulation until the ‘best’ context is identified while the current simulation time is stopped” [43]. “This process starts with the current context and current SIM and alternately simulates the transition to each candidate context”[43]. Temporal SIMs are generated for each context transitioned to in order to ascertain whether the current goal is achieved by the transition. Saeki & Gonzalez [43] note that the transition that “best projects the satisfaction of the current immediate goal is selected as the winner” and this context is then activated in the ‘real’ simulation. If a context competition is required within the time-warp simulation, Saeki & Gonzalez [43] calls this *nesting*. They note that the active context would now be randomly selected.

The authors [28] showed an agent utilizing this approach to act appropriately in a simple driving scenario based on its most important goal and the attributes of the various contexts. They furthermore suggest ways to improve the competing context approach. Among the ways suggested are: making the selection of candidate contexts a dynamic and continuous process by anticipating future events, making the context matching dynamic and continuous and the elimination of the time-warp simulation in some cases in favor of a thorough evaluation of the situation.

In either approach, the agent lacks the capability to learn. The experiments conducted by Saeki & Gonzalez [28] in modeling the agent driver do not take into consideration the effect of stress, emotions, and other factors that affect human behavior. Based on their [28] experiments, when the agent is faced with the options of walking to its destination versus fixing the car tire, the effects of fatigue and weather should be considered. It would be unexpected for some humans / agent to walk beyond a certain limit, say - 2 miles to a meeting under very high temperatures just because walking is calculated as being the best option. Whereas, the agent could as well fix the flat tire and increase its speed to meet it's most pressing need of getting to the meeting on-time and reducing the effect of fatigue. In essence, the agent should be intelligent enough to know how to adjust its speed to catch up for lost time spent in fixing the tire.

2.2.5 The DUAL Architecture

DUAL is a context-sensitive cognitive architecture conceived by Kokinov [90]. It is an implementation of Kokinov's dynamic theory of context [34]. It is made up of "a unified description of mental representation, memory structures, and processing mechanisms."

One of the major principles of DUAL is that the interaction of smaller structures form a larger one. DUAL-based models can be analyzed at three different levels of granularity:

- Microlevel: this is the smallest granule of an agent. The internal structure of the agent, the information processing capabilities of the agent and differences amongst agent types are analyzed at this level.
- Mesolevel: this is a coalition of DUAL agents. Kokinov [90] defines a coalition as "a set of agents and a pattern of interactions among them". There are two distinct properties of a coalition; emergent and dynamic. At this level, the interactions between agents, the "emergence of non-local phenomena out of local activities" and the dynamics behind the organizational structures of the DUAL agents in a coalition, are considered.
- Macrolevel: this level deals with the formations created by DUAL agents and the models. Kokinov [90] describe formations as a big population of agents. "At this level, concepts like working memory, mapping and analogy are taken into effect during analysis"[90].

Kokinov [90] notes that these three levels are interdependent and that it is difficult to distinguish one from the other because an analysis of a coalition would depend on the individual properties of the members of the coalition. If a change is made at a level, it affects the other levels.

A cognitive system developed with the DUAL architecture is usually made up of multiple simple agents that are highly interconnected with each other. Each of these agents contains a specific knowledge for the performance of a specific task. The interconnections between agents could be permanent links, or created dynamically during the course of achieving a goal. Exchange of information is only possible between agents that are close and have direct links with each other. The behavior produced for any given situation is a combination of the parallel actions of all active DUAL agents in the system. The action of individual agents is dependent on their activation level. At different occasions, the activation level of the agents in different groups would dictate the computation necessary for that situation, and thereby produce a certain behavior. In the DUAL architecture, there is no distinction between external and internal context. The activation level amongst the agents, help in explaining the various contexts and priming effects[14].

The architecture of DUAL agents is a hybrid one, meaning there are two parts to an agent. Kokinov [90] calls these parts the “L-Brain and R-Brain” with no relationships to the human brain structures. The L-Brain is designed according to the symbolic paradigm while the R-Brain is designed according to the connectionist paradigm. In any given context, the L-Brain represents a piece of knowledge while the R-Brain is the relevance of this knowledge to the context. R-Brains operate in a parallel manner.

Kokinov [90] utilizes a frame-like representation scheme for representing agents from the symbolic point of view and the connectionist perspective is used in the representation of contexts. The relevance of each agent in any given situation helps in the representation of contexts in a distributed way. The measure of relevance is the degree of connectivity that exists between an agent and other agents.

Kokinov notes that the “R-Brains are processors that calculate the activation values and outputs of the nodes on the basis of their input values and current activity.” He

[14] Priming effect according to Kokinov [90] “is the change in human response to a target task caused by changes in the subject’s preliminary setting” while Context effect “is the change in human response caused by changes in the environment of the target stimulus.”

states that "it is important that the activation of a node is a function of both the environment and the currently received activation from the net, and the previous activation level of the node."

A few successful applications and models have been developed on the DUAL architecture, e.g. AMBR [91, 94] (Associative Memory-Based Reasoning) [15]

2.3 Contextual Learning

Most studies on context claim to have some form of learning capabilities. Turner [36] states that the agent used to control the autonomous underwater vehicle (AUV) learns by merging c-schema objects to form new c-schema's that define the current situation. The problem with this is that the agent would always behave the same way under the same conditions even if its behaviour were bad. This doesn't result in true learning.

Bonzon [129] developed a contextual learning model that stores "sequences of inference steps that lead to discovery of object-level concepts to be used later"[129]. He tries to achieve generality in learning with this approach.

Kokinov [34] dynamic theory of context in which context is defined as a dynamic state of the human mind has potentials of incorporating learning. He attempts to show the difference between AI approach to contexts and psychological approach. He states that AI's approach to contexts "may be characterized as navigating between and within the context boxes" [34]. Kokinov acknowledged the lack of learning in his DUAL cognitive architecture [90].

[15] AMBR adopts "an interactionist approach that identifies analog access, mapping, transfer, etc as parallel subprocesses rather than the conventional serial stages" in analogy-making [91]. More information on AMBR can be obtained from the works of Kokinov [91, 94]

Balkenius & Moren [119] notes that “context learning is an entirely passive process” that doesn’t depend on actions of an agent. They showed how stable context representations are learnt from “a dynamic sequence of attentional shifts between various environmental stimuli”.

In this dissertation, it is asserted that:

> *True learning of contexts exists when a learning agent is able to adjust and modify its beliefs on its actions in that context. In other words, true learning is said to occur when a learning agent understands the attributes of the current situation (context) and thereby modifies its actions when in that context as a result of its interaction with its environment.*

A method that implements a learning agent modifying its actions and understanding of any situation by interacting with its environment is presented in this dissertation.

2.4 Others

There are many other modeling techniques that represent human behavior. The Contextual Control Model (CoCoM) of Hollnagel [122, 123] attempts to take the contextual effect of the environment on the performance of the operator. It is based on three concepts: competence, control and constructs. Competence is the set of actions that are possible in any given situation based on the needs of that situation as recognized by an operator; Control defines the way competence is applied. There are four modes of control: scrambled, opportunistic, tactical and strategic. These modes range from no control at all (scrambled) to a completely deterministic control policy (strategic). Finally, construct is the known information about the current situation. It is the basis on which actions are selected by the system.

EPIC (Executive-Process/Interactive Control) has a goal of accurately accounting for the timing of human perceptual, cognitive and motor activities. It provides a framework that allows for the easy construction of human-system interaction models that are accurate with detailed information processing units for practical problems.

Rational Behavior Model (RBM) developed by Byrnes [135] is a three-level intelligent control architecture for autonomous agents. The three levels are: strategic, tactical and execution. At the execution level, the emphasis is on the control of the hardware, at the tactical level, the emphasis is on the selection of the appropriate sequence of behavior for the agent and the strategic level deals with the plan and mission logic. Some successful applications of RBM include the control of autonomous underwater vehicles (AUV) Holden [136].

The modeling paradigms described in this dissertation are the important models relevant to this research. There are other modeling paradigms not described in this dissertation because they are not relevant to this research. The described modeling paradigms show that many techniques exist for modeling human behavior and it is left for the modeler to decide what modeling technique best fits the aspect of human behavior (s)he intends to model. By and large all the modeling paradigms mentioned in this research all suffer from the same problem, i.e. the lack of a robust, self-enhancing, learning mechanism for the acquisition of knowledge used in modeling. The models built with these techniques are overly-dependent on expert knowledge for their successful implementation and functionality. Thus models built for tactical situations might fail to achieve their mission objectives. This research produces a method that seeks to eliminate these issues.

2.5 Comparison of HBR Models

With so many modelling techniques available for representing human behaviour, the question of which technique is best unavoidably arises. With this in mind, some have attempted to compare some modelling paradigms against some benchmarks. Bolton et al. [20] compares three HBR modelling techniques as to how they generate instructional materials for Navy training. The results show that the modelling techniques compared led the participants of the training exercises to performance improvements that were equivalent from a statistical perspective.

The US Air Force Research Laboratory has recognized that although there has been progress in the HBR research arena, the academic and commercial sectors aren't producing human behavioural representation methodologies / technologies that sufficiently meet all the requirements of the Air Force's modelling and simulation needs. Investments in this area were undertaken by the Air Force through a program named Agent-based Modelling and Behaviour Representation (AMBR) [11, 51]. A primary objective of the AMBR project was to improve the developments made in the cognitive and behavioural modelling of military applications. The AMBR project compared various HBR modelling approaches. One of the modelling goals was multi-tasking, done in a an enroute air traffic control domain. The HBR modelling paradigms compared where the ACT-R, D-COG, EPIC-Soar, and iGen[16]. It was noticed that all models built by the various techniques successfully approximated trends and central tendencies of the data

[16] iGen is based on the COGNET model

used by behaving similarly, but the way the various models implemented the multi-tasking capability of human behaviour differed across the four models.

Kokinov [90] compares his DUAL architecture to that of Anderson's ACT-R [101]; he notes the similarities between both approaches but asserts that the declarative knowledge is separated from the procedural knowledge and different mechanisms control each one in ACT-R and as such, the priming effects cannot be explained. His DUAL architecture has this ability. Furthermore, ACT-R considers only static environments, whereas his DUAL takes note of the dynamic nature of the environment. Kokinov [90] concedes that ACT-R is superior to DUAL because of its learning capabilities.

Nason [118] notes some differences between the implementation of their Soar-RL architecture and ACT-R. These include: 1) soar can allow the encoding of information for the preference of a particular rule over another whereas ACT-R can't; 2) in soar, an operator can have many rules that depend on different goals, whereas in ACT-R the mechanisms relate to only a single goal.

A combination of different modelling techniques have been developed over the years to account for the shortfalls of the individual techniques, for example the integration of the Soar modelling technique with Reinforcement learning [118], the integration of EPIC modelling technique to Soar [117].

Brown [137] compared CxBR with other traditional rule-based reasoning systems and found that CxBR performed better and was more concise in the representation of knowledge. Gonzalez et al. [4] compared CxBR in the control of an autonomous vehicle with other traffic generating methods with emphasis on car-following algorithms and found that expanding a system developed from CxBR is considerably easier. They further noted that in comparing the size of the program based on number of lines of code, and in terms of the designed behavior, CxBR had slightly more lines of code with more behaviors compared to the car-following algorithm which had fewer behaviors. This represents the concise nature of representing knowledge in CxBR architecture. The execution rate for the CxBR model was the same. Lorins et al. [92] also compared the CxBR modeling technique to that of the CxG modeling technique in decision making and concluded that more exploration on the advancements of both techniques and their implementation is needed to have a complete comparison based on the way context is represented, the transitioning between contexts and the way knowledge is acquired.

Although there are no known direct comparisons on the CxBR technique to ACT-R, Soar or DUAL, we can intuitively determine what the outcome of one would be, based on the underlying principles for the creation of models using each technique. DUAL and ACT-R are based on the generation of a particular "behavior" from the formation of a smaller "behaviors". The question that isn't answered is what level of granularity would determine a small "chunk" of knowledge? These architectures are also designed for general problem solving and as such are structured around that. ACT-R is designed as being cognitively correct. That is, it represents the human cognitive process. On the other hand, CxBR was designed specifically for its efficiency, effectiveness and ease of modeling human behavior in tactical situations. The ease at which an agent being controlled with CxBR identifies and transitions between contexts is inherent in its design. CMB closely resembles CxBR, but its concepts are yet to be fully implemented and tested as has CxBR. COGNET does not take the context of the situation into its model as intuitively as CxBR. It also lacks an inherent learning mechanism. CxBR also lacks a learning mechanism. However, it has been shown that it facilitates learning and some techniques have been developed that take advantage of its contextual decomposition. Its hierarchical structure allows a learning problem to be broken down into smaller problems. Learning these smaller problems could be facilitated. There are many other advantages CxBR has over the other behavioral modeling techniques. This is why this research was carried out using the CxBR model.

An addition of a learning mechanism to the CxBR architecture would be implemented using a reinforcement learning algorithm to allow for the effortless augmentation of the SME's knowledge based on the goals. More about this later.

2.6 Summary

An introduction to context as it relates to modeling human behavior was presented. Some modeling techniques that utilize contexts in modeling human behavior were discussed. Learning in contexts was summarized and a comparison between some human behavior modeling techniques was carried out, during the comparison and analysis, CxBR was determined to be the technique of choice for this research because of its intuitive nature in the modeling of tactical behavior

CHAPTER 3: PROBLEM DEFINITION

3.1 Problem Statement

The first step in the resolution of any problem is to identify the essence of the problem. In this research the problem being addressed is a subset of the overall problems being studied in the Human Behavioral Representation (HBR) area. That is, to model an agent that would act the way a human acts when faced with different scenarios. In general, for an agent to behave like a human in tactical decision-making, at least four problems have to be overcome, these are:

- To efficiently represent the behavior of the human
- To effectively represent the behavior of the human
- To acquire these behavior in an efficient and effective way
- To validate the acquired behavior

This dissertation addresses the last two problems. As has been described in Chapter 1, the major modeling techniques do not address all four problems simultaneously. Most investigations have focused on how to efficiently represent the behavior of humans. However, the acquisition of these behaviors has been mostly achieved through question and answer sessions with subject matter experts or through observing the performance of a subject matter expert. This limits the process to incorporating what a subject matter expert knows and is able to articulate or demonstrate. This knowledge is often incomplete and/or flawed.

A combination of methods is usually needed to tackle all four problems listed above. Context-Based reasoning (CxBR), because of its modular design and ability to prune the search space, has the capabilities of encompassing solutions to all four problems. Therefore, this research was based upon CxBR as the underlying modeling paradigm. This research also improves the CxBR technique by using RL.

Traditionally, to limit the ambiguities and errors introduced during knowledge acquisition (KA), multiple domain experts are needed to provide different and complimentary viewpoints on a simple problem. This was done either by question and answer sessions or observing the expert performing the task. The need for domain experts could become heavily dependent on each other, because when one expert reaches his/her knowledge threshold, he/she calls upon another domain expert, and the team of experts continues to grow until the minutest ambiguity is resolved. The question of when to stop bringing specific subject matter experts could arise because the cycle of an expert having only specific knowledge of certain aspects of his or her domain would always exist. Therefore, several subject matter experts would be needed to totally cover a domain. This could lead to the developed system being inefficient in carrying out the tasks assigned to it because of the retrieval of information needed to resolve the simple problem.

Some of the issues highlighted above are partially resolved by Context-Based Reasoning (CxBR). CxBR, as described in the previous chapter conforms to a *Markov process*, “in which the next state of a system is determined by the current state of that system and not by the previous states” [8]. The dictionary [1] defines a markov process as “a simple stochastic process in which the distribution of future states depends only on the present state and not on how it arrived in the present state”. For example, as stated by Gonzalez et al [9], it is highly unlikely for a tire blowout event to occur while a car is waiting at a traffic light. This is because the current event is the car waiting at a traffic light and all events that occur next would be based on this fact.

Although there are many successful applications of various HBR techniques, as noted in Chapter 1, they all suffer from the same fate; i.e. they suffer from the limitations inherent in the way knowledge is acquired - the total dependence of knowledge acquisition and representation on subject matter experts. Usually, the experts determine what actions to perform in a given situation (context) and how the agent should behave in all situations as perceived by the expert. Ranges of valid values are provided by the experts for a given context and as such, all actions by the agent are predefined based on these ranges. The question of what happens when the SME lacks knowledge for a specific context or provides the wrong knowledge for that context arises. Also, how do you reconcile differences in expert opinion for the same context? Usually, the knowledge engineers start as novices in the domain being modelled. Would the KE rate one expert highly over another without any basis? The fact that the SME provide the knowledge and thus determine the behaviour of an agent isn't wrong. What is wrong is the inability for these models to be improved beyond the SME's level of competence. The enhancement of the agent model should be geared towards achieving the overall mission goal. A system where the acquired knowledge - the actions and thus the behaviour of the agent can be enhanced based on the mission goal, irrespective of the SME's imparted knowledge is most desirable. Conceptually, this can be achieved by placing the model developed from SME's knowledge in a simulator and exposing the model to situations not imagined by the SME. The model is run multiple times until the knowledge acquired from the SME is modified to address these new situations. The model is thus enhanced to perform better, based on the mission goal.

Enhancing the behavior of HBR models is the goal of this research. This is achieved by breaking the barrier of dependence on SME for the knowledge. This dissertation investigates the feasibility of using experiential knowledge from an agents experience in a simulator to enhance the agents' behavior.

A few attempts have been made towards filling the voids left by incomplete knowledge in a HBR model; In particular, knowledge acquisition methods that involve observing expert actions are an attempt to eliminate some problems introduced during SME introspective sessions. One of these is the lack of explanation of implicit actions performed by the expert. Fernlund [10, 138] developed the GenCL model that captures expert knowledge through observation. This captured knowledge is then transformed into contexts that control an agent that behaves like the expert. Fernlund's [10, 138] method eliminates some of the errors that exist during the acquisition of the knowledge and representation; it also eliminates occasional ambiguity caused by the experts language and the difficulty in explaining implicit knowledge. Although the goal of Fernlund's research [138] was to automatically create agents by observing expert actions, it would be interesting to make these agents perform better by learning through their experiences in a simulator. In an example in his work [138], an agent that was created by observing a 'reckless' driver who ran through a red light, behaved exactly like the driver by running through the red light in a simulator. Although Fernlund achieved his goal, the method presented in this research goes a step further and refines (enhances) the agents behaviour by making the agent learn that running through a red light represents a failure. The agent learns this through the experience it gained in the simulator. In essence, it would be advantageous for an agent to learn good from bad, relative to a goal even after observing an expert execute a mission.

With the existing limitations of knowledge acquisition techniques for human behaviour representation, this research produces a method that enhances a model by subjecting a learning agent to situations not experienced by the SME, yet realistic in the execution of a mission. Furthermore, the new methodology also addresses the lack of expert explanation of implicit knowledge which could be a cause of incomplete knowledge in the model.

3.2 Hypothesis

This research proposes the following hypothesis:

> *Reinforcement learning can be used to automatically and efficiently enhance a tactical agent's behaviour from the experience gained by the interaction of the agent with its environment. Additionally, based on the mission goals, these agents will perform better than the agents developed from knowledge acquired from experts.*

The learning process of the agent is based on the predefined knowledge acquired from the SME. The model built upon this knowledge is then refined (enhanced) in a simulation based on the experience gained by the agent during its interaction with the environment. Learning by the agent is non-monotonic, in that it can retract previously learnt actions during its interaction with the environment when a new action is found to contradict an existing one.

3.3 Contributions

The contributions of this research are:

- Providing a technique that breaks through the barrier of SME knowledge limitations.
- Show that a human behaviour model, in particular a CxBR model, can be automatically enhanced from the experience gained by the agent during its interaction with the environment without human intervention.
- Providing an algorithm for the model enhancement process that can be applied in other domains.
- The modified contexts would provide a more robust knowledge base that include behaviours missed by the SME.
- Provide an analytical basis for knowing a fully enhanced model
- Provide a prototypical framework for the enhancement of models
- A by-product of the experiential learning technique is the provision of an unbiased validation method for human behavioural representation systems.

The following are advantages of this research:

- The synergistic combination of CxBR and Reinforcement Learning would be the foundation for achieving the automatic enhancement and creation of the human behaviour models.
- The synergistic combination of CxBR and RL can aid in the acquisition of the behaviour of experts via simulated agents in an efficient and effective way for

them to be correctly represented in a CxBR system thus eliminating the errors that exist when this process is done manually.

- The elimination of the common criticism against artificial intelligence researchers creating contexts as a movement between fixed boxes, by providing a mechanism that automatically creates a context (behaviour) on the fly.
- Providing a HBR methodology where the knowledge acquisition methods are goal oriented and thus an agent would identify and fix any lapses in the acquired knowledge based on the predefined goal.

CHAPTER 4: RELEVANT MACHINE LEARNING TECHNIQUES

We propose to break the SME knowledge barrier by applying machine learning techniques that enable experiential learning. In this chapter, some machine learning strategies are described and a strategy that best achieves the goal of the research is selected. This chapter is divided as follows: section 4.1 provides an introduction to machine learning; section 4.2 describes supervised learning and includes some examples; section 4.3 describes unsupervised learning while section 4.4 describes reinforcement learning. Section 4.5 compares these machine learning groups and makes the case for the learning strategy of choice for this research.

4.1 Introduction

The oxford dictionary defines learning as “Behavioural modification especially through experience or conditioning” [1]. Mitchell [53] states that “learning is improving with experience at some task”. Dietterich [54] states that “machine learning is the study of methods for programming computers to learn”. Dietterich goes on to argue that although it is relatively easy to develop applications that can be applied to solving a wide variety of tasks, there are generally some tasks for which it is difficult or impossible to do this. He groups such tasks into four categories:

- Problems were no human experts exist.
- Problems were human experts exist but cannot explain their expertise because of the implicit nature of what they do.

- Problems where the underlying parameters / attributes change rapidly, e.g. the stock market.
- Problems that are user specific in a large domain, e.g. a mail filtering system for an organization, each user would have different criteria for filtering junk mails; "self customizing programs" Mitchell [53].

Dietterich [54] believes that machine learning addresses most of the same issues that statisticians, data miners and psychologist address, but the major difference lies on the emphasis placed on the issues. While statisticians want to know and understand how the data has been generated, data miners look for patterns in these sets of data. Psychologists on the other hand try to understand why different people exhibit various learning behaviours. Machine learning, on the other hand, is concerned mostly about the accuracy and effectiveness of the resulting computer system.

Dietterich [54] states that a learning task can be classified along many dimensions, but believes that an important dimension in which all learning tasks should be classified is the distinction between empirical and analytical learning. Dietterich defines empirical learning as one that relies on some external experience whereas analytical learning requires no external inputs.

Dietterich [56] draws the relationship between learning and reasoning. He attempts to show the ways in which machine learning research has either incorporated reasoning or left out reasoning.

Machine learning could be divided into three types; Supervised Learning, Unsupervised Learning and Reinforcement Learning. Supervised learning is when a "teacher" is present, i.e. the agent learns from training samples available to it.

Unsupervised learning is when there is no teacher; there are no training examples available to it. Reinforcement learning is when there is no teacher, there are no training examples; the agent is rewarded either positively or negatively for being in certain states. Over time the agent identifies what states are best to be in and what states to avoid. Reinforcement learning agents can be said to start with no training examples and build approximate training examples from their environment. [17]There is some controversy as to whether RL can be classified as a supervised learning technique. Some researchers believe it falls under supervised learning while some others believe it falls under unsupervised learning. Yet another group of researchers believe RL is the third group of machine learning strategies/techniques and falls under its own group. There are major differences between supervised learning techniques and RL. These include the absence of a teacher vis-a-vis explicit training samples [176]. In RL, the rewards received do not tell if an action is good or bad - it is left for the agent to make that judgment based on the results of its decisions / actions. There are also major differences between RL and unsupervised learning. For example, most unsupervised learning techniques are based upon classifying the training samples based on “some distance” or “closeness” to a particular property. Bartow and Dietterich [176] state that a supervised learning problem can be converted to a RL problem, with the resulting problem becoming more difficult. A RL problem however, cannot be converted to a supervised learning problem. With these differences between RL and other learning techniques, we assert that RL be classified as being in its own group.

[17] The different types of machine learning are described in detail in later sections

4.2 Supervised Learning

Supervised learning is when a teacher is present during training. It is akin to a student going to school everyday and being taught by a teacher. Christodoulou et al. [57] state that an essential ingredient of this type of learning is the presence of an external teacher. Usually the learner is given training sets that contain input-output pairs. For each input shown to the learning agent, there is a corresponding output assigned to it. Nilsson [174] states that finding a hypothesis that closely agrees with the mapping of a function to the training samples is an objective of this type of learning. Dietterich [54] notes that a key challenge for this type of learning is generalization. After a few training input–output samples are presented to the agent, the learning agent is expected to learn some function that correctly identifies or predicts what the output of a new input set would be. Usually, the input–output pairs used during training are thought to be independent of each other for proper training to occur. An example of a supervised learning implementation is a simple feed-forward Artificial Neural Network. An example of a supervised learning concept is observational learning.

4.2.1 Observational Learning

Observational learning is also known as learning by doing nothing. It can be considered a supervised learning technique. Bandura, as narrated by [60], has demonstrated that the "application of consequence" is not necessary for learning to take place. He also suggests that learning can occur through the process of observing another person's activity. Bandura, as narrated by [60], states a four-step pattern for learning.

- Attention – this is the step in which the individual notices something in the environment
- Retention – this is the step were the individual remembers what was noticed
- Reproduction – this is the step where the individual produces a copy of an action that was noticed
- Motivation – a consequence is delivered by the environment that changes the probability of the behavior being carried out again.

Another definition of observational learning is that an observer's behavior changes after viewing the behavior of a model [59]. Consequences can affect an observer's behavior; these consequences could be some form of reinforcement in the case of positive consequence and some form of punishment in the case of negative consequence. The guiding principles behind observational learning are as follows [177]:

a. The observer will 'imitate' the behavior of the human that it finds attractive or desirable. An example of this is when a child imitates the behavior of a cartoon character he admires, say 'Spiderman'. This child most likely would attempt some of the actions he observes Spiderman perform, he may or may not be successful in attempting these actions but he does attempt them.

b. The observer reacts to the way his 'idol' is treated and mimics the idol's behavior. When there is a reward given to his idol for behaving a certain way, the observer will attempt to copy the behavior but if there is a punishment, the observer would avoid trying out that behavior. An example is a person learning how to drive by observing his trainers behavior, if the trainer goes above the speed limit and gets a speeding ticket from the police, the observer will not attempt to go above the speeding limit, because he knows he could get punished for that.

c. There is a distinction between acquiring a behavior and performing a behavior. By observing an idol, an observer can acquire the behavior without attempting to perform them. But the observer can attempt to perform the acquired behavior at a later time when the need arises. An example would be an automobile owner watching a repairman change his tire. The owner of the car (observer) has acquired the skills of changing a car tire but didn't necessarily perform the acquired knowledge because the need didn't arise, however if he gets a flat tire some days later, he could apply the acquired knowledge and change the tires without the need for the auto repairman.

One of the most popular methods of knowledge acquisition is by observing expert actions. This is so because it has an added advantage of acquiring implicit knowledge of the expert. A disadvantage though, is that if the expert performs badly, so will the model built from the acquired knowledge. Furthermore, the built model is limited to what the expert knows.

4.2.2 Artificial Neural Networks

A neural network as defined by Christodoulou & Georgiopoulos [57] is a "network of many simple processors (units, nodes and neurons) each of which has a small amount of local memory". These processors are interconnected and they carry data between processors. Gurney [61] states that a "Neural Network is an interconnected assembly of simple processing elements, units or nodes, whose functionality is loosely based on the animal neuron. The processing ability of the network is stored in the inter-unit connection strengths, or weights, obtained by a process of adaptation to, or learning from, a set of training patterns". Artificial neural networks are computational models that can learn to generalize data [62]. Figure 4.1 shows a neuron.

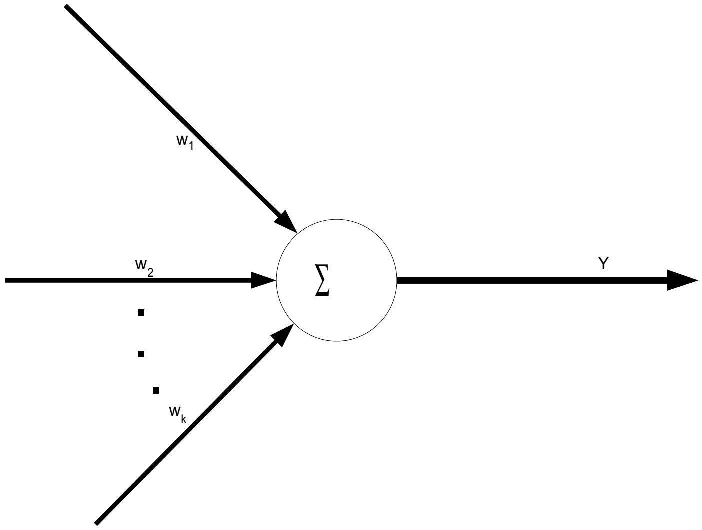


Figure 4.1 A Neuron

A simple feed forward neural network works as follows: training data is provided consisting of several input sets and each with a corresponding output. These training data are presented to the system and the weights of the system are adjusted appropriately to reflect the training. Figure 4.2 shows a neural network with multiple layers. After training the network, a sample data is passed through the system to predict what category a new dataset belongs (in classification problems).

Artificial neural networks are of different types. Some are classified as supervised while others are said to be unsupervised learners.[18]

[18] The discussion about artificial neural networks classified as supervised learning is the feed forward networks and the back propagation algorithm. Discussions about the self-organizing map which is a non-supervised learning network is discussed under unsupervised learning.

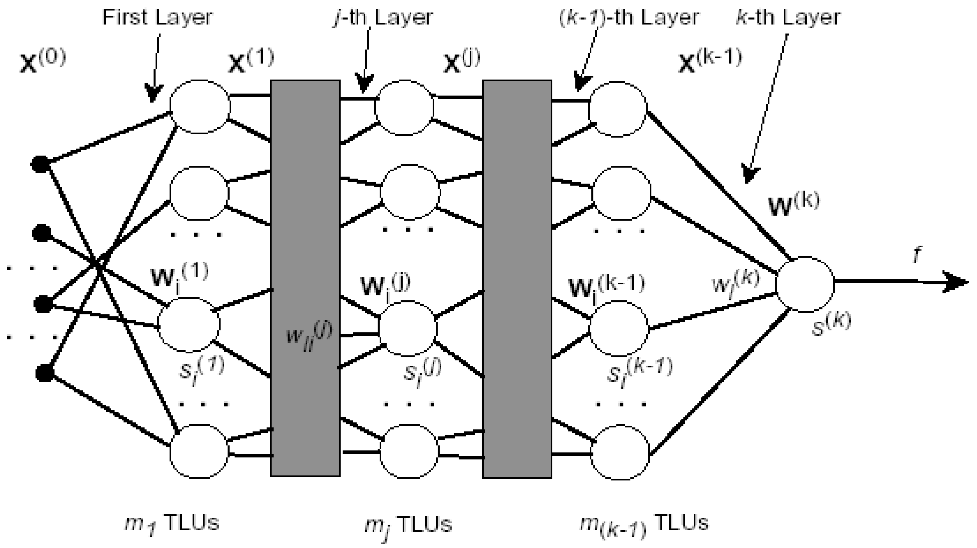


Figure 4.2 *k*-layer Network (Reproduced from Nilsson [174] without permission)

4.3 Unsupervised Learning

This is a scenario where the learning agent is not exposed to an external teacher or critic. The training inputs do not have a corresponding output. The outputs of the system are unknown. Because of the absence of an external teacher, Christodoulou et al. [57] suggests that a provision be made to identify the quality of the representation that the learning agent is required to learn. The parameters of the agent are then optimized with respect to this measure. Dayan [181] defines unsupervised learning as techniques that "study how systems can learn to represent particular input patterns in a way that reflects the statistical structure of the overall collection of input patterns". The input sets are analyzed for similarities between any features. "Procedures that attempt to find natural partitions of patterns within sets are utilized" [174]. After the training is over, the inputs introduced to the agent are grouped based on the similarity measure defined on the learning agent. Unsupervised learning techniques are usually used in classification and data compression problems amongst others.

4.3.1 Self-Organizing Networks

This is a group of neurons where the weights are adjusted to match the input vectors in a given training set. Competitive learning, which is a learning methodology that divides a set of input data into clusters that represent the input data, is used as the learning mechanism in these types of networks.

When input data is presented to the network, only one output (known as the *winner*) is selected. Euclidean distance[19] is utilized in the selection of the winner.

A self-organizing network has two stages of operation, the first stage is the training of the network, in this phase, the network organizes itself by the use of the competitive learning process. The second phase is the mapping phase where a new input is passed through the network for classification or categorization. This new input is given a location on the network, and the winning neuron whose weight is closest to the input data determines how the new data is classified.

4.4 Reinforcement Learning

Imagine you are sailing in the ocean and suddenly there is a great storm. In this storm all your crew members suddenly disappear, making you the only survivor. Your

[19] The Euclidean distance measure utilized in the selection of the winner is as follows:
$k\!: \; \| w_k - x \| \leq \| w_o - x \| \; \forall \, o$

navigational system was destroyed in the storm, but you are lucky to have a large food supply. You also have fuel and your boat engine is in excellent working condition. After the storm, you find yourself in an unfamiliar place. This is the first time you are experiencing this situation and you have to get back home (to shore). How do you achieve this goal? The only way to achieve this goal is to try to sail randomly towards shore; you utilize the actions available to you which are moving in any direction with the hope that you are moving in the right direction towards shore. When you successfully get to shore, if rescuers want to go back to where the storm occurred to attempt a rescue mission, all you need to know and tell them are the successive directions / steps you took to get you to your present location.

This same problem can be extended by noting that you have a limited supply of food and drinks to survive on and as such have to achieve the goal of getting to shore under a timed constraint. This in a nutshell is reinforcement learning, where you have a goal and attempt to achieve the goal through interacting with the environment and making decisions based on the experiences gained from the rewards and punishments during these interactions. Another example of reinforcement learning is deciding on the best route to a place you go to frequently, for example, your office. You might be faced with many routes from your house to your office. In the early stages of trying out these routes, you will have no opinion on the fastest route between your home and office. Eventually after you have attempted each of the routes multiple times, your opinion on the various routes would have been made in terms of shortest time, shortest distance, fastest routes, and best time of the day to take a route, and much more.

4.4.1 Overview

Since the advent of modern computers, it has always been the wishes of man to have his computer learn like he does, by learning through mistakes it makes. Barto [162] states that the term reinforcement emanates from experimental psychology and animal studies. It refers to the strengthening or the weakening of the probability of the response as a result of the occurrence of an event [162]. Barto [162] emphasizes this point by noting that in its simplest form, reinforcement learning utilizes a commonsense approach to events, in that, if an action produces good results or responses, that action is reinforced. Generally, in humans, the consequences of our actions always influence our behavior or the behavior of others [163]. These consequences are based on three principles [163] which are; "Consequences that give rewards increase a behavior, those that give punishments decrease a behavior and those that produce neither rewards nor punishments tend to extinguish a behavior." An example to illustrate these principles is a child that touches a hot stove. The consequence of the child's action is pain and this punishes the child, thus the child would avoid touching a hot stove in the future. If this same child performs a task for which she is rewarded with candy (say, cleaning her room) she would opt to clean her room regularly.

Gosavi [78] calls RL "an offshoot of dynamic programming; a way of doing dynamic programming within a simulator". A feature of Reinforcement learning is that it is primarily learning from experience i.e. learning from one's mistakes. It can be argued that RL also encompasses learning from the mistakes of others, which is a key feature of observational learning. As noted by Barto, the modern interest and development of reinforcement learning is driven by the need for "autonomous agents that can operate under uncertainty in complex dynamic environments and also the need for finding approximate solutions to large scale dynamic decision making problems” [162].

RL is well suited for control and optimization problems in Markov Decision Problems (MDP) [63, 162, 178, 180]. Recent investigations have indicated successes in using RL techniques for *Semi-Markov Decision Processes (SMDPs)* and *Partially Observable Markov Decision Processes (POMDPs)* [169, 180, 182]. POMDPs are situations where an agent cannot see the world outside its immediate environment. POMDPs are usually modeled by defining a mapping function between the hidden states and what it observes. Finding an appropriate mapping between what it observes and the actions that produces it, serves as the goal of a POMDP agent.

In general, the RL problem can be shown to involve the following steps:

- The execution of an action by the agent on the environment during the agents' interaction with it from a set of possible actions $a_t \in A(s_t)$. This leads the agent to a new state.
- The receipt of a reward r_{t+1} from the environment when in the new state s_{t+1}

A note should be made that the reward or punishment received and the new state are dependent upon the action executed by the agent as well as the state in which the action was executed. Figure 4.3 shows a typical Reinforcement Learning agents' interaction cycle with its environment, while Figure 4.4 shows the reinforcement learning architecture.

In figure 4.3, it can be seen that as the agent performs an action on the environment, the state of the environment changes, this in turn triggers a reward or punishment to the agent and the agent is then in a new state.

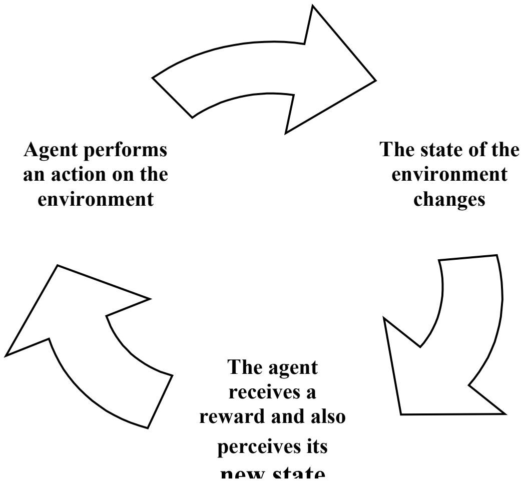

Agent performs an action on the environment
The state of the environment changes
The agent receives a reward and also perceives its new state

Figure 4.3 A Typical Agent Interaction Cycle

In figure 4.4, the agent performs an action ai, the state of the environment changes from s_i to s_{i+1}, the agent then receives a reward or punishment from the environment.

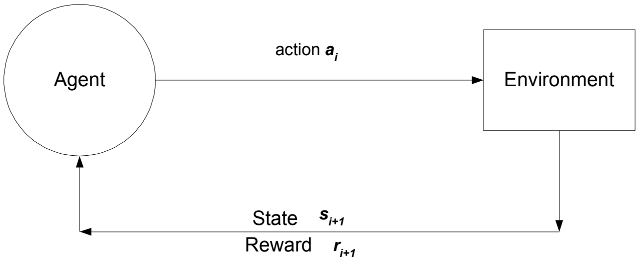


Figure 4.4 Reinforcement Learning Architecture

Barto [162] observes that a typical RL problem includes uncertainty because the agents' behavior and its environment are subject to some form of randomness. In some cases, an approximate model representing these uncertainties may be available for use in resolving the RL problem. An important part of the RL problem is the reward input. This is the aspect of the RL problem that indirectly controls[20] the behavior of the agent. The reward received by the agent in a given state after many trials, points it in the right direction. According to Sutton and Barto [63], there are four elements of RL:

- A policy: "…. defines the way the agent behaves at any time, i.e. a mapping of perceived states to actions". It is denoted by π . For example, if an agent is perceived to be x miles from a traffic light, perform an action to slow down
- A reward function: "…. defines what goals the agent has to achieve in the RL problem in an immediate sense". For example, the reward of running through a red light could be negative in the immediate sense.
- A value function: defines what goals the agent has to achieve in the long run. For example the reward of running through a red light could be negative in the immediate sense, but the value of that action may lead the agent to achieve its overall goal of arriving at its destination on time.
- And sometimes, a model of the environment could be an element of the RL problem

The main goal of a reinforcement learning agent is to "maximize the *total returns (rewards)* it receives over time"[63]. The description and definition of returns (rewards) vary. In some RL algorithms, *discounted returns*[21] are utilized e.g. [74] whereas in others average returns are used [78, 141]. Gloennec [175] defines "the *return function R(t)* as a long-term measure of the rewards." Gloennec [175] describes three expressions used in calculating returns. Generally, to maximize the total returns, the agent has to "look ahead" at the available returns. The discounted return helps in finding out the current

[20] When the agent receives a reward or punishment, based on this, it adjusts its behavior per the driving RL algorithm; hence the reward input indirectly controls the agents' behavior.

[21] Discounted returns are when the impact of future rewards are taken into consideration now.

value (weight) of future rewards. At any given time t, the discounted return for an infinite-horizon model[22] [175] is given as:

$$R(t) = \sum_{n=1}^{\infty} \gamma^{n-1} r_{t+n} \qquad 4.4.1.1$$

where γ the discount factor[23] is between $0 \leq \gamma \leq 1$

When γ is 0, the agent doesn't bother about future rewards, it only concerns itself with the immediate rewards it receives. As γ approaches 1, more weight is given to future rewards in making a decision.

For a finite-horizon model[24], a terminal state and a *period* exists i.e. sequence of actions between the initial state and the terminal state. The return is given as [175]:

$$R(t) = r_t + r_{t+1} + ...+ r_{t+n-1} \qquad 4.4.1.2$$

"*n* is the number of steps before the terminal state" [175].

For an average-reward model, the average of future reinforcements is calculated using:

$$R(t) = \lim_{n \to \infty} \frac{1}{n} \sum_{n=0}^{n} r_{t+n} \qquad 4.4.1.3$$

4.4.2 Markov Decision Process

The modeling of Reinforcement Learning problems are typically based on Markov Decision Processes (MDPs) [63, 162, 178, 180]. A Markov Decision Process satisfies the Markov property. The Markov property simply means that the state of the environment at any given time contains a summary of all states and actions the agent has encountered up to that time and this state is the only required information used in determining the agents'

[22] An infinite-horizon model is one where the sequence of actions is infinite [63, 175]

[23] The discount factor γ is used to give more weight to future rewards, the closer it is to 1, the greater the weights given to future rewards.

[24] A finite-horizon model is one where the sequence of actions is finite [175]

next actions [164]. A system is said to possess the Markov property if the present state can comparatively predict the future states as well as the history of all past and present states. This is known as a *memoryless* process.

Some researchers e.g., [162, 178] note a fundamental difference between MDPs and RL; usually in MDPs, the complete descriptions would have the probability metrics between state transitions, actions and rewards and how these parts affect each other. Also the main "objective of a MDP is to compute an optimal policy". In constrast, RL deals more with approximating an optimal policy during on-line[25] behavior instead of computing optimal policies off-line based on the known probability transition models. Also the objective of RL is to maximize the total rewards received and not to compute an optimal policy[26].

Ratitch and Precup [164] suggest some attributes that can be used to characterize MDPs, amongst which are state transition entropy and controllability. The effect these attributes have on the performance of RL algorithms that use function approximation was shown, especially on the quality of the learnt policies. These attributes were also shown to affect the speed of learning.

4.4.3 Value Functions

This is the "heart" of reinforcement learning. The majority of reinforcement learning algorithms strive to improve or calculate the value function of states. The value function defines the goals the agent has to achieve; it is literally how good a state is or how good a given action is in a state [63].

[25] A detailed description of On-line and off-line behaviors is presented later in this chapter

[26] As noted by Barto [162], maximizing the received rewards doesn't always require the computation of an optimal policy for all states because the agent may not visit all the possible states during its interaction with the environment.

As stated earlier, a policy, π defines the way the agent behaves at any time in the environment. It is a mapping of actions to states. “The value of a state under a given policy is the expected return when the agent starts in that state and follows the given policy”[63].

Given a set of states $S = (s_1, s_2, s_3,..., s_n)$ and a set of actions available in each state: $A = (a_1, a_2, a_3,..., a_n)$

The value of a state $s \in S,$

$$V^{\pi}(s) = E_{\pi}\{R_t \mid s_t = s\} = E_{\pi}\left\{\sum_{k=0}^{\infty} \gamma^{k} r_{t+k+1} \mid s_t = s\right\} \qquad 3.4.3.1$$

Where E_{π} {} is the expected value when the agent follows policy π , t is the time step, r is the expected reward, s is the state and γ is the discount factor between 0 and 1.

4.4.4 On-line and Off-line

RL algorithms are described as either on-line or off-line [63, 78], based on the *qualifier* used. These qualifiers are 1) when the algorithm is being implemented and 2) the mechanism used for updating the internal learning structures of the algorithm [78]. These two qualifiers are elaborated on below.

An RL algorithm is off-line when it is run and tested in a simulator before it is implemented in the real world. An on-line implementation means that the algorithm runs on the actual system in real-time. That is, the learning agent improves its performance as it experiences the environment.

During the on-line updating of the internal learning structure of the algorithm, the values of the mechanisms that keep track of learning are updated immediately the agent tries out an action. For example, with Q-factor [27], the state-action pair is immediately updated after each trial. An off-line update is when these values are updated after a certain number of trials, for example the TD(λ) algorithms[28].

4.4.5 Exploration Vs Exploitation

Exploration is a systematic search in a search space whereas exploitation is the utilization of available knowledge about a given situation to the greatest possible advantage. The success of a RL algorithm is based on how good the agent balances its choice of actions between exploring and exploiting. Researchers [63, 78, 162 and many others], have postulated some action selection strategies for agents to utilize when solving a RL problem. There is neither a good nor bad strategy for choosing whether to explore or exploit existing knowledge during an agent's interaction with the environment. If an agent chooses to exploit its knowledge about its environment in the early stages of a simulation, it risks not getting the potential maximum rewards available. A good strategy is for the agent to explore at a faster rate during the initial stages of the simulation and then exploit the knowledge gained towards the end of the simulation. When an agent tries (explores) a vast range of actions, the agent will be better equipped to make decisions that would maximize its rewards for any given task. Exploration and exploitation are also referred to as non-greedy and greedy strategies.

Some of the most commonly used exploration strategies are as follows:

a) The *P*-greedy policy (Pseudo-stochastic Method) [175]

$$a = \arg\max_{b \in A_X} Q^*(x,b) \qquad 4.4.5.1$$

[27] Q-factors are based on state-action pairs, it is an algorithm of RL based on the work of Watkins [74]

[28] TD(λ) is the temporal difference algorithm, the values are updated after a finite number of steps represented by λ. Sutton [72]

Here the action that produces the maximum return can be chosen with a high probability *P* else an action is randomly chosen. The maximum Q value[29] for a state-action pair produces the maximum return.

b) Pseudo-exhaustive Method: the action that produces the maximum return can be chosen with a high probability *P*, else the action that has the lowest probability of being chosen is utilized.

c) Boltzmann Distribution: The action in any given state is chosen with probability:

$$P(a/x) = \frac{\exp(\frac{1}{T}Q(x,a))}{\sum_{b}\exp(\frac{1}{T}Q(x,b))} \qquad 4.4.5.2$$

T is a positive parameter known as the temperature [63]. According to [63], low temperatures cause the probability of selecting actions with different value estimates to be different, whereas "high temperatures cause the selection of all actions to be nearly equi-probable" [63].

A reinforcement-learning agent has to explore new options and also exploit options that it knows to best suit the current situation. One of the issues with reinforcement learning is having the agent balance its choice between exploring new options and exploiting options it knows would give the greatest rewards for the current situation.

4.4.6 RL General Problems

Assigning credit to the correct decision-making process in a RL problem is one difficulty faced in RL. Credit-assignment can be either of two types; "temporal credit-assignment problem and structural credit assignment problems" [165]. Temporal credit-assignment is

[29] Q values are discussed later in this chapter.

when an agent cannot determine a particular actions contribution(s) to the overall quality of the full sequence of actions to solving the RL problem. Structural credit-assignment deals with multi-agent systems in which determining the contributions of a particular agent to a common task in a RL problem are difficult. Some researchers have postulated some solutions to the credit-assignment problems, Agogino & Tumer [165] show how the temporal and structural credit-assignment problems are the same. Mahadevan [179] compared four machine learning techniques along temporal, structural and tasks credit assignment problems and found differences in the way they handle them. Sutton [72] conceived the temporal difference algorithm to address the temporal credit assignment problem.

RL problems can be of two types, they can be discrete or continuous. An algorithm that generates adaptive controls for continuous processes was formulated by Munos [156]. This is achieved by using finite-element methods to approximate the value function. The learning dynamics as well as the structure dynamics are integral parts of this algorithm [156]. The learning dynamics, known as "*Finite-Element Reinforcement Learning*, estimates the value functions at the vertices of some triangulation defined in the state space, while the structural dynamics defines these triangulations in regions that have an irregular value function" [156].

Kimura & Kobayashi [170] present the stochastic gradient ascent algorithm that deals with problems where the action space is continuous and rewards are delayed. Their method doesn't require a model of the environment and doesn't need to approximate the value function explicitly. They showed that their method learned a policy with less cost when compared with the actor/critic algorithms described in Konda & Tsitsiklis [183].

Smart & Kaelbling [171] introduce an algorithm HEDGER that approximates the value function for continuous state control problems. "HEDGER is based on locally weighted regression where training points close to the query point have more influence over the fitted regression surface" [171] . Their algorithm has an advantage of learning quickly with a small amount of data.

Dayan & Hinton [147] propose the feudal reinforcement learning algorithm that aims to speed up the learning process. This is achieved by breaking the problem into smaller tasks where the high level tasks learn how to set tasks for the lower level tasks. The lower level tasks learn how to maximize the reinforcement received based on the set tasks.

Kretchmar [184] proposes the parallel reinforcement learning algorithm where multiple RL agents interact and learn from the same environment in parallel. Because RL environments are usually stochastic, the agents' experiences would differ and eventually converge on the same value function. By sharing information at intervals, the learning process is accelerated.

Shapiro et al [185] describe "an agent architecture *Icarus* that embeds a hierarchical RL algorithm in a language for specifying agent behavior"[185]. They show an increase in "the learning rate and asymptotic performance and decrease in plan size when background knowledge was introduced" [186].

Baird and Moore [186] derived the *Value and Policy Search (VAPS)* algorithm that can generate new RL algorithms. These newly generated algorithms all guarantee convergence to simple MDPs and POMDPs. They also include modifications to Q-learning, SARSA and advantage learning.

Dietterich and Flann [187], synergistically combine explanation-based learning[30] and RL to produce *Explanation-Based Reinforcement learning (EBRL)* because both learning methods propagate information backward from the goal state towards the starting state. In experiments they performed – comparing batch and online versions of the new algorithm with versions of RL and EBL, they showed EBRL outperformed both. They also believe one of the outcomes of their algorithm (Region-based dynamic programming) provides a possible solution to a major limitation of RL which is, lack of scaling up to large state space problems.

4.4.7 RL Techniques

There have been many notable RL algorithms developed. Amongst the most widely used are Dynamic Programming (DP), Monte Carlo Methods (MC), Linear Programming (LP), Q-learning, TD-learning and Hierarchical Reinforcement learning. Sutton & Barto [63] discuss three broad classes of algorithms: Dynamic Programming, Monte Carlo methods and the Temporal Difference algorithms.

4.4.7.1 Dynamic Programming

In dynamic programming, a model of the world is present and the agent attempts to maximize its rewards. The model includes the transition probabilities between all states. A major limitation of dynamic programming for solving RL problems is its requirements for a model of the environment. Also, for large state-space problems, using dynamic programming would be impractical because of the amount of computations and memory

[30] Explanation-based learning “computes the weakest preconditions of operators and hence propagates information backward from the goal on a region-by-region basis” [187]

needed. Sutton & Barto [63] note that this situation was termed the *curse of dimensionality* by Bellman. Although Gosavi [78] calls RL "an offshoot of dynamic programming and a way of doing dynamic programming within a simulator", he [78] illustrates the differences between RL and classical DP by noting that DP always produces an optimal solution to control and prediction problems whereas RL produces near-optimal solutions. In using DP to solve RL problems, based on the available model of the environment, the transition probability and reward matrices can be generated in a simulator and then dynamic programming algorithm applied to these transition probabilities. This produces an optimal solution to the RL problem. In the case of RL, the reinforcement learning algorithm is applied in a simulator without the model of the environment and this produces the near-optimal solutions.

4.4.7.2 Monte Carlo Methods

Monte Carlo (MC) methods solve the RL problem by averaging the rewards (returns) received over a period of time. Multiple trials which result in experiences generated by the online or offline interaction of an agent with its environment are required for MC methods. It is assumed in MC methods that these experiences are divided into episodes. After the end of an episode, the estimates of the values are then changed. MC methods are based on averaging the complete returns (rewards) received by an agent. The Monte Carlo methods have features of dynamic programming, with the exception that a model of the environment need not be available. A pseudo code for first-visit Monte Carlo methods for estimating the value of a state is presented:

```
Initialize:
        $\pi$ ←policy to be evaluated
        V ←an arbitrary state-value function
        Returns(s) ←an empty list, for all s Є S
    Repeat forever:
        (a) Generate an episode using π
        (b) For each state s appearing in the episode:
                R ←return following the first occurrence of s
                Append R to Returns(s)
                V(s)←average(Returns(s))
```

(Reproduced from Sutton and Barto [63] without permission).

While implementing the first-visit Monte Carlo method, you first of all initialize the policy to be evaluated, i.e., you map available actions to the perceived states in the environment. You initialize the value of a state chosen at random to some random number. For all states of the environment, you initialize the returns received by the agent in each state to zero. You start the simulation, for each state visited by the agent, you note the reward received at that state. At the end of a simulation cycle, the value of a state is determined by the average rewards received in that state, i.e. the total rewards received by the agent while in that state, divided by the number of visits made to that state.

4.4.7.3 Temporal Difference Learning (TD-Learning)

Temporal difference learning (TD-learning) is a combination of the ideas introduced in the Monte Carlo and dynamic programming methods [63]. This method learns from the experience achieved through the agents' interaction with the environment and this learnt knowledge can be updated based on the estimates of other learned estimates without waiting for the final outcome of the reward. It is based on the principle that one does not have to wait until the end of a task to provide an initial estimate for that task. The example provided by Sutton and Barto [63] explains the basis of td-learning as an example. Each day driving home from work, we inherently think of how long the journey will take and thus provide an estimate. While on the journey, if it rains, we immediately adjust our estimates to reflect the fact that we drive slowly when it rains. If we find ourselves stuck in traffic, we keep readjusting our estimates based on the current situation and the completed leg of journey until we complete the journey [63]. In other words, we learn of a new estimate immediately, based on the previous estimate.

Tesauro [64] made the temporal difference learning algorithm famous with his td-gammon learning system. From the work of Tesauro [64], RL was shown to achieve results that other learning methods have not achieved. Tesauro [64] showed that using the TD method of Sutton [72, 73] to solve the RL problem, the game of backgammon could achieve masters' level. This was achieved by the game playing about 1.5 million games against itself and learning from the rewards and mistakes in those 1.5 million games. With this much success of the application of the TD method towards the RL problem in a practical scenario, researchers have high hopes for RL techniques and also believe that RL techniques have great potentials.

There are some skeptics to the work of Tesauro however, Pollack & Blair [66] in their work, try to attribute the success of the temporal difference methodology with the game of backgammon by Tesauro [64] to the domain in which the TD method was used i.e. because of the inherent nature of the game of backgammon. Pollack and Blair [66] carry out a simple hill climbing algorithm in a relative fitness environment on the game of backgammon and claim that any learning method can achieve what Tesauro [64] did.

However, they failed to achieve the same results as Tesauro [64]. A pseudocode for TD(0) for estimating the value of a state is presented:

Reproduced from Sutton and Barto [63] without permission

```
Initialize $V(s)$ arbitrarily, $\pi$ to the policy to be evaluated
Repeat (for each episode):
   Initialize s
   Repeat (for each step of episode):
      $a \leftarrow$ action given by $\pi$ for $s$
      Take action a; observe reward, r, and next state, $s'$
      $V(s) \leftarrow V(s) + \alpha \left[ r + \gamma V(s') - V(s) \right]$
      $s \leftarrow s'$
   until s is terminal
```

SARSA, which is an *on-policy TD*[31] control algorithm is presented:

Reproduced from Sutton and Barto [63] without permission

```
Initialize $Q(s, a)$ arbitrarily
Repeat (for each episode):
   Initialize s
   Choose a from s using policy derived from Q
         (e.g., $\epsilon$-greedy)
   Repeat (for each step of episode):
      Take action a, observe r, $s'$
      Choose $a'$ from $s'$ using policy derived from Q
         (e.g., $\epsilon$-greedy)
      $Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma Q(s', a') - Q(s, a)]$
      $s \leftarrow s'$; $a \leftarrow a'$;
   until s is terminal
```

4.4.7.4 Q-Learning

Q-learning is an *off-policy* TD control algorithm created by Watkins [74]. It is based on the selection of actions in states based on their Q-values. Q-values are a collection of rewards for each state-action pair for each state. The state-action pair that produces the

[31] On-policy is the same as online RL and off-policy is offline RL

greatest reward in each state is chosen[32]. Humphrys [173] notes that the actions available to the agent in each state could be different. In the paper on Q-learning by Cybenko [79], an agent in an unknown environment seeks to maximize the total reward it achieves when starting from any state by choosing actions that would maximize its total rewards in that state. The problem is that the agent doesn't know what actions would maximize its total rewards because if it did, the problem being solved would turn out to be a supervised learning problem [173]. Through trial and error, the agent has to learn to choose actions in each state that would bring about a total maximum reward. A pseudocode for Q-learning is presented: (Reproduced from Sutton and Barto [63] without permission)

```
Initialize $Q(s,a)$ arbitrarily
Repeat (for each episode):
   Initialize s
   Repeat (for each step of episode):
      Choose a from s using policy derived from Q
         (e.g., $\epsilon$-greedy)
      Take action a, observe r, $s'$
      $Q(s,a) \leftarrow Q(s,a) + \alpha\left[r + \gamma \max_{a'} Q(s',a') - Q(s,a)\right]$
      $s \leftarrow s'$;
   until s is terminal
```

According to Gosavi [78], the fundamental concepts of RL are based on the Q-factors (Q-values) and the Robbins-Monro Algorithm. The value function, which is the bedrock of RL algorithms are stored in Q-factors. The value function is based on the Bellman optimality equations as shown below [78]:

$$J^*(i) = \max_{a \in A(i)}[\bar{r}(i,a) + \gamma \sum_{j=1}^{|S|} p(i,a,j)J^*(j)] \qquad 4.4.7.4.1$$

[32] The choice of actions in each state is based on the Q-values as well as the defined policy

- $J^*(i)$ is the ith element of the value function vector associated with the optimal policy
- *A(i)* denotes the allowed actions when in state i
- *p(i,a,j)* is the transition probability of going from state i to state j when action a is performed
- $\bar{r}(i,a) = \sum_{j=1}^{|s|} p(i,a,j)r(i,a,j)$ is the immediate reward expected in state i, if action a is selected in that state, r(i,a,j) is the immediate reward when action a is selected and the system transitions from state i to j because of the selected action.
- S is the set of states in the system
- γ is the discount factor that gives some weight to future rewards

An element of the Q-factor is usually associated with a state-action pair. For any given state-action pair (i,a), the Q-factor is:

$$Q(i,a) = \sum_{j=1}^{|S|} p(i,a,j)[r(i,a,j) + \gamma \ J^*(j)] \qquad 4.4.7.4.2$$

Combining the Bellman optimality equation with the Q-factor equation, we get the relationship between the value function of that state and the associated Q-factors:

$$J^*(i) = \max_{a \in A(i)} Q(i,a) \qquad 4.4.7.4.3$$

For all state-action pairs *(i,a)*, the Q-factors equation can be written as:

$$Q(i,a) = \sum_{j=1}^{|S|} p(i,a,j)[r(i,a,j) + \gamma \ \max_{b \in A(j)} Q(j,b)] \qquad 4.4.7.4.4$$

The above equation is known as the Q-factor version of the Bellman optimality equation for all discounted reward MDPs [78].

The Robbins-Monro algorithm provides estimates of the mean of a random variable from samples of it [78]. By averaging the samples, we can get the mean of the random variable. According to [78], "let the i^{th} independent sample of a random variable X be s^i and the expected value (mean) by $E(X)$, with probability 1, the estimate of the mean is:"

$$\frac{\sum_{i=1}^{n} S^i}{n} \qquad 4.4.7.4.5$$

This estimate of the mean, tends to the real value as $n \to \infty$ "according to the laws of large numbers" [78]

$$E[X] = \lim_{n \to \infty} \frac{\sum_{i=1}^{n} S^i}{n} \qquad 4.4.7.4.6$$

These samples are usually generated in a simulator. From the averaging process shown above, [78] derived the Robbins-Monro algorithm. If X^n denotes the estimate of X after obtaining n samples:

$$X^n = \frac{\sum_{i=1}^{n} S^i}{n} \qquad 4.4.7.4.7$$

$$X^{n+1} = \frac{\sum_{i=1}^{n+1} S^i}{n+1} \qquad 4.4.7.4.7.1$$

$$= \frac{\sum_{i=1}^{n} S^i + S^{n+1}}{n+1}$$

$$= \frac{X^n n + S^{n+1}}{n+1}$$

$$= \frac{X^n n + X^n - X^n + S^{n+1}}{n+1}$$

$$= \frac{X^n (n+1) - X^n + S^{n+1}}{n+1}$$

$$= \frac{X^n (n+1)}{n+1} - \frac{X^n}{n+1} + \frac{S^{n+1}}{n+1}$$

$$= X^n - \frac{X^n}{n+1} + \frac{S^{n+1}}{n+1}$$

$$= (1 - \alpha^{n+1})X^n + \alpha^{n+1}S^{n+1} \qquad 4.4.7.4.7.8$$

If $\alpha^{n+1} = 1/(n+1)$

Thus:

$$X^{n+1} = (1 - \alpha^{n+1})X^n + \alpha^{n+1}S^{n+1} \qquad 4.4.7.4.8$$

This is known as the Robbins-Monro algorithm. When $\alpha^{n+1} = 1/(n+1)$, the algorithm becomes direct averaging. The Robbins-Monro proof shown above, was reproduced from [78] without permission. α is known as either the step size or the learning rate, it guarantees convergence to an optimal solution.

Combining the Robbins-Monro Algorithm with the Estimates of Q-factors produces the optimal Q-factors without knowing the model of the environment. It has been shown that every Q-factor can be expressed as an average of a random variable [78]. Recalling the Q-factor in Bellman's equation:

$$Q(i,a) = \sum_{j=1}^{|S|} p(i,a,j)[r(i,a,j) + \lambda \max_{b \in A(j)} Q(j,b)] \qquad 4.4.7.4.9$$

$$= E[r(i,a,j) + \lambda \max_{b \in A(j)} Q(j,b)] \qquad 4.4.7.4.9.1$$

=E[SAMPLE]

According to [78], E is the expectation operator of the random variable in equation 3.4.7.4.8.1. Robbins-Monro algorithm can be used to estimate the Q-factors if samples of random variable are generated in a simulator. If the Robbin-Monro algorithm is used, the Q-factor equation for each state-action (i,a) pair becomes:

$$Q^{n+1}(i,a) \leftarrow (1-\alpha^{n+1})Q^{n}(i,a)+\alpha^{n+1}[r(i,a,j)+\lambda \max_{b\in A(j)} Q^{n}(j,b)] \quad 4.4.7.4.10$$

The above equation does not include transition probabilities. With this algorithm, the optimal Q-factors can be generated in a simulator without knowing the probability matrix of the underlying Markov chain [78].

There have been many attempts to refine the Q-learning algorithm to produce better performing algorithms. Guo et. al. [188] produces an algorithm, *SA-Q-learning* that converges faster than Q-learning. Their algorithm also balances the exploration and exploitation choices made by the agents, and its performance isn't degraded because of excessive exploration.

Wiering [189] developed the fast online Q(λ) algorithm based on the fact that the updates to Q-values can be postponed until needed. In this algorithm, TD(λ) methods were used to accelerate Q-learning. The fast online Q(λ) learning algorithm has a complexity of *O*(|A|) per update.

Peng [190] combine Q-learning with TD(λ)-learning algorithms to produce a faster algorithm. The new algorithm also eliminates the non-markovian effect of coarse state-space quantization.

Dearden et. al. [191] produces the Bayesian Q-learning algorithm where a Bayesian approach is used to maintain the agents' value estimates of states. Probability distributions of the Q-values are used to compute these value estimates of actions that provide the best balance between exploration and exploitation.

4.4.7.5 Hierarchical Reinforcement Learning (HRL)

Hierarchical Reinforcement learning (HRL) reduces the complexities of a decision making process by breaking down a large problem into a hierarchy of smaller problems. Parr & Russell [139] note that these complexities could be reduced from an exponential size to a linear size of the problem. In HRL, high-level activities are usually decomposed into lower-level activities. As noted by Dietterich [140], "the aim of hierarchical reinforcement learning is to discover and exploit hierarchical structures within a markov decision problem." Parr & Russell [139] describe a learning technique that utilizes prior knowledge in finding solutions. They use hierarchical abstract machines (HAMs) in their solutions. Constraints are placed on the policies available to the learning agent by the HAMs. At each state of the learning process, a HAM - a program, restricts the actions available to the learning agent. A HAM is akin to one of the tenets of the Context-based Reasoning technique created by Gonzalez & Ahlers [7]. An example of a HAM would be the classical example provided by Gonzalez et al. [9], that is, it is not possible for a tire blow out to occur when a car is waiting in a traffic light and as such an agent wouldn't consider a tire blow out event when in a traffic light scenario. As such, the HAM for this scenario would constrain the actions available to the agent by excluding an action for the tire blowout event. Parr & Russell [139] note that "machines for HAMs are defined by a set of states, a transition function, and a start function that specifies the initial starting state of the machine." There are four types of machine states in a HAM: the action states, where actions are executed in the environment; the call state, where the execution of a subroutine is initiated; the choice state, where the stochastic selection of the next machine state is carried out; and finally the stop state, where the execution of subroutine is halted and control is returned to the calling state. Figure 4.5 shows these states.

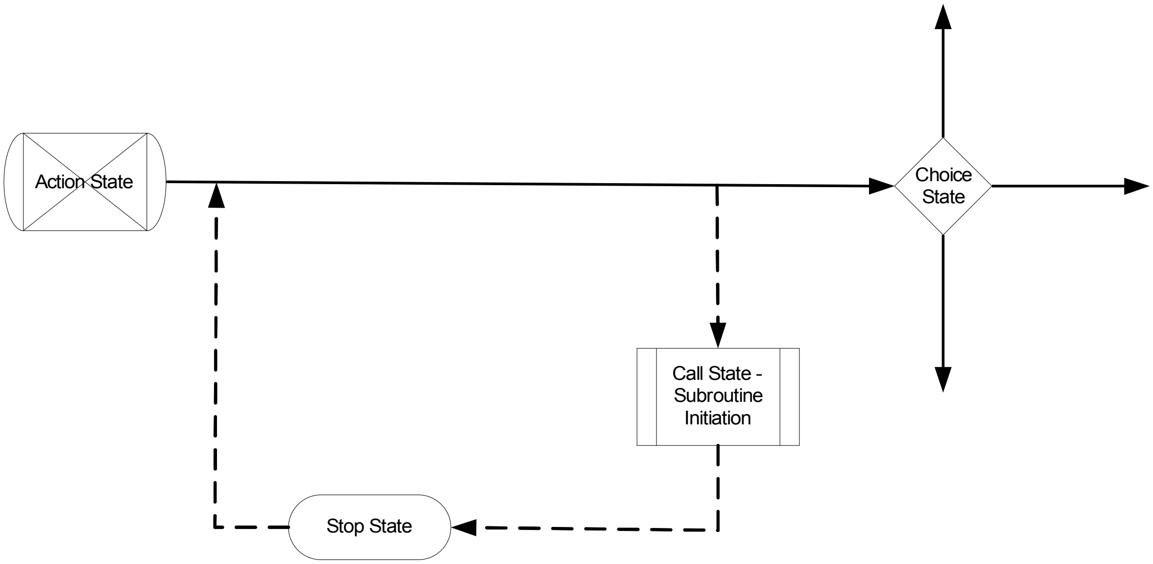


Figure 4.5 Showing the Four Machine states for HAMs. The dashed line shows calls to a subroutine.

The transition function determines what the next machine state should be. It is based on the current state of the agent and some features of the agents' environment. The start function defines a HAM, i.e. the initial machine where execution begins and the closure of all machines that can be reached from this initial machine [139].

In RL, the constraints placed by HAMs, can narrow the focus of exploration of the state space. This technique reduces the exploration phase of the learning agent and thus provides faster learning for the agent because the state space is reduced. Parr and

Russell [139] introduce the HAMQ-learning algorithm, which is a variation of Q-learning [74, 79]. According to them [139], “the current environment state, *t*; the current machine state, *n*; the environment state at the preceding choice point, s_c; the machine state at the preceding choice point, m_c; the choice made at the previous choice point, *a*; the total accumulated reward and discount rate since the previous choice point, r_c and β_c; an extended Q-table, *Q([s,m],a)* indexed by an environment-state/machine-state pair and by an action taken at a choice point” are all kept track by a HAMQ-learning agent.

For every action in the environment, a transition from state *s* to state *t* is made. For each transition, the observed rewards *r* and discount factor *β* and updated by the HAMQ-learning agent as follows:

$$r_c \leftarrow r_c + \beta_c r \quad and \quad \beta_c \leftarrow \beta\beta_c \qquad 4.4.7.5.1$$

“Thus for each transition to a choice point, the agent does:”

$$Q([s_c, m_c], a) \leftarrow Q([s_c, m_c], a) + \alpha\,[r_c + \beta_c V([t,n]) - Q([s_c, m_c], a)], \qquad 4.4.7.5.2$$

$r_c \leftarrow 0,\ \beta_c \leftarrow 1$

Dietterich [140] presents a learning algorithm which is an extension to Q-learning known as *Hierarchical Semi-Markov Q (HSMQ).* He showed that a task using this algorithm can converge to a recursively optimal policy. HSMQ has a goal of finding a recursive optimal policy. Dietterich [140] states that a recursively optimal policy is an assignment of policies to each subtask in such a way that the policy is optimal for all policies assigned to all of its dependents. It is “a kind of local optimality that has no guarantees on the quality of the overall policy.”[140] The idea for the HSMQ algorithm is that for each subtask p, the Q function Q (p,s,a) is learnt, this Q function is the expected total reward of performing the subtask p starting in state s, executing an action a and following the optimal policy.

Dietterich [140] describes the algorithm for each subtask as:

```
function HSMQ(state s, subtask p) returns float
    let TotalReward = 0
    while p is not terminated do
        choose action a=$\pi_x(s)$ according to exploration policy $\pi_x$
        execute a.
            if a is primitive, observe one-step reward r
            else r := HSMQ(s,a), which invokes subroutine a and
                returns the total reward received while a executed.
        TotalReward := TotalReward + r
        Observe resulting state s’
        Update Q(p,s,a) := (1 - $\alpha$ ) Q(p,s,a) + $\alpha$ $[r + \max_{a'} Q(p,s',a')]$
    end //while
    return TotalReward
end
```

Dietterich [140] notes that the HMSQ learning algorithm solves the hierarchical reinforcement learning problem by treating it as “a collection of simultaneous, independent Q learning problems”. This doesn’t provide a representational decomposition of the value function, and as such, the value function of each subtask is represented and learned independently. This is not good, it would be better if some “sharing and compactness” in the representation of the value function exists. Dietterich developed the MAXQ value function which does this [140, 145].

Dietterich [140] notes that the value function of many subtasks don’t depend on all the state variables in the original MDP and thus, there are three forms of state abstraction that can be applied within the MAXQ value function decomposition. These are: irrelevant variables, funnel abstractions and structural constraints. These abstractions are necessary for the reduction of the memory needed to store the value function as well as the amount of experience needed to learn the value function.

“A state variable is irrelevant for a subtask if the value of that state variable never affects either the values of the relevant state variables or the reward function.

A funnel action is an action that causes a larger number of initial states to be mapped into a small number of resulting states.

Structural constraints concerns implication relationships between a child task and its parent task.” [140]

In analyzing the design tradeoffs in hierarchical reinforcement learning, Dietterich [140] notes that a “recursively optimal policy can be far from being optimal”; the HAMQ algorithm by Parr & Russell [139] learns a hierarchical optimal policy[33] and as noted by Dietterich [140], for an agent to learn a hierarchical optimal policy, information sharing between subtasks must exist. Although, more state abstraction and reuse of subtasks can be achieved through a recursively optimal policy than through a hierarchical optimal policy, hierarchical optimality is usually better.

Hierarchical Suffix Memory (HSM) Reinforcement Learning was developed by Hernandez-Gardiol & Mahadevan [148]. They note that perceptual aliasing, a situation where the same observations are generated by different real-world states, is a problem in the solution of RL tasks [148]. They suggest the addition of memory about past events to address this. They show that when past experience is considered at some task-appropriate

[33] “A hierarchical optimal policy is the best possible policy for the constraints on an imposed hierarchy” Dietterich [140]. The policies used by the subtask might not be optimal.

variable resolution under perceptual aliasing, the speed of learning can be increased for problems with long sequences of decision making. A limitation to their method is that the past experience used is limited to the histories for each level of abstraction.

There are many other hierarchical reinforcement learning algorithms, for example the *hierarchical distance to goal (HDG)* method by Kaelbling [146], where an agent acts upon a partitioned environment with centers known as 'landmarks'. Low level actions are utilized to move towards the goal if an agent is sensed to be in the same partition as the goal; otherwise high level actions are used to determine the landmark closest to the goal.

Lane & Kaebling [149] propose a method were partial plans are developed over a hierarchical region where each plan is a representation of some knowledge on the achievement of a sub-goal within its region.

Bakker & Schmidhuber propose the *HASSLE (Hierarchical Assignment of Subgoals to Subpolicies LEarning)* algorithm where sub-goals are automatically discovered through high-level policies and learning to specialize on these sub-goals is achieved through low-level policies [150]. In deterministic and stochastic large MDPs, the HASSLE algorithm performed better than some other RL algorithms [150].

Bernhard [151] proposes the *CQ* algorithm that automatically generates a hierarchy of sub MDPs using state variables to decompose the MDP. Bernhard [152] proposes the HEXQ algorithm to solve multi-dimensional MDPs by constructing a multilevel hierarchy of interlinked subtasks. This is done without having apriori knowledge of the model. The MDP is automatically decomposed. The choice of representation of variables, the temporal relationship between the variables and the type of constraint placed on the stochasticity of the problem all affect the efficiency and effectiveness of HEXQ decomposition [152]. Bernhard [153] tries varying degrees of model resolution in approximating hierarchical decomposed MDP that are already state abstracted.

McGovern & Sutton [154] analyze the advantages of macro-actions in reinforcement learning as it relates to an agents' exploratory behavior and speed that the propagation of value information is carried out by the learning process. Their results show the effects of both to be significant with a much larger effect of value propagation.

McGovern et. al. [155] in their approach, present 'options' where both high and low-level decisions are treated the same way during problem solving. Traditionally, while solving SMDPs, the sub-SMDPs are solved in parallel and their solutions merged without taking into account the effect of actions available and transition probabilities of the neighboring sub-SMDPs. Gang & Mahadevan [157] present an approach that resolves this. With their approach of solving SMDPs, the sub-SMDP takes the different modes of interaction between them and their neighbors into account. After the sub-SMDPs are solved, the resulting policies are combined using a greedy algorithm for the problem [157]. They show that their method outperforms traditional 'flat'[34] RL algorithms in a 12-machine manufacturing transfer line, in terms of speed. Their method also performed better than some heuristics currently being used in manufacturing transfer lines.

Singh [158] presents a *hierarchical DYNA (H-DYNA)* algorithm which is an extension of Suttons [63] DYNA architecture. H-DYNA learns the hierarchy of temporal

[34] A 'flat' RL algorithm is an algorithm that seeks to solve the RL problem without decomposing the problem into sub-problems and without any special refinement of the function approximation method used for the value function. A simple Q-learning algorithm can be considered a flat RL algorithm.

abstract models of the environment that can be used in solving stochastic control problems and can also transfer learnt knowledge across different tasks.

Thrun & Schwatz [159] propose the *SKILLS* algorithm. The main idea behind the SKILLS algorithm is the reduction of all possible actions an agent can take in any given situation. This is done by giving the agent high-level skills that can be applied in many situations. The skills are represented by subpolicies the agent can follow for many timesteps.

Wiering & Schmidhuber [160] propose the *HQ-learning* algorithm which is a hierarchical extension of the Q-learning algorithm. “It is based on an ordered sequence of subagents that learn to identify and solve the markov subtask of the overall task.” They show that the HQ-learning algorithm can solve complex tasks that Q-learning is incapable of solving [160].

Goel & Huber [161] propose a technique where a RL agent discovers subgoals by searching a learned policy model for states exhibiting some types of structural properties. Barto & Mahadevan [141] discuss some hierarchical reinforcement algorithms and the limitations of these methods. They [141] propose ways of extending these algorithms to address multiagent coordination, concurrent activities, etc. and list existing huddles facing hierarchical reinforcement learning.

From the above literature, it can be seen that there are many algorithms for hierarchical reinforcement learning currently in use. These algorithms tend to address one or more aspects of the shortfalls of hierarchical reinforcement learning. However different these algorithms look, they all share the fundamental structure of hierarchical reinforcement learning, which says that for any given complex task, the task can be decomposed into subtask that would reduce the complexity of the problems. Stated differently, they all share the goal of hierarchical reinforcement learning which is the discovery and exploitation of hierarchical structures within a complex markov decision problem.

As noted earlier, on first look at hierarchical reinforcement learning, one would assume it is a paradigm that models tactical behavior like CxBR, with learning included in it. In reality, the only similarity between hierarchical reinforcement learning and context-based reasoning techniques is in the decomposition of complex tasks into smaller subtasks; in CxBR, a complex task is thought of in terms of contexts and the appropriate actions an agent would exhibit in each context is addressed in it. In HRL, the complex task is decomposed into a hierarchy of abstract machines. Each abstract machine calls the subtasks (subroutines) and the subtasks can all operate in parallel towards a solution. On closer look, the dissimilarities between both techniques abound. In HRL, the lower hierarchy must return control back to the calling function whereas in CxBR as soon as a change in situation is noticed, control can be transferred to a new context that correctly identifies the new situation, it doesn't matter if control comes from a major-context, a sub-context or a sub-sub-context. In CxBR, the flow of control of an agent's action is intuitive; meaning as soon as there is a recognized change in situation, much work isn't required to identify the new context. In HRL however, the flow of control of an agent's action is based on the observed or sensed rewards from the environment. In CxBR, localized optimality[35] is directly proportional to global optimality[36] whereas in HRL, localized optimality is not always directly proportional to global optimality. Another

[35] Localized optimality is the optimal value of an attribute or action in a given context, sub-context or sub-procedure towards the goal of the context, sub-context or sub-procedure.

[36] Global optimality is the optimal value of an attribute or action in a given context, sub-context or sub-procedure towards the overall goal.

distinguishing feature between CxBR and HRL is the fact that contexts can be refined in terms of their attribute values, transition criteria between contexts, etc. whereas in HRL, once a hierarchy is discovered, the boundaries are fixed.

4.4.8 RL Applications

There is a significant body of literature on reinforcement learning. The existence of these publications is fueled by the need to make the reinforcement learning algorithms perform better. Reinforcement learning algorithms are said to perform better if they find a solution to the reinforcement learning problem faster by converging to an optimal solution quicker. Most researches are focused on achieving faster algorithms for control and predictions tasks for example [188, 190]. In most cases, a RL algorithm is said to perform better if it converges to a solution in the shortest possible time with the smallest amount of computation steps. Overall, most investigations in reinforcement learning are focused on making the learning agent learn some value function or utility of a state or an action faster. There are some criticisms to the study of RL, amongst which are the lack of practical applications of RL. Most works on RL are theoretical, and most examples are simulated. Unfortunately, in terms of real world applications of reinforcement learning, only a few successful applications have been developed, amongst which include [67, 68, 69, 64, 70, 71, 167, 168]. As pointed out by Pratt [81], RL may not be a good framework for describing animal intelligence, most works carried out in RL are on grid worlds with the learning agent moving in a north, south, east, west fashion. Pratt [81] also points out the technical hurdles in RL, which include "Curse of Dimensionality and the slow learning with Primitive Actions" [81]. In most RL algorithms, the learning agent doesn't take into account behavioral variances caused by emotions, contexts, etc.

There are many papers on the future of RL, Sutton [65] talks about the current state of RL and also about the future, stating that researchers should focus on the structures that enable value function estimation and the possibility of a machine constructing the features that affect learning and other structures automatically, instead of people doing them [65]. Sutton [65] also talks about the idea of a developing mind as currently being studied in psychology, called constructivism being part of the future.

Although there are some limitations to what reinforcement learning algorithms can do, it proffers the best solution to the problem described in chapter 2 when compared with the other two classes of machine learning algorithms. In the next section, a brief comparison between all three classes of machine learning algorithms is made with a focus on our defined problem.

4.5 Comparison of the Three Machine Learning Groups Towards the Enhancement of Tactical Models

It has been established that machine learning techniques are used to acquire knowledge for modeling tactical decisions. Usually, this knowledge is transformed from their raw states to a form that can be understood by the modeler. In most cases, supervised learning techniques are used to acquire expert knowledge. Based on some presented examples and the conclusions arrived at with these examples, a learning technique would acquire this knowledge and model the decision making process of an expert. According to Henninger [126] inductive, connectionist, case-based and analytical methods, SOAR chunking and interactive machine learning techniques can be used to accomplish the acquisition and transformation of raw knowledge. Inductive techniques can be supervised learning or unsupervised learning [126]. Connectionist techniques involve neural networks which are usually supervised learning, pattern recognition and Hopfield networks which are unsupervised learning [126]. Case-based and analytical methods usually keep cases that identify problems and their solutions based on the similarities between the current problem and past cases in the case library.

Unsupervised learning techniques are typically used in classification problems where no goals exist. In the refinement of a human behavior model, the goals of the agent exist and based on this goal, the refinement process occurs. The singular reason of the existence of a goal precludes the use of unsupervised learning techniques in this research.

Observational learning, also a supervised learning technique is used to acquire implicit knowledge from expert. When combined with CxBR, observational learning in CxBR captures expert knowledge and can automatically build contexts based on the captured knowledge [138]. In the work by Fernlund et al [10, 138], building agent behaviors automatically was discussed. This involved building the transition from one context to another through observation. The underlying motivations for human behavior modelers to use observational learning in acquiring knowledge from a SME are exactly the same motivations for this research with an additional motivation here of expanding the acquired knowledge beyond the knowledge of the expert. For example, lack of explanation for experts' implicit actions and many others. There are limitations to the use of observational learning and thus, supervised learning techniques to refining and enhancing knowledge. This is because the acquired raw knowledge and the final model are based on input/output pairs from the expert actions and the results of these actions.

The goal is always the result of the action which is obtained directly from the expert which is limited to a particular action.

There are major differences to the approaches and expected results when knowledge is acquired through observation in a CxBR model and when this knowledge is refined through RL.

While the work on observational learning suggests 'watching' the expert or "manned vehicle as it performs that behavior in battlefield situations similar to that to be seen by the model" [10], the agents in this research interact with the environment. In this research, the agent is an actor in the environment; it performs actions on the environment that causes a state change in the environment. Whereas in observational learning in CxBR, the agent is not an actor, it doesn't perform any action in the environment and cannot effect any state change in the environment. In reinforcement learning in CxBR, the agent learns from the reinforcements or punishments it receives as a result f its actions; whereas in observational learning in CxBR, the agent learns from the reinforcement or punishment received the actions of others. The latter limits the range of experiences that the agent can experience.

Another issue with observational learning is the same one associated with expert questions and answers sessions. The expert performing an observable 'act' will only perform what it knows. This is a great limitation to the agent learning by observing this expert. Reviewing the work by Sidani [58], an agent tries to learn implicit knowledge by observing an expert perform actions. The work failed to address the situation when the agent watches two or more experts react differently to the same scenario! In a driving scenario, if a pedestrian suddenly crosses a roadway, expert driver #1 might immediately hit the brakes, while expert driver #2 might swerve to the right to avoid hitting the pedestrian, likewise expert driver #3 might swerve to the left. If an agent is learning by observing these expert actions, which does the agent retain and apply when faced with a similar situation? [37]

In summary, the relationship between this research and that of most researches in observational learning in CxBR is minimal only in the sense that they both don't advocate the 'hard-coding' of transition rules between contexts based on knowledge acquired through question and answer sessions. Observational learning in CxBR acquires the knowledge required for modeling and thus generates its actions and transition rules by observing the expert operate and transition between contexts, alternatively, this research builds on the knowledge acquired through Q&A sessions and / or observation. The model built is then refined by the agents' constant interaction with a simulated environment. This enables the agent to learn from its own experience and apply what is learnt to the model – thus refining it to perform better. While observational learning has a disadvantage of having the agent learn only what it observes, reinforcement learning allows the agent to explore different actions and transitions between and within contexts thereby allowing for flexibility for events that where never planned to occur or never thought of by the expert. This is the reason why this research was carried out using Reinforcement Learning techniques.

4.6 Chapter Summary

In this chapter, the three classes of machine learning techniques were presented. Examples of the various machine learning techniques were also presented. Emphasis was

[37] It can also be argued here that the agent would generalize, but there is always going to be an optimal action to take, and generalization might not take this into effect.

placed on reinforcement learning and the techniques used in solving reinforcement learning problems. A comparison on the three classes on machine learning techniques was done based on the problem being researched and a justification was made as to why reinforcement learning techniques were used in this research.

CHAPTER 5: CONCEPTUAL APPROACH

5.1 Introduction

The approach used to address the problems inherent in the knowledge acquisition process utilizes experiential learning (reinforcement learning). The approach used in this research is based on the popular sayings “hindsight is clearer than foresight”. Our approach to breaking the SME knowledge barrier is to refine / enhance conventionally-built models through reinforcement learning. This process consists of subjecting a model developed with the help of a SME to several different scenarios in a simulator. If the model embodied in an agent successfully completes the mission, decisions made are reinforced and subsequently subjected to a new scenario that is a modified version of the last one. If the agent fails, changes are made to the model and the same scenario is re-run. This continues until the model successfully accomplishes the mission. Context-based reasoning is used as the basis of the model.

This research is not the first attempt at synergistically combining reinforcement learning and contexts. Wan & Braspenning [192] propose an extension to the RL framework to incorporate the role of contexts in solving RL problems. They had encouraging results in an experiment where the agent had to learn to intercept a moving target from any position in a path-finding problem. The difference between their work and this investigation is that this investigation focuses on enhancing the overall model of the agent through refining the individual contexts the agent encounters.

Bridle & McCreath [144, 193] propose a method for learning transition models in a RL agent. This reduced the number of trials required by the agent in finding an optimal policy. This was done by taking the agents' context into consideration.

Balkenius & Moren [119] present a computational model for context processing that learns context representations from the sequence of attentional shifts between environmental stimuli.

Balkenius & Winberg [121] note that in RL, policies for states are learnt individually without taking into consideration the similarities between different states. They state that it would be good if actions learnt could be generalized amongst states and that the generalization could be introduced in the RL algorithm in many ways. One way they suggest is to divide the input from the state into two parts – one part for the situation (context) and the other part to control the actions. They believe this would cause learning to generalize for similar states as well as similar contexts. This will cause the roles of state and context to be symmetric [121]. They formulate *Contextual Reinforcement Learning* that achieves this in some experiments carried out. They note the limitations of their method to include further investigations on the "relationship between stimulus and context generalization – how a context influences the generalization of an action to similar states and also how the learning history influences it" [121]. A major limitation to their approach is the lack of relationship between "the concepts of a context to that of a goal". This research touches on the latter and enhances a context definition based on the mission goal.

5.2 RL-CxBR Integration

Recalling the components of CxBR in Chapter 2 and those of RL in Chapter 4, it can be seen that the components common to both architectures are the agent, the environment and the model as shown in Figure 5.1. The model in CxBR are the predefined contexts and their transitions therein. Enhancing this model to address the shortcomings of the knowledge acquisition process is what this research is about. Other components are necessary for the enhancement process to occur. Modifications to some existing components would also facilitate the enhancement process. Before the components of the RL-CxBR architecture are described, some questions must be answered to illustrate the functionality of these components in the enhancement process.

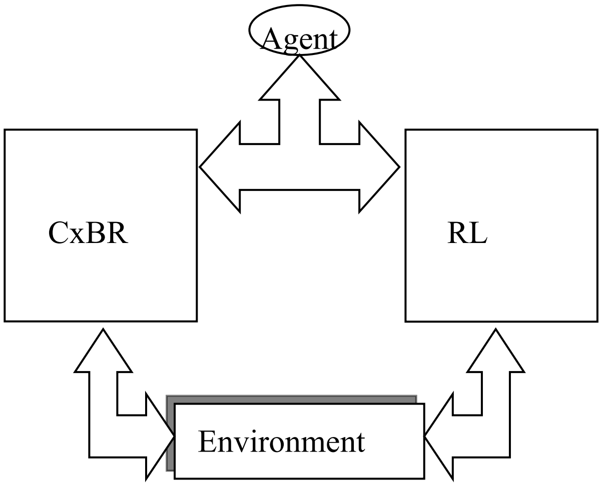


Figure 5.1 RL-CxBR Block Diagram

The answers to these questions are needed to define a conceptual approach to the problem and thus provide a formalized algorithm and flow of activities. These questions are presented below:

1. How does one implement the enhancement process – does one seek to learn contexts

with a reinforcement learning agent or seek to incorporate a reinforcement learning algorithm within the CxBR agent?

2. How are the rewards presented to the agent?

   2.1. Who sets these rewards? If the expert sets them, why will there be any improvements on the agents' behavior based on the rewards set by the same expert? In other words, what effect would the reward have on the overall performance of the model if it is defined by the same expert whose limitations we are trying to break through?

3. Can an active context be enhanced during the enhancement process in realtime?

   3.1. If an active context is enhanced, how would the enhancement occur, would the agent know of the enhanced or refined attributes / values immediately?

4. What are the criteria for stopping the enhancement process?

   4.1. Are these criteria valid for the enhancement process of the whole model or only for one Context within the model?

5. How does the agent know the correct results from actions it performs as defined by the environment? Note that if the outputs for a given action are given, it is reduced to a supervised learning problem.

6. What actions are available to the agent at any given time?

7. Should the entire environment be visible to the agent at all times?

8. Reinforcement learning seeks to learn the behavior of an agent in different states of its environment. CxBR addresses the behavior of an agent in an environment based on the context. How should the concepts of states and contexts be represented? Does one represent a context as a state or a context as a group of states?

9. How should generalization be incorporated into the enhancement process? For large state-space models, would each action be executed / attempted to properly enhance/refine the model? Or will a generalized enhancement approach be sufficient?
10. Because contexts in a CxBR control an agents' behavior based on a predefined model of the environment, how would the agent explore and exploit this model? Would the agent start the enhancement process by exploiting the knowledge it has of the environment even though this knowledge might be wrong or incomplete? Or would it forget all knowledge it has about the environment (all information defined in the context) and start exploring the model?
11. Would the enhancement process deal with the issue of dynamic goals in the agents' environment? If so, how?
12. How would the learning mechanism address any conflicting knowledge present in the system?

Answers to the above questions are presented in the following sections.

5.2.1 Representation of the Enhancement Process

In our approach to breaking the SME knowledge barrier, it should be noted that the knowledge to be enhanced is already defined and organized into contexts. Therefore, creating a model from scratch through reinforcement learning is not relevant. This is because it is the authors' opinion that attempting to learn contexts from scratch in a reinforcement learning problem would produce a method for organizing the expert knowledge – which isn't the objective of this research. Also, by organizing expert knowledge irrespective of the quality of the knowledge will assist the reinforcement learning algorithm to converge faster. This should be another topic for investigation.

Incorporating a reinforcement learning algorithm within the CxBR architecture is most appropriate for the problem at hand. This is because having the agent explore new actions within a context and its transitions between contexts can enable the enhancement of the model. By having the agent perform actions within a context in a simulator and learn from the rewards and punishments it receives, the agent would know the best action to perform when put in the same situation in the simulator (or real world). Knowing the best action in any context (situation) will enable the enhancement of the context.

Mathematically, an enhanced Context $C_i^{'}$ is represented as:

$$C_i^{'} \leftarrow \boldsymbol{R}\, C_i \qquad 5.2.1.1$$

Where $\boldsymbol{R}$ is the reinforcement learning algorithm applied to the context.

5.2.2 The Reward Function

The design of the reward function is essential for true and efficient learning to occur. The reward function essentially directs the learning agent on what behavior to reinforce and which to discard. It does not, however, tell the agent what is right or wrong. For the enhancement process, the reward function directs the agent on the contexts to enhance and the optimal values and attributes within these contexts. In a typical mission, the reward function will be defined by the achievement of the goal and sub-goals of the mission. Rewards will be attached to the constraints of the mission. For example, if an agent completes a task successfully and on time[38], a positive reward could be attached to the achievement of this goal. If the agent completes the task successfully but late, no reward is given. If the agent is unsuccessful at the task, a negative reward[39] is given.

The reward function will be defined by the SME, the knowledge engineer or the application system developer. Allowing the SME set the reward might appear counterintuitive at first because breaking the SMEs knowledge barrier is what this research is all about. However, on a closer look, having the SME define the reward function will highlight problems with the model to him or her. Furthermore, the reward function neither contains the details of the actions nor the transition rules within the model. For example, assuming a SME defines a maximum speed limit of 30 mph in a context and wants an agent to drive to a meeting 50 miles away. If this SME defines a reward function that rewards the agent for arriving at the meeting within an hour, the enhancement process identifies this error because the agent would never achieve its goal. Another attribute of the reward function that is evident in the enhancement process, is the identification of expert implicit knowledge, also, the lack of SME knowledge in any given mission will be exposed.

Mathematically, the reward function $\Re$ is a function of the goal, the sub-goals and the constraints of the mission, i.e.:

$$\Re = (G, sG, C_o) \qquad 5.2.2.1$$

From equation 5.2.2.1, the reward an agent receives is constrained by the mission goal, the immediate sub-goals of the agent as well as the overall constraints placed on the

[38] On time means within the allowed timeframe

[39] The concept of negative reward is utilized in the reinforcement learning community. it means punishment

mission. Placing a reward on a mission without taking the constraints of the mission into account will be counter productive as the rewards received by the agent will not be a true representation of the overall mission. For example, if a constraint exists that an agent driver must arrive at its destination with its car at a given time and the agent arrives at the destination at the given time without the car, the agent should not receive a positive reward.

5.2.3 Enhancing/Refining an Active Context

The enhancement/refinement of an active context is possible in real-time in a simulator. A copy of the active context will be created and placed in a repository known as the *Enhanced / Refined Context Repository*. The function of this repository is to hold copies of all contexts that have undergone some form of modification. As soon as a context becomes active, a copy is created in this repository. As the agent explores with different values and settings within a context, these are reflected in the copy in the repository alongside the rewards received for each setting.

Mathematically, for an active Context $^{A}\boldsymbol{Ci}$ that exists in the context library, a copy ***CCi*** exists in the repository. The copy ***CCi*** contains the various values and settings explored and the corresponding rewards.

$$CCi \Leftarrow \left\{(s_i, a_i), (s_i, a_j), \ldots..(s_n, a_n) \mid (r_i, r_j, \ldots\ldots, r_n)\right\} \qquad 5.2.3.1$$

Where s_i is the state, a_i are the actions and r_i are the rewards obtained for performing action a_i in state s_i.

5.2.4 Enhancement Process Stopping Criteria

The criteria for ending the enhancement process involve the agent receiving the maximum reward available and the mission goal being achieved. The actual value of the maximum reward is not known to the agent. The agent has to determine this through its interaction with the environment. The enhancement process is two-fold: the enhancement of a given active context when the context is isolated and the enhancement of the entire model. The former can lead to the agent performing better when run in the simulator with the individual contexts active, whereas the later would have the agent perform better as a complete model[40]. A criterion for determining this involves when the change in reward received ε is zero or is negligible after a given number of time steps in the simulator. Another criterion is when the total reward received begins to decrease.

In some cases, these two criteria can lead to early stoppage of the enhancement process. This will produce a model that is not completely enhanced. This problem is alleviated by introducing a function that compares the current calculated reward $\Re$ with that of the generalized reward $g\Re$.

Mathematically, the criteria for stopping the enhancement process are:

1) Change in reward ε is zero:

$$\left\{\begin{matrix} \Re_i - \Re_j = \varepsilon_i \to \varepsilon_j \\ \varepsilon_i - \varepsilon_j \to 0 \end{matrix}\right\} for\ g\Re < \Re \qquad 5.2.4.1$$

2) Total rewards received $\Re_T$ begin to decrease:

$$\Re_T \to 0, for\ g\Re < \Re \qquad 5.2.4.2$$

5.2.5 Available Actions

CxBR was designed to limit the actions available to an agent in any situation. This is one of its many advantages. This makes CxBR an intuitive and efficient modeling tool. This

[40] The effects of both methods are investigated further in different scenarios and their results presented in chapter 6.

also places a constraint on the learning capabilities of CxBR. In this investigation, the constraints placed on limiting the actions available to an agent in any context will be softened during the learning phase of the algorithm. This is needed to enable the agent to explore its environment completely and choose actions that would maximize its total reward. To make this happen, an additional module is added to the environment. This module is known as the *action-base* module and its sole purpose is to store all perceivable actions needed in the environment of the agent. This is analogous to real life situations where a person can attempt any action in his/her environment although most actions will yield nothing because they can't be performed in a given context. For example, an automobile driver should be able to turn into a one-way road in the wrong direction even though this action will probably lead to a negative reward. In a few cases, however, it may be the only option available to succeed (e.g., escaping a dangerous situation).

5.2.6 Environment Visibility and Accuracy of Actions

The portions of the environment necessary for the agent to make a decision will be visible to the agent at all times. In this dissertation, visibility is defined as "the greatest distance a person or an agent can see under normal conditions without the use of any instrumental assistants or the knowledge of distant events available to the agent at any given time" [unknown]. Invisibility is defined as "the distance beyond which a person or an agent can see clearly or lack of knowledge of an event not available to an agent irrespective of the distance" [unknown]. Ideally, an agent is expected to make decisions on its actions based on the visible part of the environment and projections on what it expects to exist in the invisible parts of the environment. This scenario is typical for all learning beings. In any mission, humans typically are not immediately aware of all elements of their environment. After they’ve interacted with their environment over a period of time and gained enough experience, they might think they are fully aware and make some projections on future states of their environment. Take, for instance, a person driving home from work. This person becomes familiar with this route and can project how long it would take to get to different landmarks on that route. Occasionally, the person might be wrong in his/her projections because of rain, accidents, and other environmental factors, but with considerable experience, he/she can make projections with some degree of accuracy.

Occasionally, knowledge about the invisible parts of the environment is presented to the agent through remote sensors or communications. For example, the agent driver hears on the radio about an accident and subsequent high traffic on a particular route it wanted to take. How would this issue be tackled in the enhancement process? Does the agent automatically consider the new knowledge as visible even though it cannot see it? Situations like this – provision of real-time knowledge would be handled by the simulator and would occur randomly during the course of the simulation. The enhancement process would handle this knowledge the way it handles all other knowledge – i.e. it can exploit it or keep on exploring. At the time the knowledge is provided, it is considered visible to the agent. It is left for the learning agent to decide whether to exploit its knowledge of the situation ahead or to explore it. This happens in real life when we hear radio announcements of traffic jams on a particular route, but as we approach that route, the traffic jam is cleared. The agent doesn’t know whether actions it performs are right or wrong because there isn't a "teacher" in its environment. It can only learn how to maximize its rewards and adjust its knowledge, based on actions that provide high rewards.

The visibility of the environment in a mission gradually increases as the mission evolves. The environment is a union of its visible and invisible parts. As the agent approaches its *sphere of visibility*[41] it takes actions and makes projections based on this. The question of what level of visibility [42] would the agent have arises. It is assumed the agent would initially have a 20/20 vision of events ahead and behind it.

$$E = v \bigcup_{t_k}^{\infty} iv \qquad 5.2.6.1$$

$$iv \to 0, v \to \infty$$

Where E is the environment, v is the part of the environment visible to the agent at time t_k and iv is the part of the environment invisible to the agent at the same time step.

Actions rules $\mathbf{AR_{MC}}$ and projection capabilities available to the agent at any time step t_k are directly proportional to the state in the visible portion of the environment $s_i v$

$$\mathbf{AR_{MC}} \propto \{s_1 v,\ s_2 v,\ \ldots.,\ s_i v\} \qquad 5.2.6.2$$

As the visibility of an agent increases, the agent is expected to perform better and make future decisions better.

5.2.7 States Vs Contexts

The definition of a context was presented in Chapter 2, as was that of a state. The CxBR technique is flexible enough to allow a context to span multiple states or allow a context

[41] Sphere of visibility relates to the radius of the environment visible to the agent.

[42] Experiments on various levels of visibility were conducted and the results presented in chapter 6.

to represent a single state. Models built with the CxBR technique allow a context to represent multiple states of the environment. For example, an *UrbanDriving Context* could be argued as having a single state – driving in an urban setting with a set of fixed actions. It could also be argued as having multiple states – driving in an urban setting with different variables to contend with, e.g. Pedestrians, slow traffic, etc.

In this investigation, a context is a grouping of similar states that dictate how an agent acts or reacts in a given situation. For example, in an *UrbanDriving* context between points A and B, between these points there can exist many states. These states share common characteristics of specified or defined behaviors expected from an agent when being controlled by the context, for example the maximum speed limit.

For contexts equal to a state:

$$C_i = s_1 v \qquad 5.2.7.1$$

For contexts equal to a group of states:

$$\mathrm{C}_i = \{ s_1 v \, , \, s_2 v \, , \, \ldots., \, s_i v \, \} \qquad 5.2.7.2$$

5.2.8 Generalization of Actions

Generalization of actions during the enhancement process is an integral part of the learning architecture. This is so because in large state problems, executing (exploring) every action in every state is impossible (the Bellman's curse of dimensionality[43]). The question on how to generalize a reinforcement agent arises. Previous works depended on function approximation – neural networks, etc. The issue of generalization does not arise in CxBR, because actions and reactions to anticipated pre-determined events in the agents' environment are presented prior to the start of the simulation.

[43] Bellman's curse of dimensionality was presented in Chapter 3

For the enhancement process, the agent generalizes its actions over some states and then observes the effect these have on its total rewards. A feed-forward neural network or variations of it, is used to generalize. Typically, when generalizing with a feed-forward neural network, input-output pairs exist and finding a 'weight' that can be applied to new inputs to produce the desired outputs is a goal of this network.

$$d = f(x) \qquad 5.2.8.1$$

where *d* is the dependent variable and *x* is the independent variable. Usually, the input – output pairs are denoted as: $(x_1,d_1), (x_2, d_2), \ldots.., (x_j, d_j)$.

$x_i(1 \leq i \leq j)$ are the inputs to the neural network and $d_i(1 \leq i \leq j)$ are the desired outputs. During the training phase, these outputs are known, but during the implementation (performance) phase these outputs are not known.

$$d = \sum_{k=0}^{k} w_{ik} x_k \qquad 5.2.8.2$$

where *d* is the desired output and w_i is the calculated weight and x_k is the input. The above equation is from Christodoulou & Georgiopoulos [57].

According to Sutton and Barto [63], most function approximation methods assume the training sets to be static over multiple training passes. However, in most reinforcement learning algorithms, it is desirable for learning to occur online, during the agents' interaction with the environment or with a simulated model of the environment. For this to occur, the function approximation method utilized must be able to learn efficiently from data acquired incrementally at various intervals. Also, function approximators used in RL should be able to handle non-static target functions (i.e. functions that change over time) [63].

Another issue usually addressed when generalizing in RL is the performance measure utilized in evaluating the chosen function approximation method. Generally, most supervised-learning techniques attempt to minimize the mean squared error over a distribution, *D*, of the inputs. In some RL techniques, the inputs to the function approximator are states and the target function is the true value function V^{π} , hence, from [63], the mean squared error (MSE) for an approximation V_t, using parameterθ_t, is

$$MSE(\theta_t) = \sum_{s \in S} D(s)(V^{\pi}(s) - V_t(s))^2 \qquad 5.2.8.3$$

where *D* is a distribution weighting the errors of different states. This distribution is important because it is usually not possible to reduce the error to zero at all states.

In the enhancement process, the inputs are the states within a context and the desired outputs are the values of these states based on the rewards received during the agents' interaction with its environment.

The generalization module and its algorithm are included in the architecture of the enhancement process; but for the purpose of this dissertation, they are not implemented because of the small state size for the chosen prototype. Look up tables are used and direct calculations of all state values is carried out.

5.2.9 Exploring and Exploiting Contextual Knowledge

The knowledge in contexts would be initially exploited to get a baseline of anticipated rewards in each state and anticipated values of states for the model. Thereafter, the agent would continually explore its environment by attempting various actions in each state in each context. The rate, at which the agent explores a given state in any context, would decrease exponentially with the number of visits to that states. As the simulation progresses and a state becomes visited many times, if an action stands out as being the most desirable in that state, this information would be exploited. The agent is said to perform better if the explored actions lead to state values that are better than the baseline.

5.2.10 Dynamic Goals

The issue of dynamic goals isn't addressed much in most human behavior models. CxBR does address it, however. An example of a typical human behavior that involves dynamic goals is a police officer rushing to attend to a distress call. On his way, he witnesses a separate life threatening accident. Does he then continue with his goal of attending to the distress call or is the accident severe enough for his immediate goals to change? CxBR addresses this issue by listing contexts for each situation and providing transition rules that would enable the activation of the listed contexts. In most cases, the SME omits or never envisions a situation where an agent's goal would change when in a given context and thus doesn't provide transition rules between the contexts.

This problem is addressed by having the agent determine what contexts to transition to in any given situation based on the knowledge it has learnt. In the example provided, the police officer would act appropriately based on what he has learnt on the situation, i.e., is the reward of attending to the distress call greater than that of attending to the accident victims? In other words is the value of proceeding to the distress call from the current state greater than the value of attending to the accident victims from the current state? This can be determined by trial and error in an experiential learning environment.

5.2.11 Conflicting Knowledge

It is expected that the SME would have some knowledge in a context that conflicts with either a sub-goal of the mission or the mission goal itself. Likewise, some knowledge in a context can also conflict with knowledge in another context. Knowledge in the enhancement process is non-monotonic. Knowledge can be retracted and added during the enhancement process. The enhancement process itself is based on these additions and removal of knowledge from the agent model. An example of a conflict in knowledge is a context that limits the maximum speed of an automobile to 30 mph and a sub-context within this context having a minimum speed of 40 mph. How will this conflicting information be used?

The defined rewards of the system identify this and a *conflict resolver* module addresses the situation. The conflict resolver module is based on some hierarchical principles in which the mission goal takes precedence over sub-goals and sub-goals take precedence over contextual information.

5.3 Flow of Events

The flow of events for the enhancement process is presented below. The details on achieving this are explained in the high level design section of this Chapter.

I. The Reinforced values *(Rx)* for each context, state and action tuple are all initialized to zero

II. The default context is activated and controls the agent initially in a simulation exercise.

III. An action in the active context is carried out taking the agent to a new state

IV. The value of that action in that state in the current context is calculated and the *Rx* values are updated based on the rewards for the mission goal - $Rx(c_i, s_i, a_i) = \mathrm{R}(c_i, s_i, a_i) + \gamma .\max[c_i s_j a]$

V. The sentinel rules are checked to see whether the current active context needs to be deactivated (and another activated).

   a. If a new context is called for, a new context is activated and control returns to step III

   b. The *context selector* module is activated. The context selector module searches through all defined contexts to see whether any match the current situation.

      i. If there is a match, this context is activated and a copy of the previously active context is made in the *context repository*. This copy is refined/enhanced by calling the *context modifier* module; the context modifier does this by adding the active context amongst the list of compatible contexts.

      ii. Control is returned to step III

   c. If none of the predefined context match the current situation the *context creator* module is called. This module creates a new context based on a predefined context template by adding the various parameters of the current situation to this template as obtained from the global fact base. Control is then returned to step III.

VI. If the mission goal is achieved, this marks the end of an episode[44]. A new episode is then started until the change in *Rx* values are negligible or zero, i.e. the values converge.

VII. Based on the *Rx* values, choose the action that produces the most reward for each state in a given context by choosing the max *Rx* value for each context-state-action combination. Compare the original predefined actions and attributes in a state in a context with the newly learned actions (actions calculated) and attributes for the same state in the same context

   a. If the newly-learnt action (calculated action) and attributes are different from the original action, create a copy of the context in the context repository and call the context modifier to refine / enhance the context with the newly learnt action and attributes for that state in the context.

   b. If the original predefined action and attributes are the same as the calculated action, do nothing

The flow of events detailed above can enhance actions and attributes within a context and context transitions as well as create new contexts based on a predefined context template.

[44] An episode is the beginning to end of an agents' interaction with its environment. It is essentially a run of the agents' activities from start to finish in the simulator. Many episodes need to be run in the simulator when training the agent.

5.4 Components of the Enhancement Process

For a CxBR model of human tactical behavior to be enhanced, the enhancement process utilizes new components in addition to the existing components of CxBR and RL. These components are described below. Figure 5.2 shows the architecture of the enhancement process.

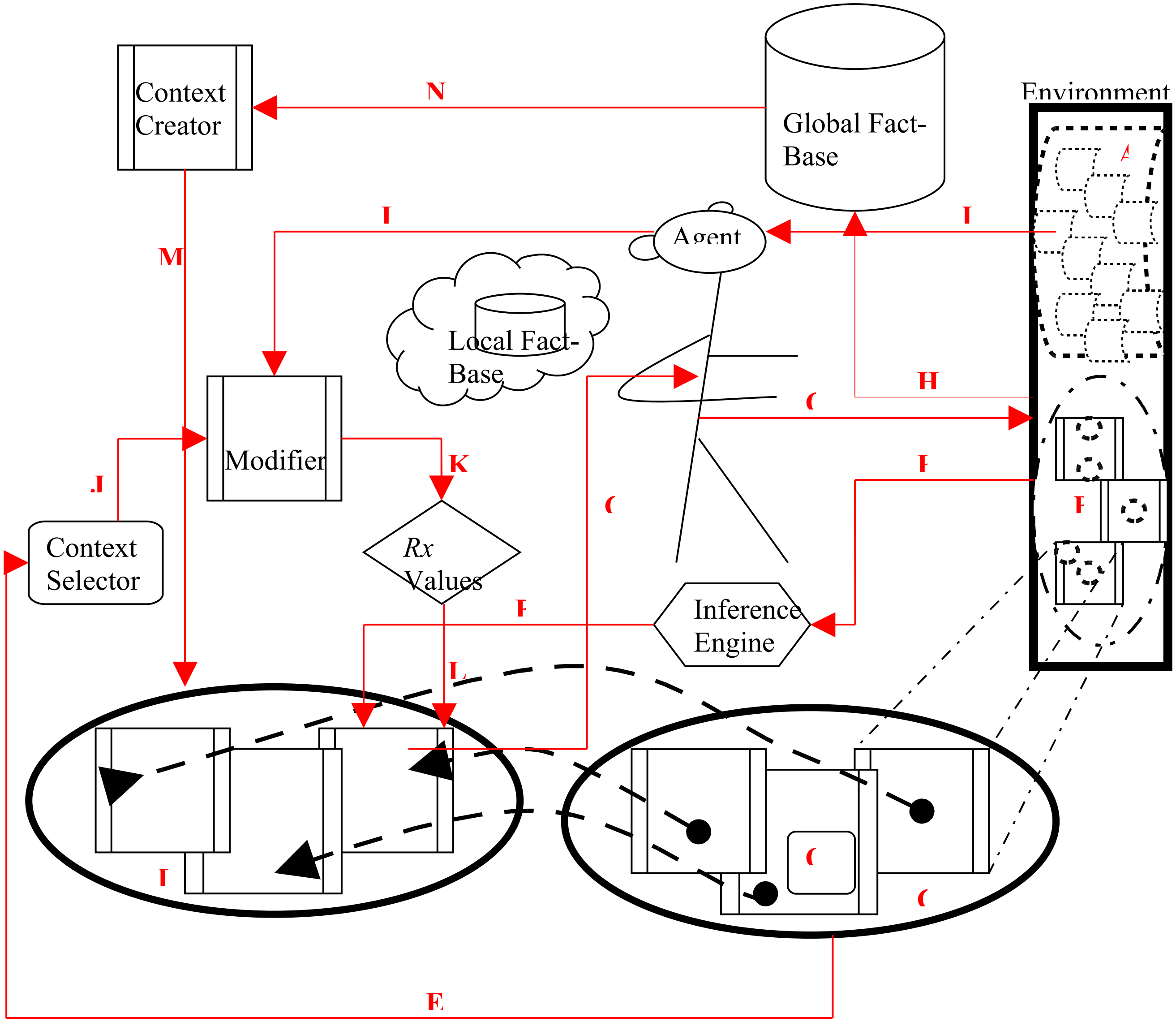


Figure 5.2 Enhancement Process Architecture

The labels shown in figure 5.2 are explained in the following subsections.

5.4.1 Action-Base (**A**)

The action-base is a component of the environment that contains all available actions in that environment. The actions in the action-base are not restricted to any state or contexts; they are available for the agent to explore in any state of the environment. Because the agent possesses the ability to execute any action in any state doesn't necessarily make the action appropriate for the state. Through experience, the agent would learn what actions are appropriate and those to avoid when in certain states and thus enable the enhancement process. The availability of an action-base relaxes the CxBR principle that only a few things can realistically occur in any context. Although it relaxes this principle for the purpose of learning, the principle still holds true in reality. Therefore, after the model is enhanced, the contexts eventually restrict the actions of the agent in any given state based on what is learnt. The reasoning behind relaxing the principle is for the agent to be able to explore actions not thought of by the SME when in a given context. The exploration only occurs when the agent is learning or if it perceives a change in its environment. A direct negative impact this would have on the overall learning process is an increase in the time it takes to train the agent because of an increased number of actions the agent will need to explore / perform in every explored state of its environment.

The action-base has an input of the state of the environment and has an output of an action. As the agent learns the best action in a state, this information is sent to the modifier module through ***I*** from fig. 5.2, likewise to the global fact-base through ***H***.

5.4.2 Environmental States / Contexts (***B***)

States are components of the environment. An environment consists of many states. A context is a group of states where similar actions can be performed. Typically when an agent performs an action in the environment from a state, the action leads the agent to a new state. The new state can be in the same context as the previous state or it can be in a different context. Based on the state of the environment, the agent performs an action or a group of actions and gets to a new state or remains in the same state. The process continues until the mission goal is accomplished. The current state of the environment is processed and analyzed by the inference engine through ***F***, the result enables the picking of a context.

5.4.3 Context Library (**C**)

The context library is a collection of all predefined contexts for the model. The definition and descriptions of contexts have been presented in Chapter 1. Only one context in the context library can be active at any given time. The contexts in the context library take the states of the environment as inputs, thus the only allowed input to the library is a 'state' signal. The outputs of a context are the actions the agent can execute from any given state, thus an output of the context library is the prescribed action for the agent. Another output of the context library is the active context with all its attributes. A copy of an active context is automatically copied to the context repository in preparation for its refinement. The link ***L*** in figure 5.2 shows the active context being copied over to the context repository.

5.4.4 Context Repository (**D**)

This component is also known as the enhanced / refined context repository. It is labeled as D in Figure 5.2. A formal description along with the mathematical formalization was presented in section 5.2.4. Copies of all modified contexts are stored in the context repository. The functions of this repository are to provide an efficient backup mechanism for the learning process. The repository enables the addition and retraction of new knowledge by keeping track of all changes made to a context and at what "state" in the world the changes occurred.

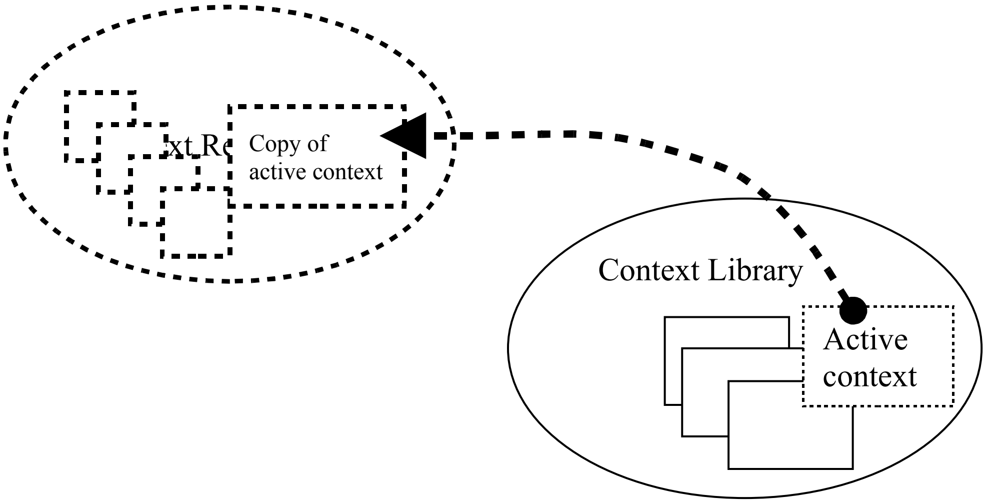


Figure 5.3 Context Repository

This repository will take contexts as inputs. The outputs are the actions in the refined contexts. A call to the context repository module immediately creates a copy of the active context. Control is automatically transferred from the active context to the copy created in the repository. Actions are explored in this context and the values of these actions in the states of the context are stored in the *Rx* table.

$$\boldsymbol{CCi} \leftarrow {}^{A}\boldsymbol{Ci} \qquad 5.4.4.1$$

$$CCi \Leftarrow \{(s_i, a_i), (s_i, a_j), ...(s_n, a_n) \mid (r_i, r_j, ..., r_n)\} \qquad 5.4.4.2$$

The algorithm for the context repository module is as follows.

```
Upon activation of a new context
Create a copy of the active context in the context repository
Automatically transfer control of the agent to the newly created copy of the active context
For the current state $\mathbf{s_i}$, do until context $C_i$ becomes inactive
    attempt action $\mathbf{a_i}$ and note the resulting reward obtained
    update the Rx values according to the equation
            $Rx(c_i, s_i, a_i) = \mathrm{R}(c_i, s_i, a_i) + \gamma.\max[c_i s_j a]$
    explore or exploit an action in the new resulting state
```

5.4.5 Context Selector

When called, the context selector module chooses the appropriate predefined context for any given situation. Typically, in a CxBR model, the list of contexts that can be transitioned from any given context is predefined within the context. This list is based on expert knowledge about the given situation and the characteristics that would necessitate a transition from the active context. Most times, this list is mostly correct, but occasionally the list is incomplete or wrong. For example an *UrbanDriving* context that has a list of compatible contexts that excludes a *Freeway* context or *Ramp* context is incomplete. There are no mechanisms to prevent having a wrong or incomplete list of compatible contexts. Therefore, a CxBR simulation ends (fails) when faced with this situation because there is no valid defined context to transition to, based on the list of compatible contexts. Likewise, when faced in situations of incomplete knowledge or wrong knowledge, most human behavior modeling systems fail or act abnormally. The enhancement process attempts to eliminate this by first searching through all predefined contexts to see whether the attributes of any context match the current situation. If the attributes of a given context match the current situation, the context is selected as the new active context and the previously active context is sent to the modifier as represented by ***J*** in figure 5.2. The modifier then modifies and enhances the context by adding the newly active context amongst the list of compatible context.

In some cases, the attributes of the predefined context do not explicitly match the current situation. The contexts that match the current situation are then selected and sent to the modifier where some of their attributes are modified and tested to see whether they match the current situation and still maintain their previous attributes. A comparison of the contexts modified to address the current situation is carried out as depicted in figure 5.2 by ***K***. Changes to the context that provides the highest *Rx* values are kept. This context is sent to the enhanced / refined context repository.

This module takes as input all predefined and enhanced contexts and the output is one or more contexts that appropriately address the current situation. The way the context selector module works is almost akin to the competing context concept conceived by Saeki & Gonzalez [28]. The major difference is in the way contexts are selected. While the competing context concept eventually chooses a context at random during the hyper simulation, the context selector module makes it choice based on the calculated *Rx* value.

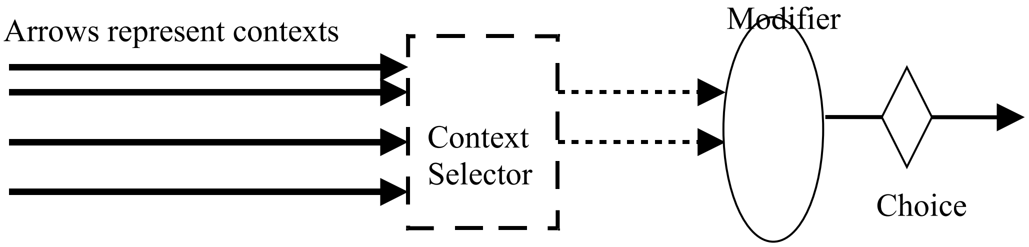


Figure 5.4 Context Selection Process

5.4.6 Context Modifier

The context modifier module enhances predefined contexts by modifying the attribute values available to the agent in a given state in the context, for example, actions. After a context or group of contexts have been selected by the context selector module to match the current situation, the contexts are passed through the modifier module where various actions defined within them are attempted as well as actions defined in the action-base. After these actions are executed, the action that produces the maximum Rx value for that state is chosen as the most appropriate and the context is modified to reflect this. In cases where only one context is selected by the context selector to appropriately address the current situation, the previously active context is sent to the modifier and the list of compatible contexts modified to reflect the newly active context. The modification of the previously active context also occurs in cases where many contexts are chosen by the context selector and an appropriate context is chosen based on the highest Rx value.

For a context to be modified, the predefined actions as well as actions from the action-base are performed at random and the rewards from these actions are tabulated in *local memory base* available only to the context modifier module. These rewards are back propagated from the goal of the agent. After these actions are executed and their *Rx* values known, a choice is made on the action and context that best address the current situation relative to the goal of the agent.

From figure 5.2, the modifier takes as inputs the contexts through ***J*** and the actions from the action-base through ***I***. the output is either a list of modified contexts with their Rx values which is depicted by ***K***.

Formally, the copy of a selected context going through the modifier is as follows:

For a given state in context *i*, different actions are attempted.

$$CCi \Leftarrow \{(s_i, a_i), (s_i, a_j), ...(s_i, a_n) \mid (r_i, r_j, ..., r_n)\}$$

5.4.6.1

The state/action combination that produces the maximum reward is chosen

$$(s_i, a_i) \Leftrightarrow \max(r_i) \qquad 5.4.6.2$$

The previously active context is then modified to highlight the new context as a compatible context.

The flow of events for the modification of a context is as follows:

- *Upon a change in situation*
- *Search through the list of compatible context for a context that best addresses the current situation*
- *If no context addresses the current situation*
    - *Search through the list of all contexts (predefined, enhanced and newly created) to determine the context that most nearly matches current situation. (Determination of a context that nearly matches the current situation is done by directly comparing the attributes of*

the current situation as retrieved from the global fact base and those of all contexts as defined by the action-rules, sentinel-rules, etc. that are needed for the context to be activated).

- *A score is provided to each context as it relates to the current situation and the context with the highest score is selected for modification. The way the score is obtained is by calculating the total number of attributes that match the current situation as a function of the total number of attributes of the current situation. For e.g., if 10 attributes are listed by the global fact base for the current situation, a context that satisfies 8 of those attributes is said to have a score of 80%.*
- *If more than one context address the current situation*
  - *Rank the contexts by their scores*
  - *Modify each ranked context by performing the predefined actions and actions defined in the action-base randomly*
  - *Note the Rx value as calculated from*

$$Rx(c_i, s_i, a_i) = \mathrm{R}(c_i, s_i, a_i) + \gamma\,.\max[c_i s_j a] \qquad 5.4.6.3$$

- *The action that produces the highest Rx value for the given state – action combination is chosen*
- *The context is modified to reflect this*
- *The previously active context is also modified to reflect the addition of the newly modified context among the list compatible next contexts*

5.4.7 Context Creator

The context creator module creates a new context on demand. When the enhancement of a context yields suboptimal[45] values, the enhanced context is reinstated back to its previous form from the context repository and a new context is created. The new context is created from a predefined context template. This template contains sections for action-rules, sentinel-rules, compatible contexts, and others. It primarily contains sections for all the definitions of a typical context.

As soon as a decision is made to abandon all pending changes to an existing context (enhancement) based on the *Rx* values being received for the enhanced context or the system has determined that the number of attributes of existing context that match the current situation is low (less than 40%) a call is made to the context creator module.

Upon calling the context creator module, a copy of the context template is made. The attributes of the current situation are filtered from the global fact base and these attributes are inserted to the context template copy. From the attributes, the title (name) of the newly created context is generated. The newly created context is then sent to the context repository where different actions from the action-base are attempted and their Rx values noted. The actions and transition rules that produce the maximum Rx values for each state in the context are noted. The constraints of the context are part of the attributes.

The algorithm for the creation of a new context is as follows:

Upon a change in situation

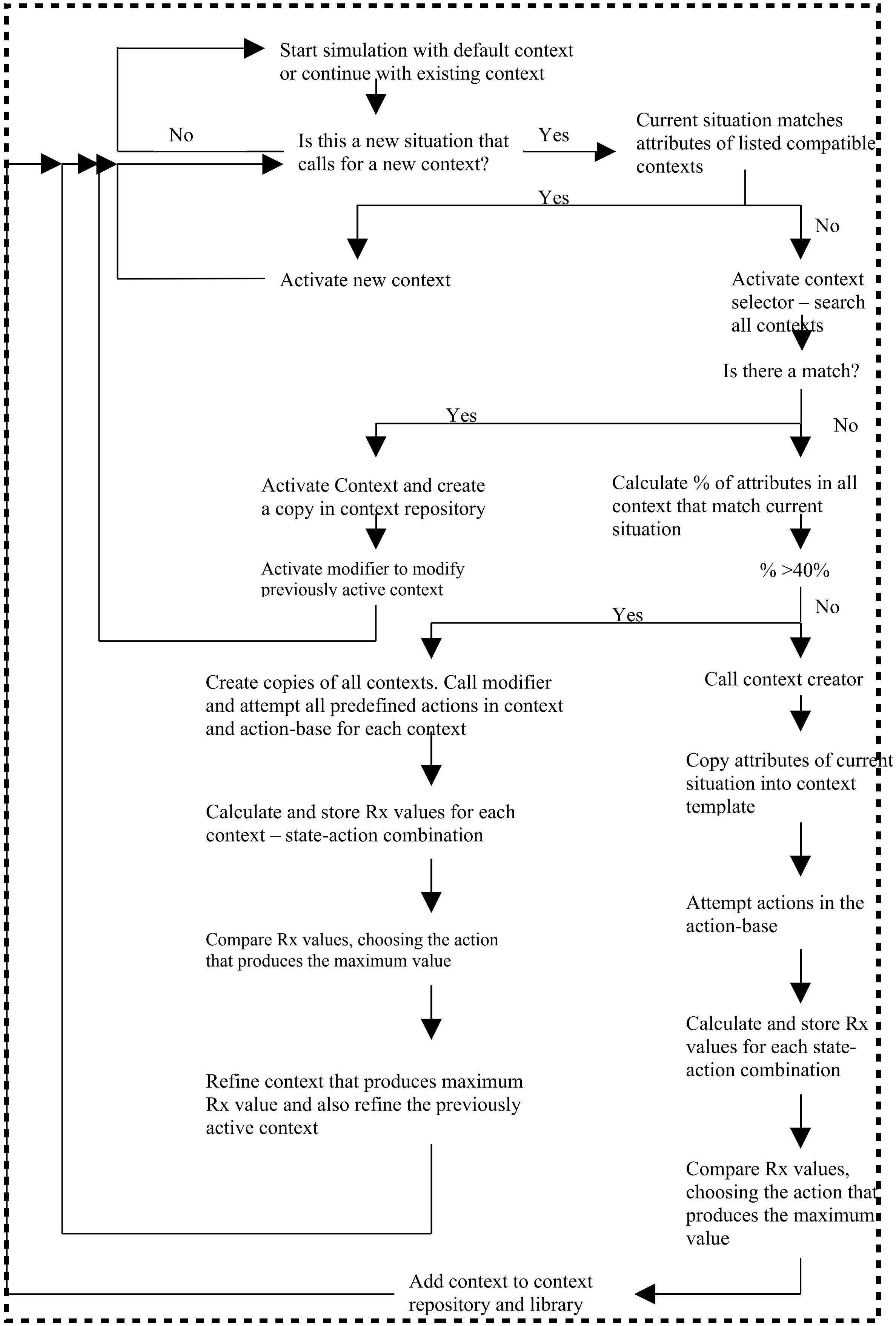
[45] Suboptimal values of a refined context are *Rx* values that are extremely low for the given state-action and the *Rx* values of previously calculated context-state-actions are reduced. For example, a context-state-action that previously produced the maximum *Rx* value and thus maximum rewards, if after being refined this same context-state-action produces a lower *Rx* value, the Refined context is said to be suboptimal.

```
Upon searching through all existing contexts and finding no matching context or upon finding a context with a low score[46] (<40%)
Call the context creator module.
While the simulation is paused temporarily
    Make a copy of the context template and place it in the context repository
    Get all the attributes of the current situation from the global fact base
    Filter these attributes according to various parameters, for e.g., location, constraints, type of road, etc. (based on the system being modeled)
Continue simulation by making calls to the actions in the action-base
As each action is executed, a note is made on the Rx value obtained for that action-state combination.
The action that produces the highest Rx value for each state-action combination is chosen as the appropriate action
Update the context with the newly gathered information about its action-rules and transition rules
```

[46] A score is defined as the total number of attributes of a context that match the current situation divided by the total number of attributes of the current situation. 40 % was chosen because it was intuitively determined that more time will be spent modifying a context with more than 40% matched attributes than creating a new context.

5.5 Enhancement Process Flow Chart

Figure 5.5 below presents the enhancement process flow chart.


Start simulation with default context or continue with existing context
No
Is this a new situation that calls for a new context?
Yes
Current situation matches attributes of listed compatible contexts
Yes
No
Activate new context
Activate context selector – search all contexts
Is there a match?
Yes
No
Activate Context and create a copy in context repository
Calculate % of attributes in all context that match current situation
Activate modifier to modify previously active context
% >40%
Yes
No
Create copies of all contexts. Call modifier and attempt all predefined actions in context and action-base for each context
Call context creator
Copy attributes of current situation into context template
Calculate and store Rx values for each context – state-action combination
Attempt actions in the action-base
Compare Rx values, choosing the action that produces the maximum value
Calculate and store Rx values for each state-action combination
Refine context that produces maximum Rx value and also refine the previously active context
Compare Rx values, choosing the action that produces the maximum value
Add context to context repository and library

5.6 Formal Representation

In this section, formal formalizations are provided for the enhancement process as well as the context creation process.

For the enhancement process, an existing context is refined to appropriately address the current situation.

For any given context C_i, there exists a set of predefined actions for the set of states in the context.

$$C_i = \{(s_i, a_i), (s_i, a_j), (s_i, a_n);(s_j, a_i), (s_j, a_j), (s_j, a_n);\ldots\ldots\ldots\} \qquad 5.6.1$$

There also exists a set of predefined actions in the action-base which is available to all states in all contexts.

$$a\text{-}b=\{a_1,a_2,a_3,a_4,\ldots\ldots.a_n\} \qquad 5.6.2$$

For the given context C_i and state s_j, a comparison of Rx values is carried out.

Rx (context, state, action$_1$) = Reward (context, state, action$_1$) + γ .Max [Rx (same context, next state, all actions]

$$R_X(C_iS_ja_1) = R(C_iS_ja_1) + \gamma .Max[R_X(C_iS_{j+1}a)] \qquad 5.6.3$$

$$R_X(C_iS_ja_2) = R(C_iS_ja_2) + \gamma .Max[R_X(C_iS_{j+1}a)] \qquad 5.6.4$$

$$R_X(C_iS_ja_3) = R(C_iS_ja_3) + \gamma .Max[R_X(C_iS_{j+1}a)] \qquad 5.6.5$$

.

.

.

$$R_X(C_iS_ja_n) = R(C_iS_ja_n) + \gamma .Max[R_X(C_iS_{j+1}a)] \qquad 5.6.6$$

From equations 4.6.3 to 4.6.6, the context is constant, except in cases where the next state falls within a new context. Taking C_i out of the equations, you have:

$$R_X(S_j a_i) = R(S_j a_i) + \gamma . Max[R_X(S_{j+1} a)] \qquad 5.6.7$$

The appropriate action for each state in the context is thus:

$$\mathbf{Max}\ \{\boldsymbol{R_x}\} \qquad 5.6.8$$

The formalization of the context creation process follows the same principle as that of the enhancement process with only one exception, i.e. the only actions attempted are the actions defined in the action-base.

5.7 High Level Design of Architecture

There are five sub-systems that interact together to perform the model refinement (enhancement). The sub-systems have been described in previous sections and include the context creator, context modifier, action-base, context selector and context repository. Figure 5.6 shows the inputs and outputs from these sub-systems. At the center of all activities is the CxBR core, i.e., all existing components of CxBR architecture as described in Chapter 2 e.g. the global fact base, the inference engine, amongst others. The CxBR core, communicates directly with the context repository. It also communicates directly with the context creation module by sending filtered [47]attributes about a situation to it. The context creator, context repository and action-base are the next layer and communications between these sub-systems are as shown in figure 5.6. The context selector and context modifier sub-systems communicate with other sub-systems as shown in the diagram.

[47] Filtered attributes refer to the attributes of a situation that correspond with the attributes required in the context creation template

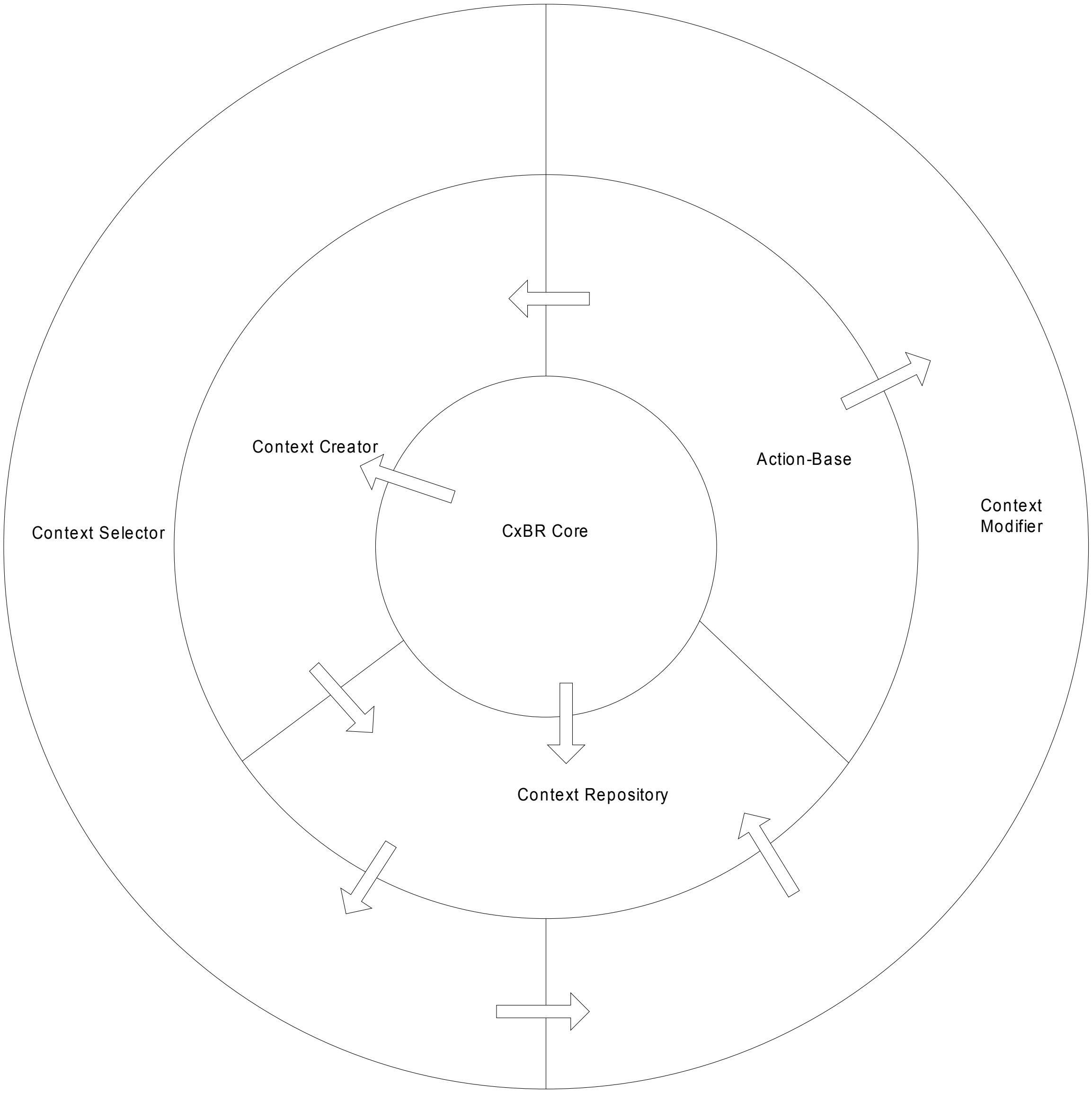


Figure 5.6 High-level Design Showing Inputs and Outputs between Sub-systems

5.8 Preview of Prototype

The prototype used for this research is in the automobile driving arena. An agent driver is expected to behave optimally on a driving mission when faced with various scenarios. The choice of an automobile driver prototype is supported by the existence of results from some automobile driver prototypes using the CxBR architecture. An automobile agent driver is given a mission goal of going from home to work with some constraints like arriving on time by choosing the fastest route, choosing the shortest route and many others. There exist different routes from home to work, and each route comes with its unique features, e.g. raining, potholes, intersections and many others. The prototype consists of a hand-built CxBR model with purposely incomplete knowledge. The prototype should enhance the model to enable the agent achieve its mission goal.

5.9 Chapter Summary

In this chapter, the conceptual approach to resolving the problem defined in Chapter 3 was presented. The definition of new modules that would help in achieving this is carried out as well as a full description of these modules in the overall flow structure of the enhancement process. An algorithm and the formalization of the enhancement process were presented. The high level design of the enhancement process architecture was presented as well as a sneak preview of the prototype.

CHAPTER 6: A PROTOTYPE IMPLEMENTATION OF THE MODEL ENHANCEMENT PROCESS

6.1 Introduction

Two prototypes implementing the technique for breaking the limitations on SME knowledge in HBR systems and thus improving the performance of the agent are described in this chapter. The prototypes are built and tested in the automobile driving domain and in the submarine warfare domain. The automobile driving domain consists of the agent driver; the environment which is composed of the different routes; the context base; and the enabling functions for the simulation. In this dissertation, some pertinent facts and attributes that are typical in automobile driving and submarine warfare domains are neglected as they add little value to prove or disprove the hypothesis set forth in Chapter 3. These attributes include the pressure on the acceleration pedal, wind velocity, gravitational forces and others for the automobile driving domain. For the submarine warfare domain, the size of the submarine, the functioning of the periscope, flood-tubes, and others are also neglected. All aspects of a typical automobile driving domain and submarine warfare domain that are not explicitly mentioned and used in the design and implementation of this prototype are neglected.

A model of an agent driver and submarine is built a priori from the knowledge acquired from a subject matter expert (SME) through various methods that include observation, question and answer sessions, and others. How the pre-existing model was built is likewise irrelevant to this research.

6.2 Prototype Descriptions

The underlying techniques for building the automobile driver prototype and submarine warfare prototype are the same. The only differences are the context definitions and the environment in which the agent operates. I will provide a description of the automobile driver prototype in subsequent sections of this chapter and only discuss the submarine warfare prototype in the section describing the environment and building the contexts.

The prototype used to implement and evaluate the model enhancement process consists of an agent driver, the context-bases, context-base functions, functions that enable the enhancement process and an environment in which the agent operates (the world). The prototype was developed using Oracles' PLSQL programming language and an Oracle database. The prototype consists of the various modules described in section 5.4 of this dissertation. Database tables are created in an Oracle database. These tables are used to store various data on the enhancement process as well as the entire simulation. Among the information and data stored are the log of the entire simulation, context definitions, context actions, context attributes, global facts and rewards.

The original CxBR Framework developed by Norlander [124] is not utilized in the design and implementation of the prototype because the Framework does not support the learning mechanism needed by the enhancement algorithm to enhance CxBR models. In the prototype, contexts are defined and created in a context table. More on this in the design section of this chapter.

The first step in the operation of the prototype is to create the underlying database structures that store information on the agent, the context-bases, the world and the entire simulation. This is known as the back-end of the application. After the creation of the underlying database structures, the various modules that control the workings of the agent are implemented.

The prototype operates in two phases. The first phase is the training of the agent, (the learning of contextual and environmental attributes and actions; the learning of new contexts) while the second phase is the execution of the enhanced agent. The agent recognizes whether it has been previously trained by searching through the rewards table and global fact base table. If rewards exist and the entries in the global fact base point to a *successful completion*[48] of learning, it is assumed that training has occurred. The prototype then provides the option of either retraining the agent or using its current knowledge. If a context definition or world definition has been changed since the last time the agent was trained, for example, if the maximum speed defined for a road segment has been increased, or the allowed depth[49] of the submarine has been increased, then the prototype automatically retrains the agent.

The training of the agent for the automobile driving domain consists of the agent learning the optimum maximum speed defined for the different contexts it encounters during its interaction with the world; and consists of learning the appropriate depth for the submarine warfare domain.

6.3 Prototype Requirement Specifications

The function of the model enhancement prototype is two-fold. 1) To provide a test bed for evaluating the context-based human behavior model enhancement technique. 2) To show that the enhancement technique enhances an agents behavior by breaking the

[48] Successful completion of learning is when the agent has successfully enhanced the contexts in the context bases or learnt a new context. The function that identifies this is shown in a subsequent section in this chapter

[49] The depth of the ocean which the submarine must not go beyond, to be discussed later.

barrier of SME knowledge limitations and the limitations inherent in most knowledge acquisition techniques currently in use. The prototype utilizes a pre-built CxBR agent driver model. This model is executed in a simulator and subjected to the enhancement technique. After the model is enhanced, the agent is expected to perform better by achieving its goals within the defined constraints. Furthermore, this enhanced model highlights any inconsistencies in the knowledge acquired from an SME.

To efficiently design, implement and understand the prototype, references must be made to the preceding chapters. A few standards to which the design and implementation of the prototype must adhere include:

a. The coding standards for this prototype include:

- Each context will be represented by a record (row) in the context table.
- Sub-contexts can be reused by all major contexts
- The design of a context shall exclude the definition of more than one situation. The definition of a situation is… "One or more states with similar properties closely located to one another" …. "A set of similar states or circumstances"[1]. By limiting the design of a context to one situation, it prevents ambiguities that could arise from having multiple situations defined in a given context, for example having a traffic light situation defined in the same context as city driving or freeway driving situations.
- The use of goto statements is not allowed
- The names of all modules / functions / classes should reflect the activities of the module or function or class
- Code should be commented adequately for readability

- Context depth level shall be restricted to 2 for the purpose of this prototype, i.e., you can have a context and a sub-context. No definitions of sub-sub-contexts, or below shall exist.

6.3.1 Assumptions

The following assumptions were made with respect to this prototype:

- Other agent drivers do not exist on the road. Hence, there are no contexts defined for "following other cars", "overtaking other cars" and such other contexts that interact with other cars.
- The agent knows of an event in the global fact base as soon as it occurs, hence no delay in the transfer of knowledge
- Agent has no knowledge of the environment or segments of it, meaning the agent cannot see farther than the information provided in the global fact base on the environment. The environment is predefined and knowledge about agents' current location is passed through the global fact base (GFB) to the agent.
- The dynamics of the environment and the car are neglected. These include frictional forces, wind force, driving at night vs. day, gravitational forces, acceleration, car design, car size, angular velocities as well as angular representations of routes (hilly routes, valleys) and other such issues.
- The width and elevation of the road isn't taken into consideration.

6.3.2 Stakeholders of the Model Enhancement Methodology

The model enhancement technique has several potential and required stakeholders.

a. Required stakeholders include:

- Human Behavior Representation Research Community – HBR researchers need a tool that can validate and enhance models designed from knowledge acquired from SMEs. This tool can be used in the enhancement as well as the validation of SME knowledge.
- Model Engineer - The model engineer / knowledge engineer can utilize this prototype to learn more on a subject being modeled and focus their questions to SME's on the enhanced aspects of the model.

b. Potential stakeholders include:

- SME – the SME can use this methodology / prototype to expand his/ her knowledge on a subject. After a model of the knowledge provided by the SME has been built, the SME can use the enhancement technique to learn about information they provided to determine which is wrong and safeguard against this in the future.

6.3.3 Sequence of Events

The sequence of events during the agents' simulation is described in Table 6.1 below. The prototype operates a two-phase process. The first phase is when the training of the agent occurs and the second phase is the execution of the enhanced agent. Most events that occur during both phases of the simulation trigger an external stimulus as well as some manipulation of the internal data and the state of some parts of the simulation. Table 6.1 analyzes what these events are, the external stimuli that triggers the events, the external responses and internal data changes and state of agent during and after the event.

Table 6.1 Event Table for Automobile Driving Prototype

Event Name	External Stimuli	External Responses	Internal Data and State
Start Simulation	None	None	1. A determination is made by the agent if training is complete or if training is needed. 2. Variables are initialized, simulation time is initialized, log table is truncated (cleared) and rewards table is truncated.
Parking Lot Driving – Default Context	None	None	1. The default context is activated and the agents' behavior is controlled by it (the parking lot context). This context sets the max speed and actions available to the agent. When the situation changes, it is no longer active.
City Driving	None	None	1. Agents' behavior is controlled by context – maximum speed and action available to the agent.
Dirt Driving	None	None	1. Agent's behavior is controlled by context – maximum speed and action available to the agent
Freeway Driving	None	None	1. Agents' behavior is controlled by context – maximum speed and action available to the agent.
Ramp Driving	None	None	1. Agents' behavior is controlled by context – maximum speed and action available to the agent.
Traffic Light Driving	None	Light is red, yellow or green	1. Agent responds to the color of light by stopping, slowing down or continuing at current speed. This is a sub-context.
Intersection Driving	None	Stop Sign Present	1. Agent responds by slowing down and subsequently stopping at the intersection. This is a sub-context.
End Simulation	None	None	1. Simulation cycle ends. In training phase, the agent receives a reward. Reward received is used to train agent on learning maximum speed and appropriate actions in the context being enhanced. In execution phase, agent either achieves its mission goal or does not achieve it.
Modify Context	None	None	1. Attributes of an existing context are modified after the agent has been trained. This enhances the context.
Create New Context	None	None	1. Attributes of current situation are copied and a new context is created.

6.3.4 Use Case Diagram

Figure 6.1 shows the use case diagram of the prototype. There is one actor in the environment, the agent. The use case diagram shows how the agent interacts with the CxBR system. The actions performed by the agent as well as the appropriate responses received by events in the environment.

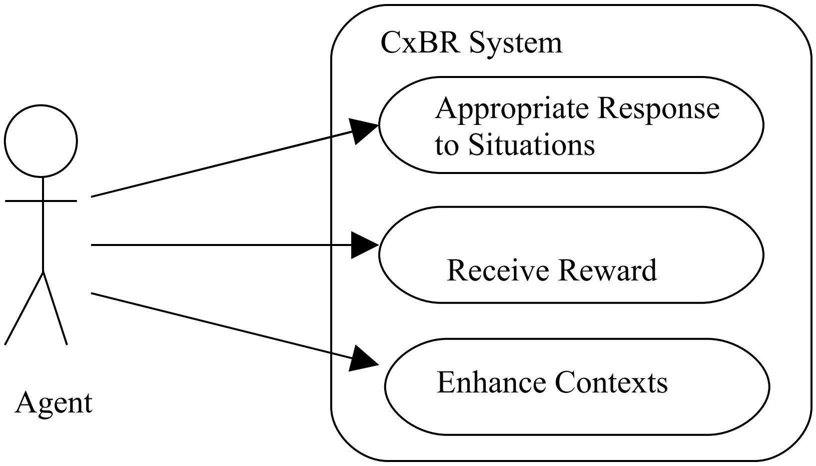


Figure 6.1 Use Case Diagram

6.3.4.1 Use Case Descriptions

a. Appropriate Response to Situations (Perform Action)

- Description: Pre-defined contexts are used to control the agents' actions / responses to situations in the agents' environment. Contexts contain information on the agents' actions and responses to various events and situations in the agent's environment, for example, increasing or decreasing its current speed, knowing the pre-defined speed limits in a context and many more. The knowledge included in this information allows the context to control the agent during the simulation.

  - Exceptions: These arise when there are no predefined contexts to address current situation. The context creation module is activated when this exception arises. The function that does this will be described later in this chapter.

b. Receive Rewards

  - Description: As the agent encounters new states during its interaction with the environment, it performs actions in these states. Feedback is received from the environment in form of rewards. These rewards describe how good or bad the action performed is, but doesn't say whether the action is right or wrong. Rewards are given via a predefined reward function that places rewards on the agent achieving its mission goal. The reward function is described in a later subsection in this chapter.
  - Exceptions: none.

c. Enhance Contexts

  - Description: Based on the rewards received, the agent's actions and responses to situations are refined and enhanced. This refinement leads to the enhancement of the agents' overall behavior.
  - Exceptions: Attributes and actions in contexts are not refined if deemed the best for the state.

6.3.5 Specific Requirements

This section describes the specific requirements for this prototype.

6.3.5.1 FUNCTIONAL REQUIREMENTS
1. The system shall allow users to define the agents' environment according to the provided guidelines and format
2. The system shall identify if training is complete and thus use existing learnt knowledge or if further training is needed.
3. The system shall enhance an agents' behavior by modifying actions and attributes in a context, based on a predefined mission goal
4. The system shall create contexts that represent agents' behaviors in unknown situations. These contexts shall be created from information received from the global fact base as well as rewards received from the environment.
Evaluation Method: Test Plan

6.3.5.2 INTERFACE REQUIREMENTS
1. The system shall not interface with any other application
2. There shall be a front end client application where the main function of the application will be initiated.
Evaluation Method: N/A

6.3.5.3 PHYSICAL ENVIRONMENT REQUIREMENTS
1. The system shall operate on Microsoft Windows© 95/98/2000/XP or Linux/Unix operating systems or other systems that have PLSQL programming language with Oracle database.

Evaluation Method: Test Plan

6.3.5.4 USERS AND HUMAN FACTORS REQUIREMENTS
1. The system shall support modelers, SME's and knowledge engineers.
Evaluation Method: Test Plan

6.3.5.5 DATA REQUIREMENTS
1. The system shall take as input the entry to execute the main function of the application.
2. The data for creating the original model shall be obtained through any method from a SME. The system is required to then enhance this model as represented in contexts.
Evaluation Method: Test Plan

6.3.5.6 RESOURCE REQUIREMENTS
1. The space required is dependent on the model being designed, likewise the memory requirements.
Evaluation Method: Test Plan

6.3.5.7 SECURITY REQUIREMENTS
1. The system shall not require any security settings at this time
Evaluation Method: N/A

6.3.5.8 QUALITY ASSURANCE REQUIREMENTS
1. The agents behavior in the simulation shall be controlled by a context designed for that particular situation in the simulation.
2. The system shall use a "reasonable" amount of system memory during normal operation. Memory and CPU utilized are directly proportional to the size of model being enhanced.
3. The system reliability shall be 100% when operating under normal conditions
4. The time used to learn & enhance a model shall be reasonable and acceptable
Evaluation Method: Test Plan

6.3.5.9 PERFORMANCE REQUIREMENT
1. The agent shall achieve realistic[50] mission goals
Evaluation Method: Test Plan

6.4 Prototype Design (Experimental Test-bed Design)

A description of the initial, hand-built model of the agent is presented. The prototype consists of many parts and these parts all work cohesively to provide the learning required to enhance the model. The designs of the different parts of the prototype are described below.

[50] Realistic mission goals are those that are achievable within the context of known scientific researches (as of today). An example of an unrealistic mission goal is having an agent arrive at a destination 100 miles away in 1min while driving a car having a maximum speed of 60 m/h!

6.4.1 The Environment

The automobile driving environment consists of three unique routes composed of different road segments. These routes are used in the execution phase[51] of the experiments. Among the road segments in the routes are a parking lot - which acts as the default starting point for some routes; a city road segment, a freeway road segment, a dirt road segment, a ramp road segment, a traffic light, and an intersection segment as shown in Tables 6.2, 6.3, 6.4, figures 6.2, 6.3 and 6.4.

Table 6.2 Route A

ROUTE_ID	1	1	1	1	1
ROAD_ID	1	2	3	4	5
ROAD_NAME	PARKING_LOT	CITY	FREEWAY	CITY	PARKING_LOT
DESCRIPTION	PARKING LOT Driving	CITY driving	FREEWAY driving	CITY driving	PARKING_LOT driving
ROAD_LENGTH	0.2	2	6	3	0.15
ANGLE	80	15	35	18	60
ROAD_TYPE	PARKING_LOT	CITY	FREEWAY	CITY	PARKING_LOT
TRAFFIC	0	1	0	0	0
INTERSECTION	1	0	0	0	0
MAXSPEED	15	50	75	50	15

[51] There are two phases of the experiments, the training and execution phases. More on the training phase in chapter 7.

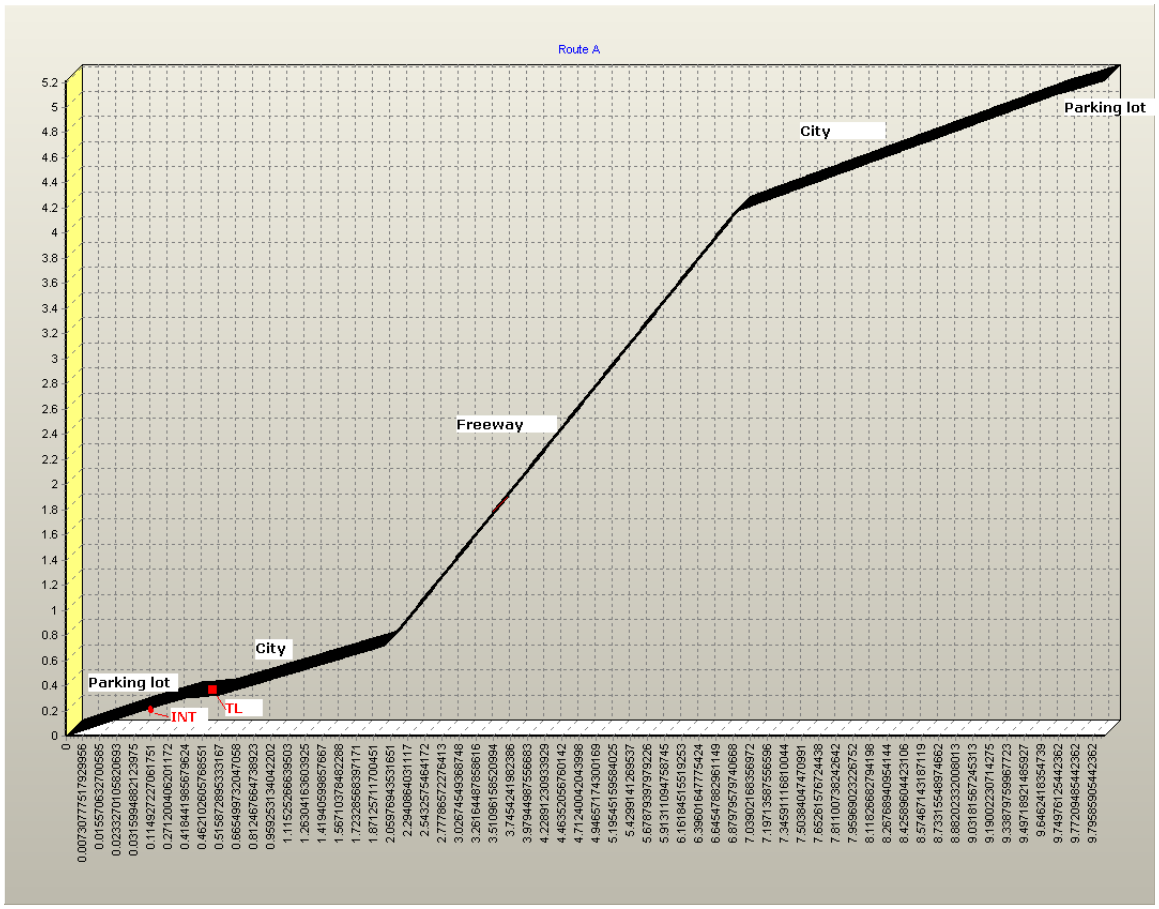


Figure 6.2 Pictorial Representation of Route A

Table 6.3 Route B

ROUTE_ID	2	2	2	2
ROAD_ID	1	2	3	4
ROAD_NAME	CITY	CITY	FREEWAY	RAMP
DESCRIPTION	CITY driving	CITY driving	FREEWAY driving	RAMP driving
ROAD_LENGTH	2.5	3.5	5	0.5
ANGLE	30	65	15	45
ROAD_TYPE	CITY	CITY	FREEWAY	RAMP
TRAFFIC	1	1	0	0
INTERSECTION	1	0	0	1
MAXSPEED	50	50	75	35

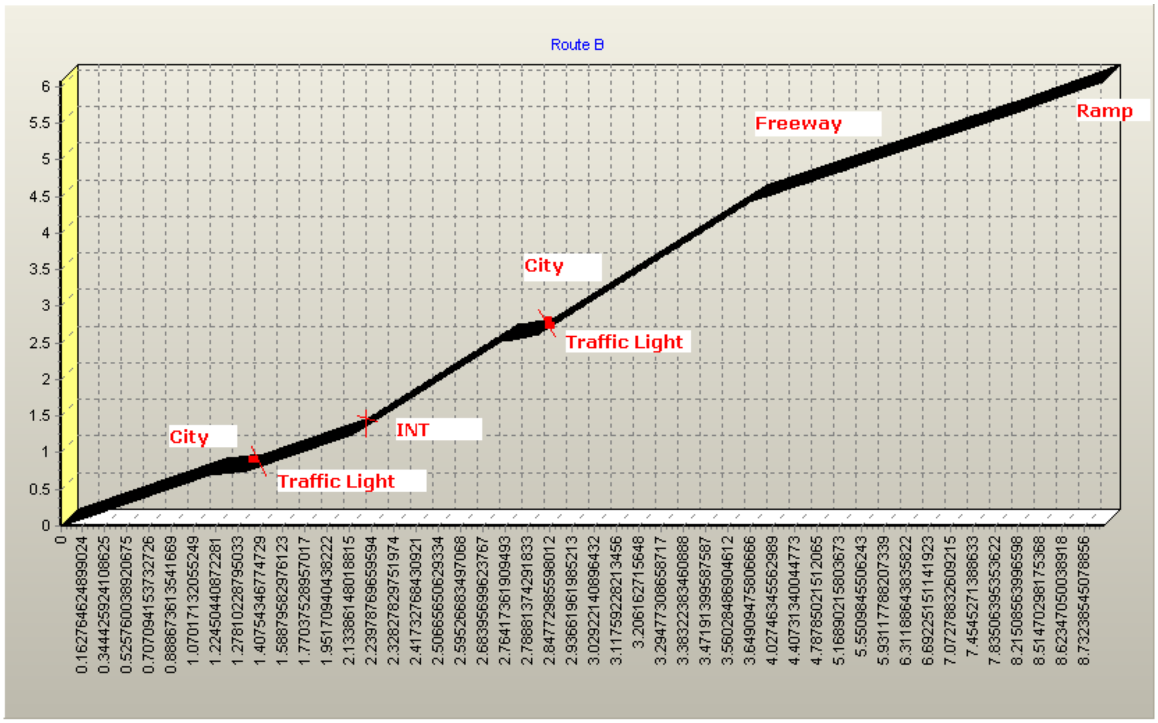


Figure 6.3 Pictorial Representation of Route B

Table 6.4 Route C

ROUTE_ID	3	3	3	3	3
ROAD_ID	1	2	3	4	5
ROAD_NAME	FREEWAY	CITY	FREEWAY	FREEWAY	CITY
DESCRIPTION	FREEWAY driving	CITY driving	FREEWAY driving	FREEWAY driving	CITY driving
ROAD_LENGTH	4	2.8	4.2	2.5	3
ANGLE	5	85	45	26	2
ROAD_TYPE	FREEWAY	CITY	FREEWAY	FREEWAY	CITY
TRAFFIC	0	0	0	0	0
INTERSECTION	0	0	0	0	0
MAXSPEED	75	50	75	75	50

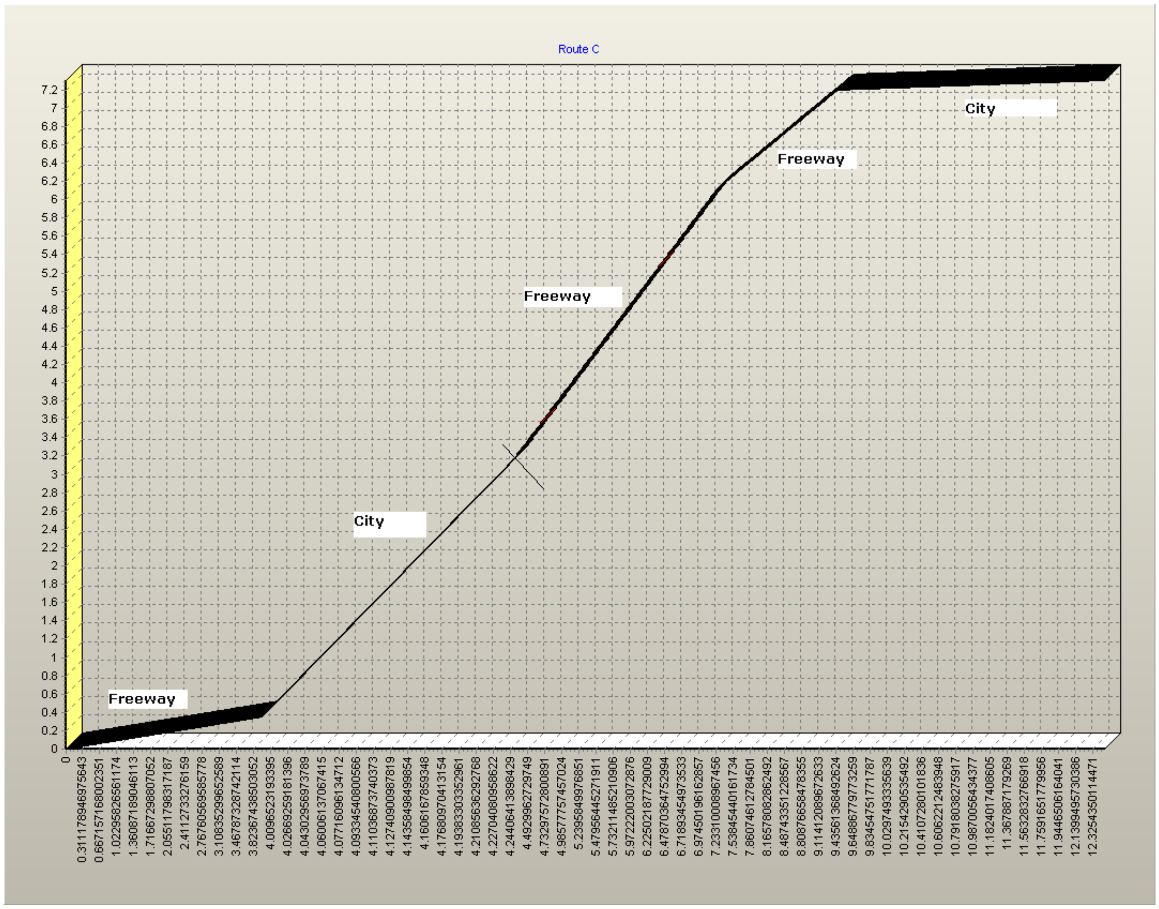


Figure 6.4 Pictorial Representation of Route C

The attributes that describe a given route were presented in Tables 6.2, 6.3 and 6.4. The description of these attributes is presented in Table 6.5. These attributes are in the environment (world) table in the database.

Table 6.5 Environment Attributes

ATTRIBUTE NAME	DESCRIPTION	EXAMPLE
ROUTE_ID	The value of this attribute distinguishes the various routes in the environment. This attribute is unique amongst all routes	A ROUTE_ID of 1, 2 or 3 is defined in the prototype because we have 3 distinct routes.
ROAD_ID	This attribute distinguishes the various road segments available in a route	ROAD_ID = 1, 2….n The total number of ROAD_ID's available in a route is dependent on the number of road segments in that route
ROAD_NAME	The name of the road segment	ROAD_NAME = "PARKING_LOT", "RAMP", "CITY", "FREEWAY"…..
ROAD_LENGTH	The length of a road segment.	ROAD_LENGTH = 4, means the road segment is 4 miles long
ANGLE	The angle of the road segment when placed in an X-Y Cartesian plane	ANGLE = 30 means the road segment is 30 degrees in an X-Y plane
ROAD_TYPE	The type of road segment.	ROAD_TYPE = "CITY ROAD", "FREEWAY"
TRAFFIC	If a traffic light exists in a road segment	This is a Boolean variable, with a value of true or false. If the value is true, records in the TLIGHT_POSITION table exist with positions of the traffic light set to different values, for example 0.3, 1.5 means the distance of the traffic light signal from the beginning of the road segment is 0.3 miles and 1.5 miles
INTERSECTION	If an intersection exists in a road segment	This is a Boolean variable, with a value of true or false. If the value is true, records in the INTERSECTION_POSITION table exist with positions of the intersection set to different values, for example INTERSECTION = 0.6 means the distance of the intersection from the beginning of the road segment is 0.6 miles
MAX_SPEED	The maximum speed a car should attain in a road segment	MAX_SPEED = 75 means an automobile driver can attain a maximum speed of 75 miles per hour in the road segment

The relationship between the tables that form the environment is presented in figure 6.5. From figure 6.5, the prototype environment consists of 5 tables:

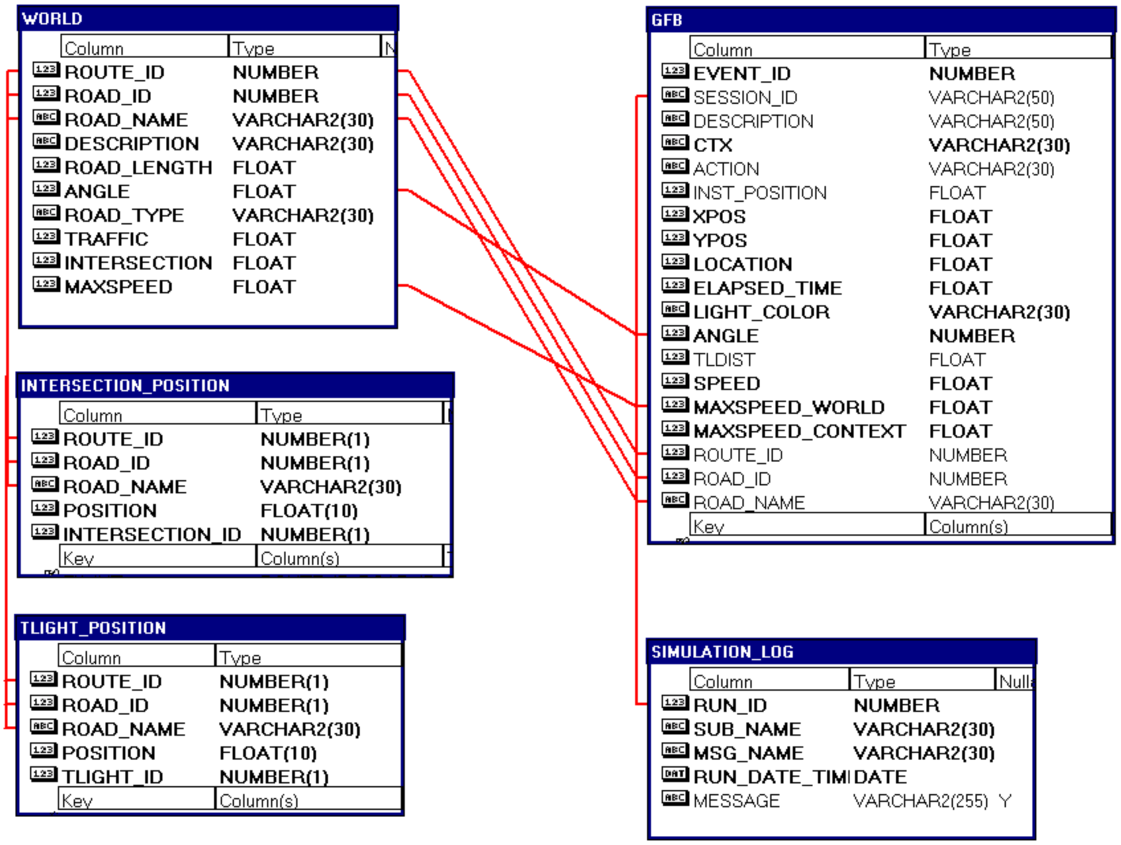
World: This table stores information about the route, for example road_id, route_id, road_name, and others as shown in figure 6.5.

Intersection_Position: This table stores information about the intersections on the route, for example, the road segment the intersection occurs and the position on the road segment.

Tlight_Position: This table stores information about the traffic lights on the route, for example, the road segment the traffic light occurs and the position(s) of the traffic light on the road segment.

G*fb:* This table is the global_fact_base that stores information on all activities of the agent in the environment, for example, the location of the agent on the route at a given time, the action performed by the agent, etc.

S*imulation_log:* This table stores information on the entire simulation process, for example, the procedures & functions called, the date and time the procedure is run, the procedure message or error message if there is an error, etc.



WORLD

Column	Type	N
ROUTE_ID	NUMBER	
ROAD_ID	NUMBER	
ROAD_NAME	VARCHAR2(30)	
DESCRIPTION	VARCHAR2(30)	
ROAD_LENGTH	FLOAT	
ANGLE	FLOAT	
ROAD_TYPE	VARCHAR2(30)	
TRAFFIC	FLOAT	
INTERSECTION	FLOAT	
MAXSPEED	FLOAT	

GFB

Column	Type
EVENT_ID	NUMBER
SESSION_ID	VARCHAR2(50)
DESCRIPTION	VARCHAR2(50)
CTX	VARCHAR2(30)
ACTION	VARCHAR2(30)
INST_POSITION	FLOAT
XPOS	FLOAT
YPOS	FLOAT
LOCATION	FLOAT
ELAPSED_TIME	FLOAT
LIGHT_COLOR	VARCHAR2(30)
ANGLE	NUMBER
TLDIST	FLOAT
SPEED	FLOAT
MAXSPEED_WORLD	FLOAT
MAXSPEED_CONTEXT	FLOAT
ROUTE_ID	NUMBER
ROAD_ID	NUMBER
ROAD_NAME	VARCHAR2(30)
Key	Column(s)

INTERSECTION_POSITION

Column	Type
ROUTE_ID	NUMBER(1)
ROAD_ID	NUMBER(1)
ROAD_NAME	VARCHAR2(30)
POSITION	FLOAT(10)
INTERSECTION_ID	NUMBER(1)
Key	Column(s)

TLIGHT_POSITION

Column	Type
ROUTE_ID	NUMBER(1)
ROAD_ID	NUMBER(1)
ROAD_NAME	VARCHAR2(30)
POSITION	FLOAT(10)
TLIGHT_ID	NUMBER(1)
Key	Column(s)

SIMULATION_LOG

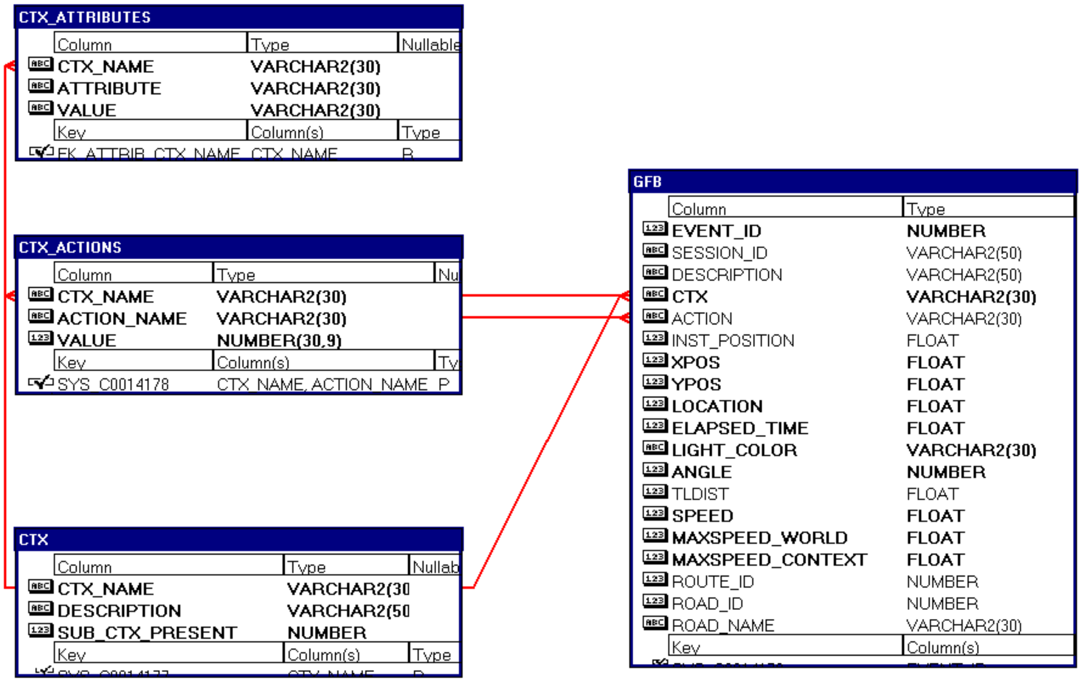
Column	Type	Null
RUN_ID	NUMBER	
SUB_NAME	VARCHAR2(30)	
MSG_NAME	VARCHAR2(30)	
RUN_DATE_TIMI	DATE	
MESSAGE	VARCHAR2(255)	Y

Figure 6.5 Entity Relationship (ER) Diagram of Environment Tables

6.4.2 Context Infrastructure

The context infrastructure required by the model enhancement prototype consists of the context library and all modules / functions that enable the control of the agents' actions and behavior by the contexts. The context library as described in the previous chapter consists of the context definitions, the actions defined in a context as well as the attributes for activating and deactivating a context. These are all defined in database tables. The choice of database table is to allow for the efficient modification of existing contexts as well as the introduction of new contexts. There are three tables in the context library, the

CTX, CTX_ACTIONS and CTX_ATTRIBUTRES tables. The relationship between these tables as well as the attributes of the tables are shown in figure 6.6



CTX_ATTRIBUTES

Column	Type	Nullable
CTX_NAME	VARCHAR2(30)	
ATTRIBUTE	VARCHAR2(30)	
VALUE	VARCHAR2(30)	

Key	Column(s)	Type
FK_ATTRIB_CTX_NAME	CTX_NAME	R

CTX_ACTIONS

Column	Type	Nu
CTX_NAME	VARCHAR2(30)	
ACTION_NAME	VARCHAR2(30)	
VALUE	NUMBER(30,9)	

Key	Column(s)	Ty
SYS_C0014178	CTX_NAME, ACTION_NAME	P

CTX

Column	Type	Nullab
CTX_NAME	VARCHAR2(30	
DESCRIPTION	VARCHAR2(50	
SUB_CTX_PRESENT	NUMBER	

Key	Column(s)	Type

GFB

Column	Type
EVENT_ID	NUMBER
SESSION_ID	VARCHAR2(50)
DESCRIPTION	VARCHAR2(50)
CTX	VARCHAR2(30)
ACTION	VARCHAR2(30)
INST_POSITION	FLOAT
XPOS	FLOAT
YPOS	FLOAT
LOCATION	FLOAT
ELAPSED_TIME	FLOAT
LIGHT_COLOR	VARCHAR2(30)
ANGLE	NUMBER
TLDIST	FLOAT
SPEED	FLOAT
MAXSPEED_WORLD	FLOAT
MAXSPEED_CONTEXT	FLOAT
ROUTE_ID	NUMBER
ROAD_ID	NUMBER
ROAD_NAME	VARCHAR2(30)

Key	Column(s)

Figure 6.6 Relationships Between Tables in Context Library and Their Attributes

Table 6.6 Context Definitions

CTX_NAME	DESCRIPTION	SUB_CTX_PRESENT
PARKING_LOT	PARKING LOT CONTEXT	1
CITY	CITY DRIVING CONTEXT	1
TRAFFIC_LIGHT	TRAFFIC LIGHT CONTEXT	0
INTERSECTION	INTERSECTION CONTEXT	0
FREEWAY	FREEWAY DRIVING	0
RAMP	RAMP TO FREEWAY	0

Table 6.7 Context Attributes

CTX_NAME	ATTRIBUTE	VALUE
PARKING_LOT	MAXSPEED	10
PARKING_LOT	COMPATIBLE	CITY
PARKING_LOT	COMPATIBLE	INTERSECTION
CITY	MAXSPEED	35
CITY	COMPATIBLE	TRAFFIC_LIGHT
CITY	COMPATIBLE	INTERSECTION
CITY	COMPATIBLE	RAMP
CITY	COMPATIBLE	PEDESTRIAN
TRAFFIC_LIGHT	MAXSPEED	0
INTERSECTION	MAXSPEED	0
RAMP	MAXSPEED	20
FREEWAY	MAXSPEED	55

Table 6.8 Context Actions

CTX_NAME	ACTION_NAME	VALUE
INTERSECTION	INCREASE_SPEED_1	5
INTERSECTION	INCREASE_SPEED_2	10
INTERSECTION	INCREASE_SPEED_3	15
INTERSECTION	INCREASE_SPEED_4	20
INTERSECTION	INCREASE_SPEED_ADD_1	1
INTERSECTION	MAINTAIN_SPEED	0.000001
INTERSECTION	REDUCE_SPEED_1	-5
INTERSECTION	REDUCE_SPEED_2	-10
INTERSECTION	REDUCE_SPEED_3	-15
INTERSECTION	REDUCE_SPEED_4	-20
INTERSECTION	REDUCE_SPEED_MINUS_1	-1
INTERSECTION	STOP	0
CITY	INCREASE_SPEED_ADD_1	1
CITY	INCREASE_SPEED_1	5
CITY	INCREASE_SPEED_2	10
CITY	INCREASE_SPEED_3	15
CITY	INCREASE_SPEED_4	20
CITY	MAINTAIN_SPEED	0.000001
CITY	REDUCE_SPEED_MINUS_1	-1
CITY	REDUCE_SPEED_1	-5
CITY	REDUCE_SPEED_2	-10
CITY	REDUCE_SPEED_3	-15
CITY	REDUCE_SPEED_4	-20
CITY	STOP	0
FREEWAY	INCREASE_SPEED_ADD_1	1
FREEWAY	INCREASE_SPEED_1	5
FREEWAY	INCREASE_SPEED_2	10
FREEWAY	INCREASE_SPEED_3	15
FREEWAY	INCREASE_SPEED_4	20
FREEWAY	MAINTAIN_SPEED	0.000001

CTX_NAME	ACTION_NAME	VALUE
FREEWAY	REDUCE_SPEED_MINUS_1	-1
FREEWAY	REDUCE_SPEED_1	-5
FREEWAY	REDUCE_SPEED_2	-10
FREEWAY	REDUCE_SPEED_3	-15
FREEWAY	REDUCE_SPEED_4	-20
FREEWAY	STOP	0
PARKING_LOT	INCREASE_SPEED_ADD_1	1
PARKING_LOT	INCREASE_SPEED_1	5
PARKING_LOT	INCREASE_SPEED_2	10
PARKING_LOT	INCREASE_SPEED_3	15
PARKING_LOT	INCREASE_SPEED_4	20
PARKING_LOT	MAINTAIN_SPEED	0.000001
PARKING_LOT	REDUCE_SPEED_MINUS_1	-1
PARKING_LOT	REDUCE_SPEED_1	-5
PARKING_LOT	REDUCE_SPEED_2	-10
PARKING_LOT	REDUCE_SPEED_3	-15
PARKING_LOT	REDUCE_SPEED_4	-20
PARKING_LOT	STOP	0
RAMP	INCREASE_SPEED_ADD_1	1
RAMP	INCREASE_SPEED_1	5
RAMP	INCREASE_SPEED_2	10
RAMP	INCREASE_SPEED_3	15
RAMP	INCREASE_SPEED_4	20
RAMP	MAINTAIN_SPEED	0.000001
RAMP	REDUCE_SPEED_MINUS_1	-1
RAMP	REDUCE_SPEED_1	-5
RAMP	REDUCE_SPEED_2	-10
RAMP	REDUCE_SPEED_3	-15
RAMP	REDUCE_SPEED_4	-20
RAMP	STOP	0
TRAFFIC_LIGHT	INCREASE_SPEED_ADD_1	1
TRAFFIC_LIGHT	INCREASE_SPEED_1	5
TRAFFIC_LIGHT	INCREASE_SPEED_2	10
TRAFFIC_LIGHT	INCREASE_SPEED_3	15
TRAFFIC_LIGHT	INCREASE_SPEED_4	20
TRAFFIC_LIGHT	MAINTAIN_SPEED	0.000001
TRAFFIC_LIGHT	REDUCE_SPEED_MINUS_1	-1
TRAFFIC_LIGHT	REDUCE_SPEED_1	-5
TRAFFIC_LIGHT	REDUCE_SPEED_2	-10
TRAFFIC_LIGHT	REDUCE_SPEED_3	-15
TRAFFIC_LIGHT	REDUCE_SPEED_4	-20
TRAFFIC_LIGHT	STOP	0

The defined contexts required for building the prototype are shown in Table 6.6, their attributes are shown in Table 6.7 and their actions in Table 6.8. These contexts include: FREEWAY-DRIVING, CITY-DRIVING, DIRT-DRIVING, RAMP-DRIVING, and

PARKING-LOT. Sub-contexts required include: INTERSECTION and TRAFFIC-LIGHT. DIRT-DRIVING isn't included in the hand-built original model because the agent is expected to learn about it.

A design decision was made to represent 'RAMP', 'DIRT' and 'PARKING-LOT' as major contexts. This decision is based on the previously-mentioned coding standard that limits the context depth to 2. Arguments can be made against making these three contexts major contexts. Figures 6.7 and 6.8 show the context topology of the prototype with 'DIRT-DRIVING' and 'RAMP-DRIVING' as major context and as sub-contexts respectively.

The creation of DIRT-DRIVING context was omitted to prove the agent can learn to create contexts after learning from its interactions with the environment.

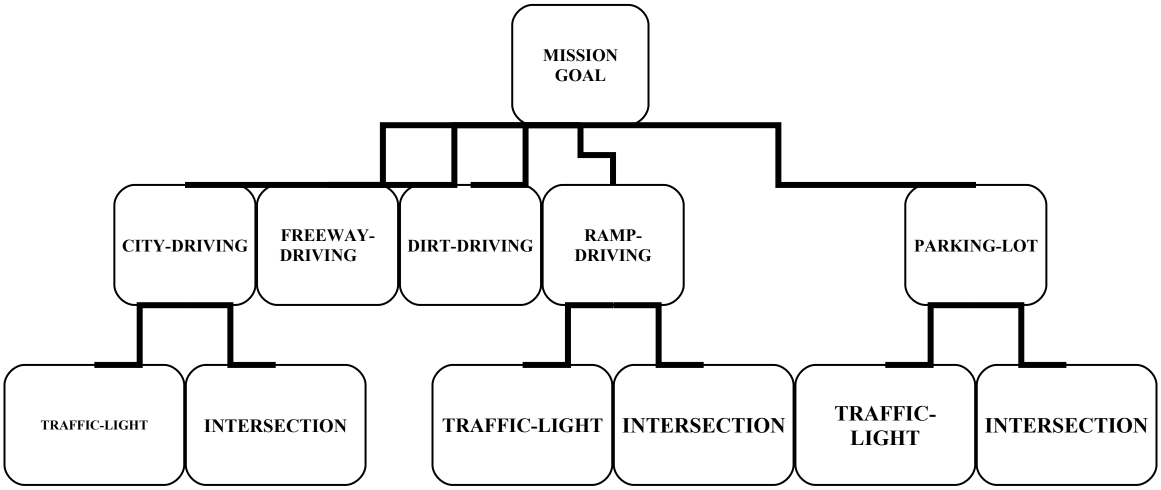


Figure 6.7 Context Topology Showing RAMP-DRIVING and DIRT-DRIVING as Major Contexts

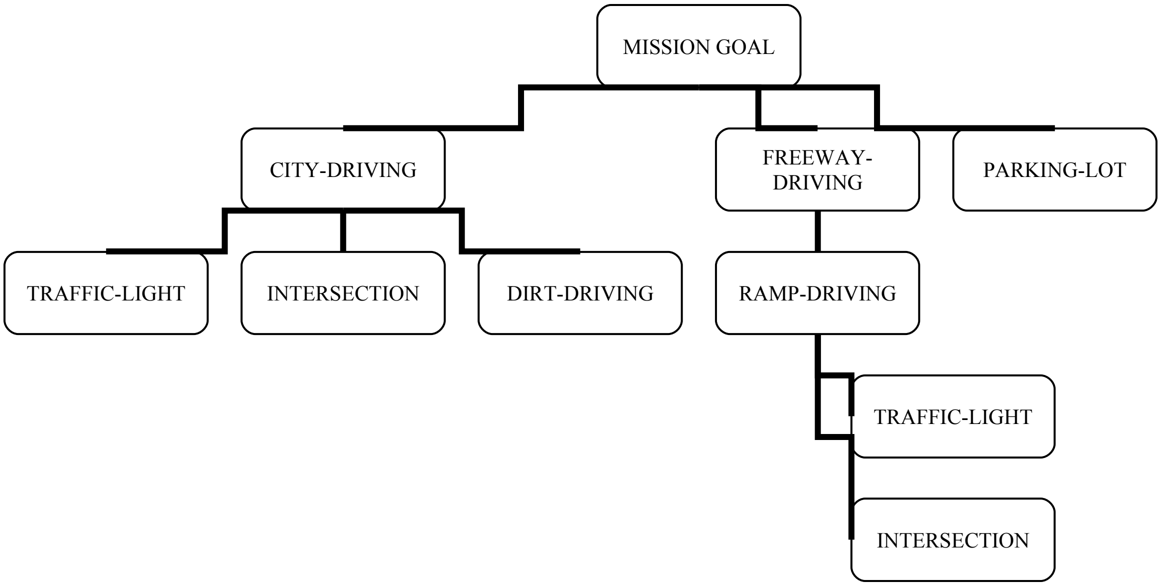


Figure 6.8 Context Topology Showing RAMP-DRIVING and DIRT-DRIVING as Sub-Contexts

The other parts of the context infrastructure are the modules / functions that tie these tables together as well as the context logic. These functions are described in subsequent sub-sections of this chapter. First, a description of the redesign of contexts to enable learning is presented.

6.4.2.1 Base Hand-Built Model

A hand-built model of a person driving from home to work was implemented as the base model prior to the learning process. This hand-built model is what is enhanced by the enhancement technique to improve its overall performance and behavior of the agent while it achieves its mission goal. The performance improved is the total time used to arrive at the destination and the arrival of the agent at the destination when no context is defined. The improved behaviors are the agents' behaviors at a traffic light and intersection. The context topology of the base model is as shown in figure 6.7. The contexts are unique records in the context database table. The context definitions are as shown in tables 6.6, 6.7 & 6.8.

6.4.2.2 Redesigning a Context to Enable Learning

As stated earlier, a redesign of the context architecture is necessary to allow the modification of contexts and creation of new contexts during the agents' interaction with its environment. This is achieved by replacing hard-coded constants with variables. The replacement of the constants with variables allow for the seamless modification of the variables during the course of the agents' interaction with its environment in a simulator. These variables are stored within tables in a database. As the agent undergoes training, these variables are modified until a value equal to or close to the value in the environment (based on the mission goal) is achieved. This value henceforth be referred to as the optimal value within the context of this dissertation. An optimal value is determined when the value of a variable converges to a single value and/or the change in the value of that variable becomes negligibly small after multiple simulation cycles.

The question of what part of the context to replace with variables arises. Does one replace the action rules, transition rules, contextual values (e.g. maximum speed limit) – attributes with variables? If these are all replaced with variables there is a tendency for the agent to primarily learn everything from the beginning because these values may not reflect what is in the environment. Learning from the beginning is acceptable, but a balanced solution will be to provide the context with values of some or all attributes that can be modified. For example, providing a maximum speed limit as a variable in a database table, and also providing actions in the model that will enable the agent to learn what the true maximum speed limit of a road segment is. In this prototype, all constants were replaced with variables but the learning mechanism is designed to learn the maximum speed variable. Tables 6.6, 6.7 & 6.8 show the attributes and actions with their values replaced with variables

6.4.3 Sentinel Logic

The sentinel module searches to see whether the context attributes no longer match the current situation and whether the end of a simulation cycle has been reached. It achieves this by calculating the position of the agent and the defined location of the current road segment relative to the start of journey. If the current position of the agent falls outside the defined range of the context, the sentinel module attempt to sense the road type/road segment on which the agent is currently, and then it activates the context that is defined for that road segment, if one exists. If no context matches the definition of the current road type, it calls the context modifier which searches through the contexts to see if any context can be modified to meet the definition of the current position (based on the number of attributes in the context that match the attributes of the current position). If no context can be modified, the context creation module is called which creates a context from the context template.

The sentinel module also identifies the end of a simulation cycle and calls the reward function to reward the agent appropriately. The pseudo code is shown below:

- *note the total length of the current road segment*
- *check the current position of the agent, if the agents' current position is outside the range defined for the road segment*

- *deactivate the current active context*
- *sense the road type and other information of the road segment on which the agent currently is*
- *search through the context library to identify a context whose attributes match the attributes of the current position*
- *if context is found*
    - *activate the context and let the control of the agent be guided by the defined actions and attributes of the context*
- *if no context is found*
    - *activate context modifier and then context creation modules*
- *if the simulation cycle is complete*
    - *call the reward function to assign an appropriate reward to the agent*

The sentinel_rule procedure takes as input the current position of the agent and outputs the current context, the traffic light position if any exists on the current road segment, the intersection position if any exists in the current road segment, the road id of the current road segment and the run id of the current simulation run.

PROCEDURE "SENTINEL_RULES"				
INPUTS			Agents current position	Run ID
OUTPUTS	Active Context	Traffic light and position on road segment	Intersection & position on road segment	Road ID
DATATYPE	Variable Character	Float	Float	Integer

6.4.4 Context Modifier

The context modifier module modifies existing attributes and actions in a context to enhance them. In the prototype, the modifier module serves two purposes, one is to modify the contextual attribute of maximum speed during training to enhance the agents performance, and the second is to modify a context whose attributes closely match the attributes of the agents' current position (environment) to enable the creation of a new context or modification of an existing context to include the current situation where one is not defined.

The modifier module is activated every time the agent is undergoing training. The values of the maximum speed in the context_attribute table containing the context attributes are modified randomly in the beginning and then after 20 simulation cycles, the value of the maximum speed attribute is modified based on the values learned during the first 20 simulation cycles, i.e. the value with the most reward. The pseudo code below shows the design of the modifier.

- *Upon activation of the modifier for this simulation cycle, note the context undergoing training*
- *Get the current value of the attribute(s) being modified – based on the mission goal. In the prototype, the maximum speed value is being modified.*
- *CASE A: training context has gone between 0 and 20 training cycles, then*
    - *Randomly select new values for the attribute(s) from the action base*

- *Apply the selected values from the action base to the existing values to come up with new values for the (maximum speed) attribute(s)*
- *Use the newly calculated values to update the context attributes table for the maximum speed attributes identified*

- *CASE B: context has gone between 21 and 40 simulation cycles, then*
  - *Modify the maximum speed attribute with the value that appears to generate more rewards from the environment.*
- *CASE C: context has gone between 41 and 50 simulation cycles, then*
  - *Randomly modify the maximum speed value*
- *CASE D: context has gone between 51 and 60, then*
  - *Modify the maximum speed attribute based on the value that has generated the most rewards in previous simulation cycles, taking note of the current maximum_speed value and the previous simulation run maximum_speed value.*
- *Deactivate the modifier, passing out the newly updated values of the attributes*

Note that the values of “0 to 20”, “21 to 40”, etc. in cases A through D above where chosen after initial test runs to see how the maximum speed attribute converges with different reward values.

PROCEDURE "CTX_MODIFIER"			
INPUTS	Learning CTX		Run ID
OUTPUTS		New CTX attribute (maximum speed)	
DATATYPE	Variable Character	Float	Integer

6.4.5 Context Creator

The context creator module creates new contexts that attempt to address situations not defined by the SME. It achieves this by randomly copying an existing context and then modifies the copied context to address the unknown situation. Typically, when human behavioral agents encounter unknown situations in a simulated environment, they either raise an exception or fail. In the prototype, when the CxBR agent encounters unknown situations, a search through the context library is carried out to identify a context that can be modified to fit the current situation. A context that can be modified to fit the current situation is determined as described in Chapter 5.

If no context can be modified to fit the current situation, the context creator module is activated. This module randomly copies an existing context from the context library and sets the name to the name of the event. The modifier is then activated to modify the attributes and actions of the newly copied context to fit the current situation.

The pseudo code for achieving this is described below:

- *Upon activation of the context creator module, randomly copy an existing context from the context table as well as the actions and attributes of this context from the context action and context attribute tables respectively*
- *Set the name of the newly copied context to match the event of the environment without a context*
- *Activate the context modifier module to modifier the attributes and actions of this context to their optimal values*

6.4.6 Designing the Reward Function

The reward function drives (controls) the agents' learning process. The design of the reward function is based on the mission goal. There are some rules that govern the design of reward functions:

a) The reward function should not contain a reward or punishment for an action. In other words, the system should not reward or punish the agent for performing a particular action or group of actions. This is so because the agent is not supposed to know the best action in any given state. If it did, the problem would be minimized to a supervised learning problem where the agent is rewarded or punished if its actions are right or wrong.

b) The reward function should contain only definitions of states, i.e. the agent is rewarded for being in a given state. This state could be the goal state or states leading to it. The actions that lead to these states are unknown to the agent and the agent is expected to learn them. An example of a reward function in the model enhancement prototype is rewarding the agent for arriving at a state where the maximum speed for the context being trained is equal to the maximum speed of the road segment that the context represents in the world. Another example of a reward function when the mission goal is to choose the shortest distance from point A to point B amongst various routes available. This would mean designing the reward function in such a way that the agent is rewarded positively for being at the state where the total distance at the end of the simulation cycle is less than the previous total distance when the agent used another route; the agent is punished at the end of the simulation cycle for being in a state where the current total distance is greater than the previous total distance.

The overall design of the reward function for the model enhancement prototype is based on the stated mission goals of the prototype which are: 1) to improve the performance of the agent in terms of arrival time at destination; 2) find the fastest route between the start and end positions; 3) Learn the attributes of an undefined / missing road segment. In order to achieve either of these goals, the agent encounters situations when one or more segments in a route are unknown or undefined and also when the defined maximum speed limits in contexts are different from what actually prevails in the environment.

The agent is rewarded for being in the goal state, i.e. at the end of each simulation cycle the distance traveled or the total time between the previous simulation run is compared to the distance traveled or the total time of the current simulation run. If the distance traveled or the total time traveled is less for this simulation run and the routes are different, the agent is rewarded positively. If the routes are the same between the current simulation run and the previous simulation run, the agent doesn't receive any reward. If the previous simulation run produces a shorter time, the current simulation run is rewarded negatively.

To arrive at the choice of route, the individual contexts in a sample route must be trained to learn the actual maximum speed defined in the environment. By learning the maximum speed, the overall performance of the agent is improved.

Some questions that might arise are: how to choose the value for the reward the agent receives – will there be a difference in the learning process if a reward of 100 points is given to the agent versus a reward of 10 points? What about if the signs of the reward change (a positive reward is changed to a punishment) or if no reward is issued. These questions are answered in the next chapter during the evaluation of the results.

Reward function pseudo code: A note should be made that the simulation runs for each mission goal is different. The reward presented below is generic and applies to the mission goals of identifying the shortest time.

- At the end of the simulation cycle, check to see the total time traveled from the beginning of the simulation cycle to the end of the simulation cycle.
- If this value is greater than the value for the previous run stored in the rewards table, and the route for the previous run and the current run are different, punish the agent (give the agent a negative reward) and store this information in the rewards table. If the routes for the previous run and current run are the same, do not punish or reward the agent, i.e. give the agent a reward of 0.
- If this value is less than the previous run, and the route between both simulation cycles runs are different, reward the agent and store this information in the rewards table. If the routes are the same, give the agent a reward of 0.

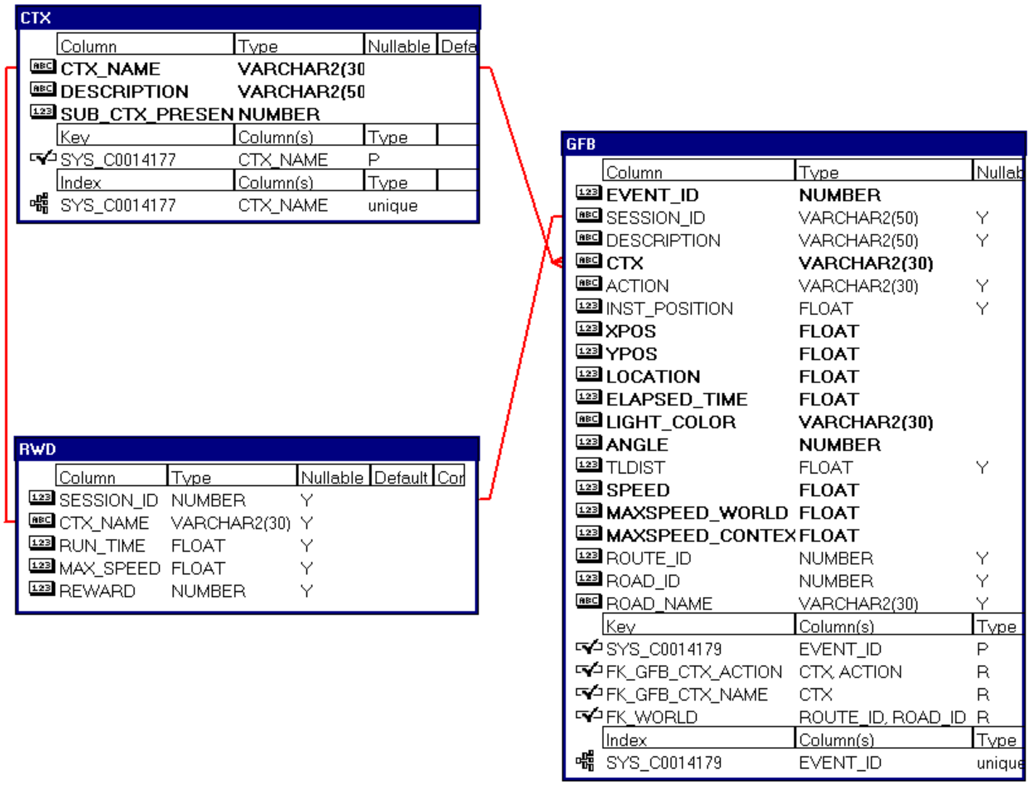
PROCEDURE "REWARD"			
INPUTS	Learning CTX		Run ID
OUTPUTS		Reward	
DATATYPE	Variable Character	Integer	Integer

The pseudo code for the reward function used in training the agent to learn the maximum speed is presented below:

- Compare the maximum speed of the context being trained with the maximum speed defined in the environment

- If the context maximum speed is greater than the maximum speed defined for the road segment in the environment, then assign a reward of -20 (assign a negative reward)
- If the context maximum speed is equal to the maximum speed defined for the road segment in the environment, then assign a reward of +50 (assign a large positive reward)
- If the context maximum speed is less than the maximum speed defined for the road segment minus 5, then assign a reward of -10
- If the context maximum speed is less than or equal to the previous maximum speed learnt for that training context, then assign a reward of -1
- If the context maximum speed is greater than the previously learnt maximum speed for the context, then assign a reward of +1
- Insert what has just been learnt into the reward table.

The reward table stores all information about the rewards received by the agent and the context that caused the reward along with the maximum speed of the context and the run time of the simulation cycle. Figure 6.9 shows the relationship between the reward table, context table and global fact base table. The definitions of the columns in the reward table are also shown.



CTX

Column	Type	Nullable	Defa
CTX_NAME	VARCHAR2(30		
DESCRIPTION	VARCHAR2(50		
SUB_CTX_PRESEN	NUMBER		

Key	Column(s)	Type
SYS_C0014177	CTX_NAME	P

Index	Column(s)	Type
SYS_C0014177	CTX_NAME	unique

RWD

Column	Type	Nullable	Default	Cor
SESSION_ID	NUMBER	Y		
CTX_NAME	VARCHAR2(30)	Y		
RUN_TIME	FLOAT	Y		
MAX_SPEED	FLOAT	Y		
REWARD	NUMBER	Y		

GFB

Column	Type	Nullab
EVENT_ID	NUMBER	
SESSION_ID	VARCHAR2(50)	Y
DESCRIPTION	VARCHAR2(50)	Y
CTX	VARCHAR2(30)	
ACTION	VARCHAR2(30)	Y
INST_POSITION	FLOAT	Y
XPOS	FLOAT	
YPOS	FLOAT	
LOCATION	FLOAT	
ELAPSED_TIME	FLOAT	
LIGHT_COLOR	VARCHAR2(30)	
ANGLE	NUMBER	
TLDIST	FLOAT	Y
SPEED	FLOAT	
MAXSPEED_WORLD	FLOAT	
MAXSPEED_CONTEX	FLOAT	
ROUTE_ID	NUMBER	Y
ROAD_ID	NUMBER	Y
ROAD_NAME	VARCHAR2(30)	Y

Key	Column(s)	Type
SYS_C0014179	EVENT_ID	P
FK_GFB_CTX_ACTION	CTX, ACTION	R
FK_GFB_CTX_NAME	CTX	R
FK_WORLD	ROUTE_ID, ROAD_ID	R

Index	Column(s)	Type
SYS_C0014179	EVENT_ID	unique

Figure 6.9 Reward table definition and relationships.

6.5 Main Function

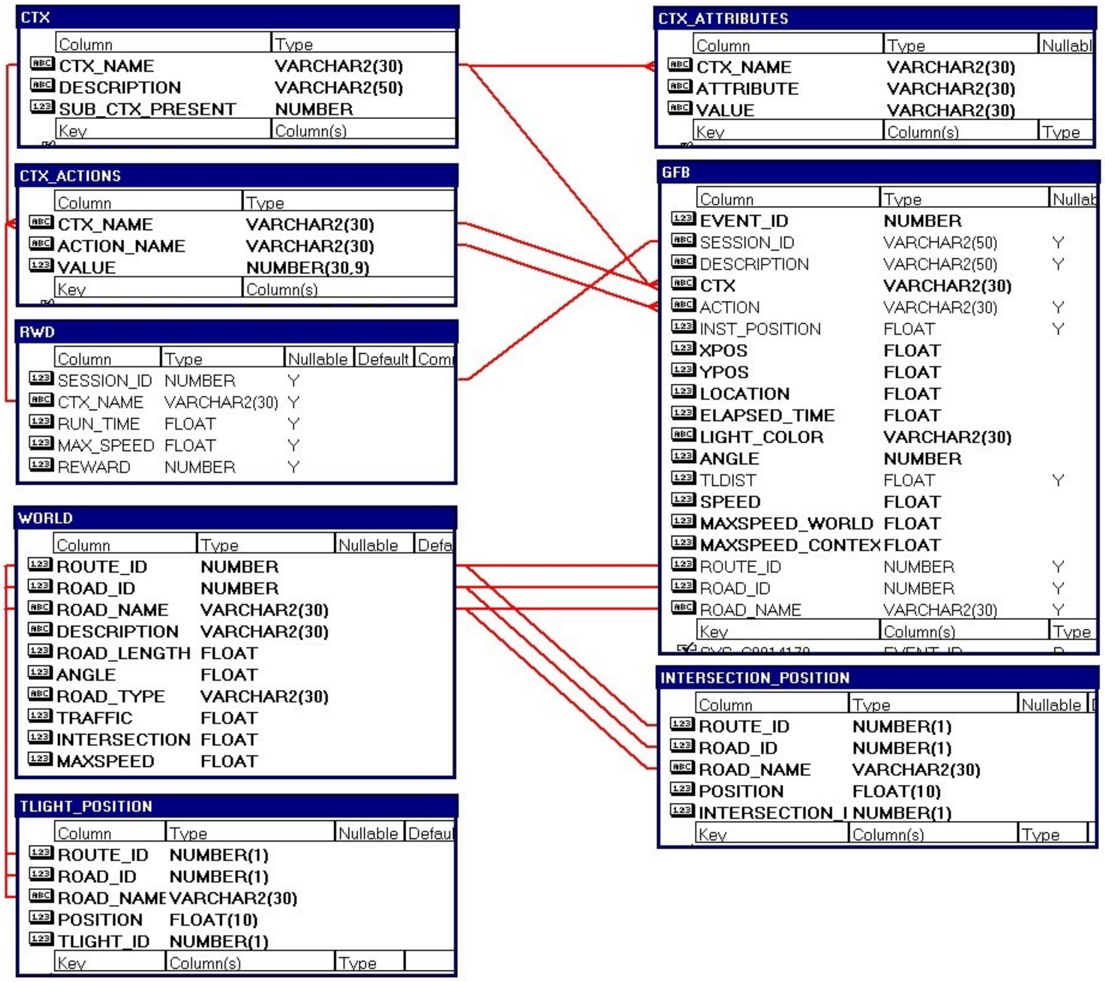
PROCEDURE "RUN_RCXBR"	
INPUTS	Learning CTX
OUTPUTS	
DATATYPE	Variable Character

The main function calls all procedures and functions that enable the simulation of the agents' behavior. The pseudo code is presented below:

- *Define all appropriate variables*
- *Get the count of rewards for the context being trained to determine if training should continue or not*
- *If training should continue*

```
        o  Call the context modifier module
•  Generate a distinct run id for this simulation cycle
•  While the end of simulation has been reached, loop through
        o  Randomly generate a traffic light color
        o  Call the sentinel rule procedure to sense the current situation
        o  Based on the situation identified,  perform the actions of the
           controlling context
        o  Insert the event id, run id, ctx and other values in the global fact
           base
        o  Set the current distance to the new position
        o  Set the current speed to the new speed
        o  Set the previous time to the current time
        o  Calculate the elapsed time
•  End loop
•  Call the reward function to assign an appropriate reward for this
   simulation cycle.
```

The relationship between all the tables in the simulation is presented below in figure 6.10.



CTX

Column	Type
CTX_NAME	VARCHAR2(30)
DESCRIPTION	VARCHAR2(50)
SUB_CTX_PRESENT	NUMBER

Key	Column(s)

CTX_ACTIONS

Column	Type
CTX_NAME	VARCHAR2(30)
ACTION_NAME	VARCHAR2(30)
VALUE	NUMBER(30,9)

Key	Column(s)

RWD

Column	Type	Nullable	Default	Com
SESSION_ID	NUMBER	Y		
CTX_NAME	VARCHAR2(30)	Y		
RUN_TIME	FLOAT	Y		
MAX_SPEED	FLOAT	Y		
REWARD	NUMBER	Y		

WORLD

Column	Type	Nullable	Defa
ROUTE_ID	NUMBER		
ROAD_ID	NUMBER		
ROAD_NAME	VARCHAR2(30)		
DESCRIPTION	VARCHAR2(30)		
ROAD_LENGTH	FLOAT		
ANGLE	FLOAT		
ROAD_TYPE	VARCHAR2(30)		
TRAFFIC	FLOAT		
INTERSECTION	FLOAT		
MAXSPEED	FLOAT		

TLIGHT_POSITION

Column	Type	Nullable	Defaul
ROUTE_ID	NUMBER(1)		
ROAD_ID	NUMBER(1)		
ROAD_NAME	VARCHAR2(30)		
POSITION	FLOAT(10)		
TLIGHT_ID	NUMBER(1)		

Key	Column(s)	Type	

CTX_ATTRIBUTES

Column	Type	Nullabl
CTX_NAME	VARCHAR2(30)	
ATTRIBUTE	VARCHAR2(30)	
VALUE	VARCHAR2(30)	

Key	Column(s)	Type

GFB

Column	Type	Nullab
EVENT_ID	NUMBER	
SESSION_ID	VARCHAR2(50)	Y
DESCRIPTION	VARCHAR2(50)	Y
CTX	VARCHAR2(30)	
ACTION	VARCHAR2(30)	Y
INST_POSITION	FLOAT	Y
XPOS	FLOAT	
YPOS	FLOAT	
LOCATION	FLOAT	
ELAPSED_TIME	FLOAT	
LIGHT_COLOR	VARCHAR2(30)	
ANGLE	NUMBER	
TLDIST	FLOAT	Y
SPEED	FLOAT	
MAXSPEED_WORLD	FLOAT	
MAXSPEED_CONTEX	FLOAT	
ROUTE_ID	NUMBER	Y
ROAD_ID	NUMBER	Y
ROAD_NAME	VARCHAR2(30)	Y

Key	Column(s)	Type
SYS_C0014170	EVENT_ID	P

INTERSECTION_POSITION

Column	Type	Nullable
ROUTE_ID	NUMBER(1)	
ROAD_ID	NUMBER(1)	
ROAD_NAME	VARCHAR2(30)	
POSITION	FLOAT(10)	
INTERSECTION_I	NUMBER(1)	

Key	Column(s)	Type

Figure 6.10 Relationship between tables in the simulation

6.6 Training the Agent

Three smaller routes were used to train the agent. These routes consisted of all road types available in the three actual routes traversed by the agent. The description of the agent training is presented in the next chapter. As a primer, there were two learning strategies utilized in training the agent. In the first learning strategy, during training, the agent randomly picks a route or maximum speed value as the ideal value all through the training simulation. That is, in all simulation cycles, the agent chooses the attribute being learnt randomly. In the second learning strategy, some learning guidance is provided to the agent based on its previous choices. In this learning strategy, the agent initially chooses the attribute being learnt at random. After a certain number of simulation cycles, the agent evaluates what it has learnt so far and then chooses the attribute value that has given it the most reward thus far. It then randomly chooses a value again for a few more simulation cycles before utilizing the value with the most reward in all simulation cycles. That is, the agent learns randomly up to a point, then applies what it has learnt so far for a few simulation cycles, then learns in a random fashion again before eventually using what it learnt in all simulation cycles.

6.7 Chapter Summary

In this chapter a description of the prototype implementing the model enhancement technique was presented. The requirements and specifications of the prototype was outlined as well as assumptions made in the design and implementation of the prototype used in the evaluation of the model enhancement technique. The detailed design of the various modules / functions of the prototype were also presented. The tables used in storing the data used in the simulations are described along with their relationships. The training of the agent towards learning the optimal maximum speed for all contexts was also presented along with graphs showing the training process.

CHAPTER 7 EXPERIMENTS AND EVALUATION OF RESULTS

The experiments performed with the prototype were used to evaluate the concept set forth in Chapter 3. The overall goals of the agent in the prototype are to enhance the knowledge in the contexts acquired from SMEs' and correct any errors therein. Errors made by SMEs' can limit agents' behavior and/or performance. The kinds of errors an SME can make are grouped into three classes:

1) The SME can provide wrong information, for example, the SME can provide an incorrect speed limit for an automobile driver.
2) The SME can provide an incorrect process or incorrect procedures in a tactical situation. For example, the SME will not tell an automobile driver to stop at intersections with stop signs or to stop at red traffic lights.
3) The SME can omit a task in process and thus provide incomplete processes or procedures that are necessary to achieve a mission goal. For example, the SME can omit providing information on a road type in an automobile driving domain, and an agent may not know what to do upon encountering that road type.

The experiments performed in this chapter address the three classes of errors described above. A total of five experiments were performed. The goal of the first experiment is to show the dangers of using incorrect knowledge in decision making. In this experiment, the goal of the agent's mission is to find the tactically optimal route to its destination, i.e. the fastest possible time to its destination while adhering to all traffic rules and constraints. Note that the goal of the agent is different from the goal (the reason) for performing the experiment, that is, what the results of the experiment are supposed to show and the usefulness of the experiment. In performing the first experiment to find the tactically optimal route, a readers' initial thought on this experiment (finding the tactically optimal route to a destination) suggests it's a trivial problem easily solved through an optimization search. However, finding the tactically optimal route while working with incomplete or incorrect knowledge can lead to making the wrong decision. The investigation carried out in this dissertation does not simply find the tactically optimal route, it fills in the missing information and corrects the wrong information obtained from the SME; in other words, it breaks the SME knowledge barrier and thus leads to finding the correct tactically optimal route.

The goal of the second experiment is to resolve situations when the SME provides incorrect information, and show the impact this can have on the performance of an agent. For example, in an automobile driving domain, where the mission goal is to arrive at a destination as quickly as possible, the SME can provide an incorrect speed limit for an automobile driver. If the speed limit provided by the SME is less than what exists in the world, if the agent drives using the speed limit provided by the SME, the time it takes the agent to arrive at its destination will be longer than what it would have been if the agent were to drive with the actual speed limit that exists in the world. Conversely, if the speed limit provided by the SME is higher than what really exists in the world and if the agent uses the SME-provided speed limit, the time it takes the agent to arrive at its destination will also be longer because the agent driver can be stopped and delayed by police for driving above the speed limit on the road. Note that like the first experiment, the goal of the experiment differs from the mission goal of the agent in the experiment. This is true for all five experiments, i.e. the goal of performing the experiments differs from the mission goal of the agent in the experiment and the seeming triviality of the agents' mission goal is eclipsed by the underlying goal of performing the experiment.

The goal of the third experiment is to resolve situations when the SME provides incorrect processes or procedures in a tactical situation, and shows the impact this can have on the behavior of an agent. For example, using an automobile driving domain and the same example as in the second experiment where the agent has to arrive at its destination on time, the SME does not tell an automobile driver to stop at intersections with stop signs or to stop at red traffic lights. Not stopping at an intersection or at a red traffic light could have devastating effects, such as accidents or being ticketed by the police. The behavior of the agent at intersections and at traffic lights are monitored in this experiment.

The goal of the fourth experiment is to resolve situations when the SME omits information about a task in a process or the process itself, and thus provides incomplete processes or procedures that are necessary to achieve a mission goal. For example, in an automobile driving domain where the agent has a mission goal to arrive at its destination, the SME can omit providing information about a road type, thus an agent will not know what to do upon encountering that road type.

The goal of the fifth experiment is to show the technique developed in this investigation can be generalized to other domains other than the automobile driving domain. The experiment was performed in a tactical submarine warfare mission as described by Gonzalez and Ahlers [200]. In achieving this, the agent contends with and resolves the incorrect information about a submarines depth provided by the SME. More on this later.

There are two phases of the experiments. There is the training phase, were the agent learns the appropriate contextual attributes, thus becoming an enhanced agent and there is the execution phase, where the enhanced agent attempts to achieve its mission goal with the correct knowledge. A smaller dataset with shorter routes and different route compositions are used in the training phase. More on this later.

Note that the comparison in all experiments is performed between the enhanced agent and the base agent. That is, the already trained agent is compared to the untrained agent, which has no capabilities for learning in real time during the execution phase. More on this later.

7.1 Evaluation Criteria

A measure of the success of a new approach is achieved by evaluating the new approach in a controlled environment. A comparison of the new approach is carried out against previously established approaches (where they exist) on known or unknown problems. In this investigation, a CxBR agent enhanced by using the new approach is compared against the base CxBR agent. Several criteria are used in evaluating the enhancement technique. Recalling from Chapter 3, the overall goal of the enhancement technique is to enhance existing human behavior representation models created from knowledge collected from SME's. This knowledge could contain errors or be missing some relevant information as explained earlier in this chapter. The enhancement process creates an avenue for implicit knowledge to be included in the final enhanced model as well as creating new knowledge within the model. The enhanced model should not only behave as well as the original model, it should behave better than the original model. The evaluation of the enhancement approach is based on the following criteria:

- Performance of the agent in known and unknown situations
- Quality & reliability of the agents behavior

These are described below:

Performance of the agent in known and unknown situations: Performance experiments measure whether the agent achieves its mission goal in the environments provided and the duration it took the agent to achieve the mission goal. Performance in the context of this dissertation is measured using the elapsed time from start to finish of a mission. The actions taken by the agent in its environment determine whether the mission goals are achieved or not. A comparison is carried out between the base agent's performance, i.e. elapsed time for the base agent to achieve its mission goal versus the elapsed time of the enhanced agent to achieve the same mission goal. A comparison is also carried out on both agents to see whether the mission goal is achieved or not in known and unknown situations.

Quality & reliability of agents' behavior: Experiments that measure the quality and reliability of the agents' behavior in the simulated environment are carried out. As previously defined, the qualities of the agents' behavior are the attributes and characteristics of the actions taken at every state of the environment. The quality is measured by noting whether the correct behavior is exhibited at every state of the agents' environment. For example, does the agent come to a complete stop at a road intersection with stop signs? A comparison of the base agents' behavior to the enhanced agents' behavior is carried out when both agents attempt to achieve the mission goal.

Reliability of the agent's behavior is defined in terms of the change in the agents exhibited behavior at a given state during the execution[52] of the model in a simulator. For example, does the agent change its behavior at an intersection on a different simulation run for the same mission after learning (training) is complete? That is, after noting the agent's behavior at an intersection or a red traffic light during the first simulation cycle, is there a change in the agent's behavior at the same intersection or red light during the second simulation cycle under the same conditions? A note should be made that quality and reliability of the agents' behavior are measured after the agents' enhancement (learning). A comparison between the base agent and the enhanced agent is carried out to measure the long term reliability of behavior.

7.2 Experiments

This section outlines the experiments performed for the model enhancement technique. There are three environments in the automobile driving domain used in the experiments, two environments contain three routes and one environment contains four routes. The first environment is the training environment that contains shorter routes and will be described later in this chapter. The second environment is the execution environment used in the execution of the already trained agent (the enhanced agent). The second environment is used in evaluating the enhancement technique by comparing the enhanced agent with the base agent. The third environment which contains four routes, is used in the first experiment to show how using incorrect knowledge to make decisions leads to making wrong decisions as shown in the agent determining the fastest route to a destination. All three environments are described later in this chapter. In all

[52] The execution of the model is done after the agent is trained. More on this in the section with the detailed description of the experiments.

environments, the same environmental conditions exist on each route with differences on the defined maximum speed limits[53] on the road segments in the routes and the arrangement of the road segments in each route. The overall objective of the testing effort is to evaluate the enhancement technique based on the criteria listed in the previous section.

7.2.1 Description of Test Environment

The hardware and software required to run the model enhancement technique includes the following:

- CPU processor of 1000 MHz or greater
- Approximately 500 MB of available disk space
- Windows 95/98/2000/XP or LINUX operating system
- Oracle PL/SQL
- Oracle Database

The enhancement technique was tested for completeness with the experiments described in the next section.

7.2.2 Experiment Descriptions

There are five experiments carried out to test the performance and behavior of the enhanced agent. The goal and reasons for performing each experiment has been explained earlier in this chapter. Before the experiments are performed, the agent is trained to learn the contextual attributes of its environment, that is, the agent is enhanced.

[53] The defined maximum speed limit for a given road type is the same in all environments, for example, the maximum speed for 'CITY' road type is the same in all three environments. The differences in maximum speed limit is between road types.

7.2.3 Enhancing the Agent

As stated earlier, the first environment consisting of three shorter routes was used to train the agent. The reasons for using a different route to train the agent are twofold; 1) to be sure the enhanced agent can generalize its actions and behavior in similar situations and 2) because of the speed in which the agent can traverse the shorter routes, that is, using the shorter routes enabled faster training. The training routes consist of all road types available in the three routes used in the execution phase.

The training of the agent consists of training the agent to achieve the mission goals of all experiments performed in the automobile driving domain. This consisted of training the agent to learn the maximum speed attribute of the various road segments (context) when a context was defined for the road segment, while attempting to achieve its mission goal of arriving at its destination as fast as possible without violating any traffic laws. By learning the maximum speed attribute of each road segment, the enhanced agent is expected to outperform the base agent and also behave better at red traffic lights and intersections. On the other hand, if there are no contexts defined for the road segment, the enhanced agent is expected to learn new contexts that accurately represent the road segment. Tables 7.1, 7.2 & 7.3 show the definitions of the routes used to train the agent. Figures 7.1, 7.2 and 7.3 show the pictorial representation of the training routes. Note that the training routes shown in tables 7.1, 7.2 and 7.3 are different from the routes used in the execution phase of the experiments.

Table 7.1 Training Route A

ROUTE_ID	1	1	1	1	1
ROAD_ID	1	2	3	4	5
ROAD_NAME	FREEWAY	CITY	FREEWAY	FREEWAY	CITY
DESCRIPTION	FREEWAY driving	CITY driving	FREEWAY driving	FREEWAY driving	CITY driving
ROAD_LENGTH	1.5	0.6	1.2	1.5	1
ANGLE	5	85	45	26	2
ROAD_TYPE	FREEWAY	CITY	FREEWAY	FREEWAY	CITY
TRAFFIC	0	0	0	0	0
INTERSECTION	0	0	0	0	0
MAXSPEED	75	50	75	75	50

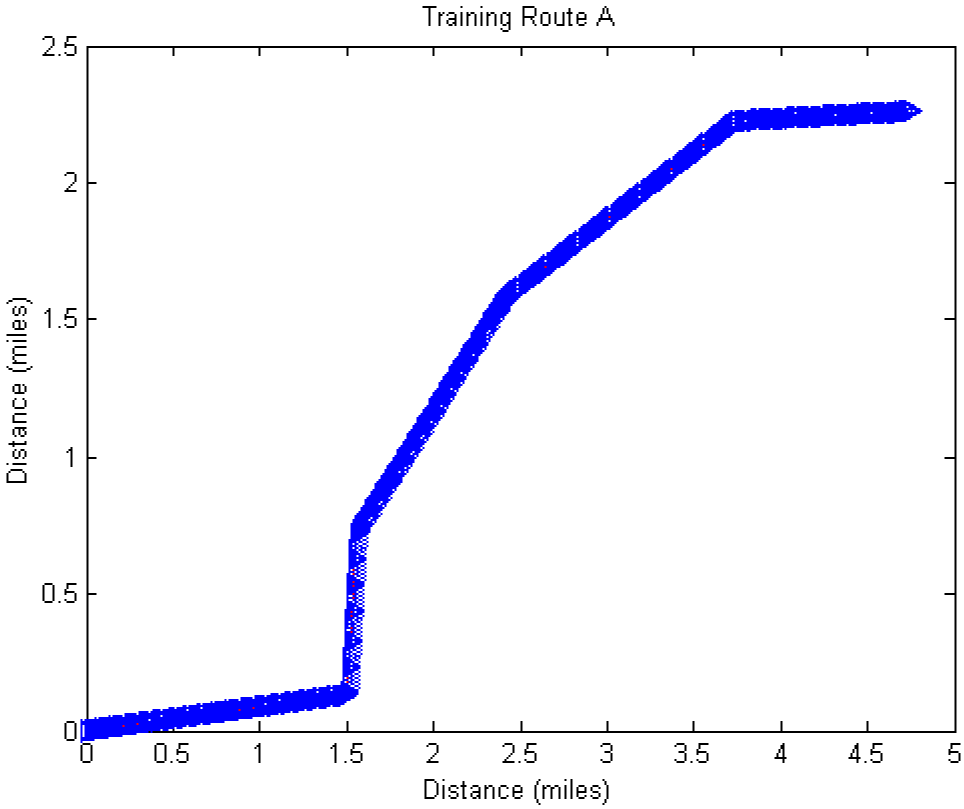


Figure 7.1 Pictorial representation of training route A.

Table 7.2 Training Route B

ROUTE_ID	2	2	2
ROAD_ID	1	2	3
ROAD_NAME	PARKING_LOT	CITY	RAMP
DESCRIPTION	PARKING_LOT driving	CITY driving	RAMP driving
ROAD_LENGTH	0.2	0.4	0.5
ANGLE	5	85	26
ROAD_TYPE	PARKING_LOT	CITY	RAMP
TRAFFIC	0	0	0
INTERSECTION	0	0	0
MAXSPEED	15	50	35

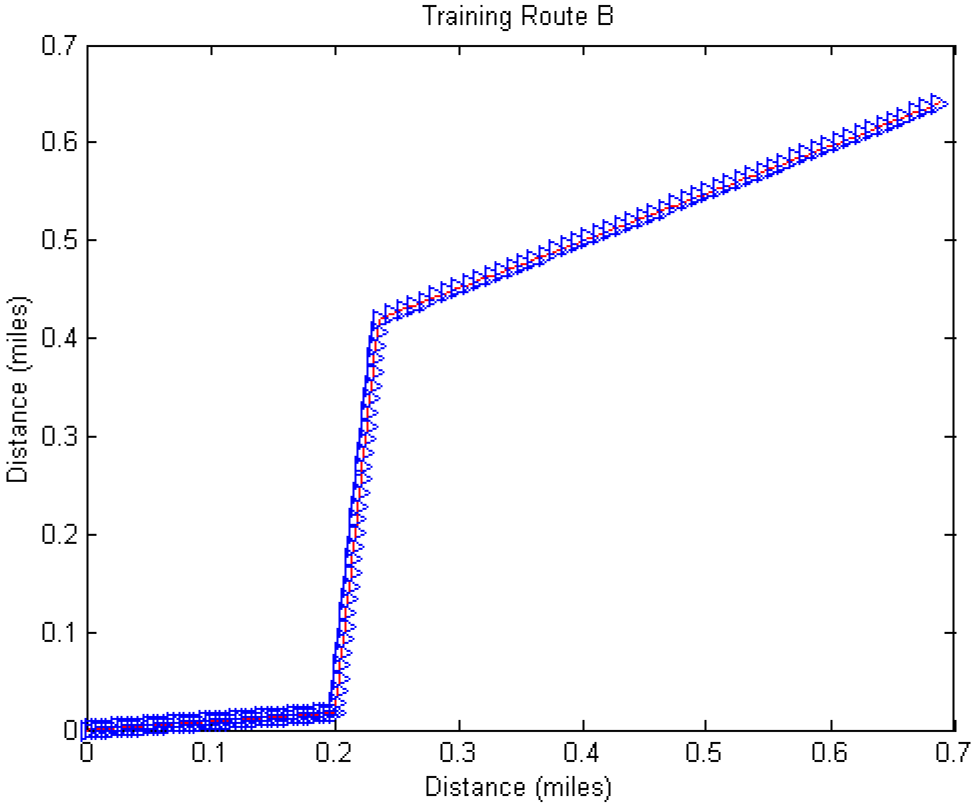


Figure 7.2 Pictorial representation of training route B.

Table 7.3 Training Route C

ROUTE_ID	3	3	3
ROAD_ID	1	2	3
ROAD_NAME	PARKING_LOT	CITY	DIRT
DESCRIPTION	PARKING_LOT driving	CITY driving	DIRT driving
ROAD_LENGTH	0.2	0.4	0.5
ANGLE	5	85	26
ROAD_TYPE	PARKING_LOT	CITY	DIRT
TRAFFIC	0	1	0
INTERSECTION	1	0	0
MAXSPEED	15	50	30

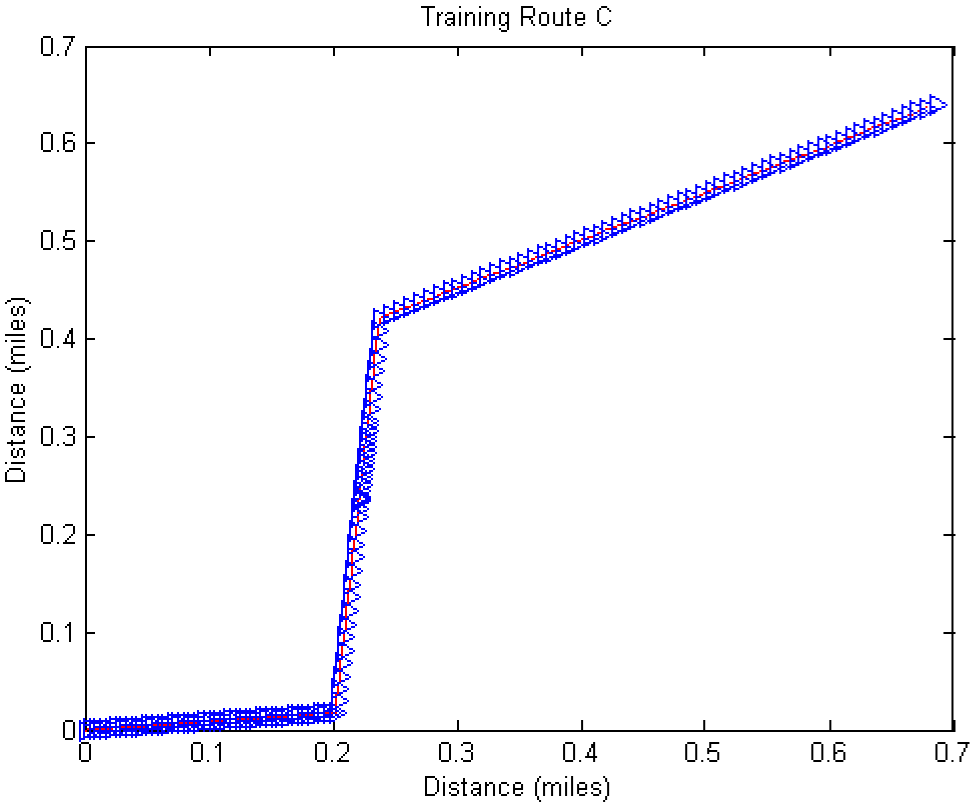


Figure 7.3 Pictorial representation of training route C.

Training route A was used in training the agent to learn the appropriate maximum speed attributes for the FREEWAY and CITY driving. Training route B was used in training the agent to learn the appropriate maximum speed attributes for the PARKING_LOT and RAMP driving. Training route C was used in training the agent to learn the appropriate maximum speed attributes for INTERSECTION and TRAFFIC_LIGHT driving as well as learning and creating the appropriate context with appropriate actions and attributes for DIRT driving.

Training the agent to learn the maximum speed attribute in the contexts commences as the simulation begins. The training algorithm described in the previous chapter is used.

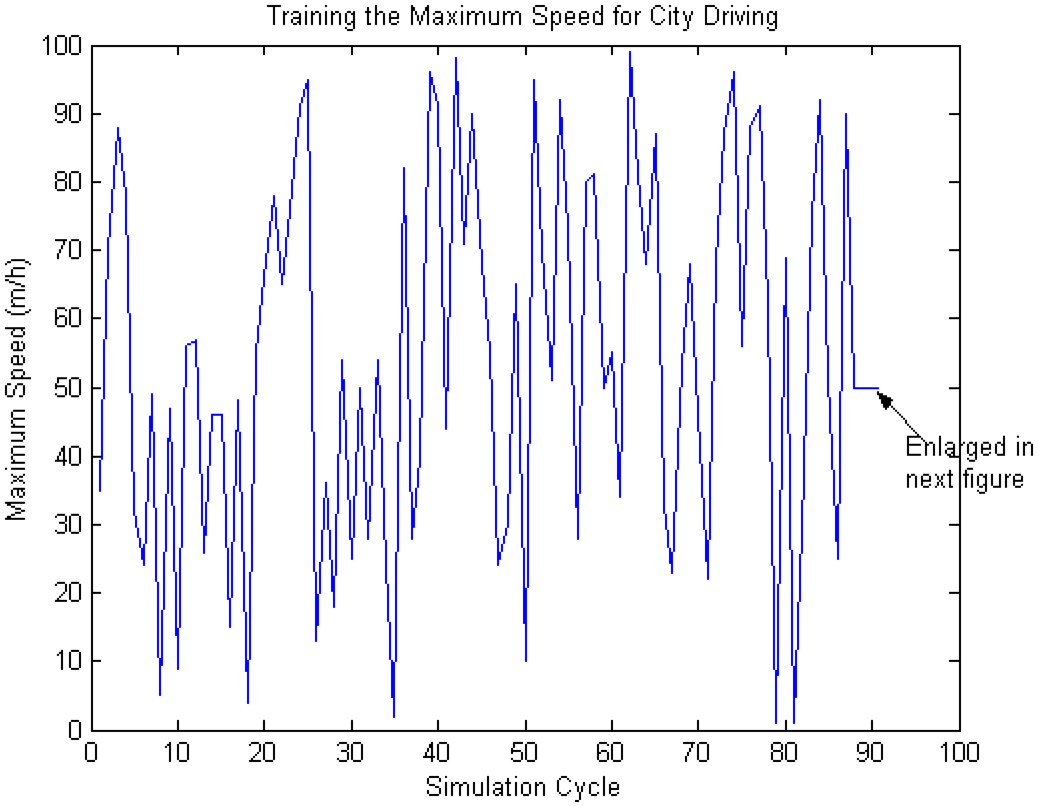


Figure 7.4 Training Maximum Speed Attribute for City Driving

Figure 7.4 shows the different maximum speed values used by the agent in all the simulation cycles. The figure shows how the training of maximum speed attribute progresses through all the simulation cycles. From figure 7.4 above, it is seen that the agent starts off the simulation with the maximum speed attribute for the CITY driving context defined as 35 m/h. The SME defined the maximum speed attribute as 35 m/h for the city driving context. After 91 simulation cycles, the agent learnt the appropriate maximum speed attribute for city driving context to be 50 m/h. The value of the maximum speed attribute for the city driving context during training and at each simulation cycle can be seen in figure 7.4. The fluctuation in maximum speed values for different simulation cycles is based on the maximum speed value being chosen at random using the learning strategy described in Chapter 6 (the random learning strategy). The maximum speed attribute eventually converges to the maximum speed value of the environment. The convergence occurs when there is no change in the maximum speed value between simulation cycles. Figure 7.5 is an enlarged version of portions of figure 7.4 showing when convergence occurs.

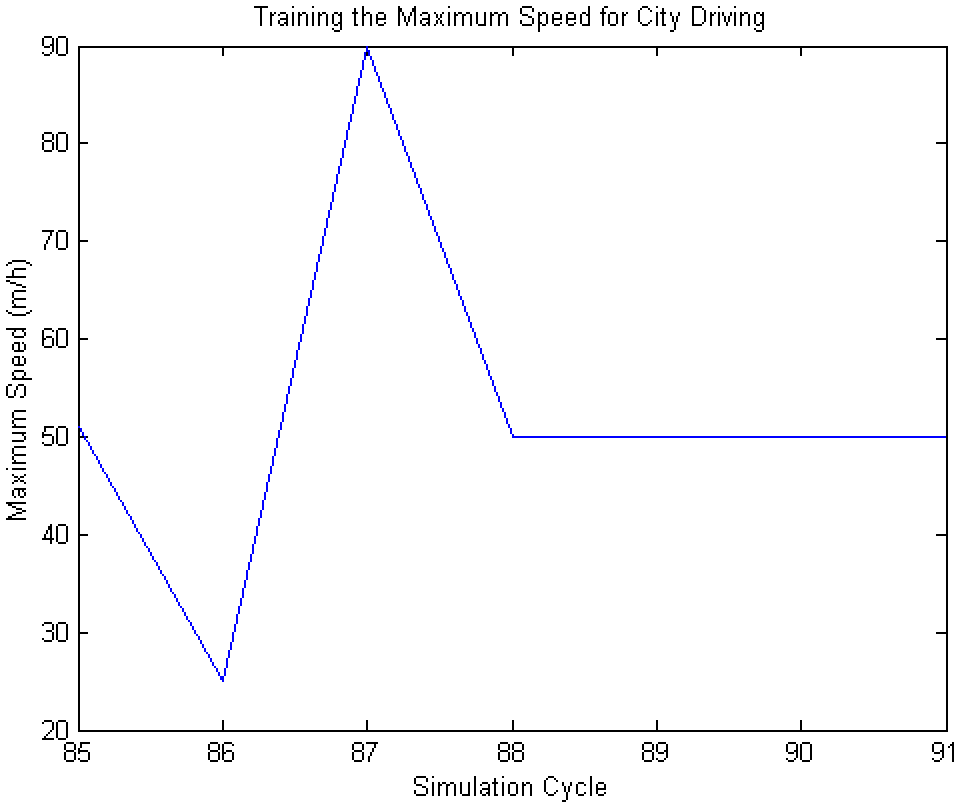


Figure 7.5 Training Maximum Speed Attribute for City Driving showing when Convergence Occurs (Enlarged Figure)

Figure 7.6 shows the reward received by the agent while learning the maximum speed attribute. It shows how the agent is rewarded in each simulation cycle for choosing the correct maximum speed value. The value of the reward assigned to the agent for being in a given state was discussed in chapter 6. It can be seen that the agent receives the highest reward of 50 points whenever the maximum speed of 50 m/h is chosen.

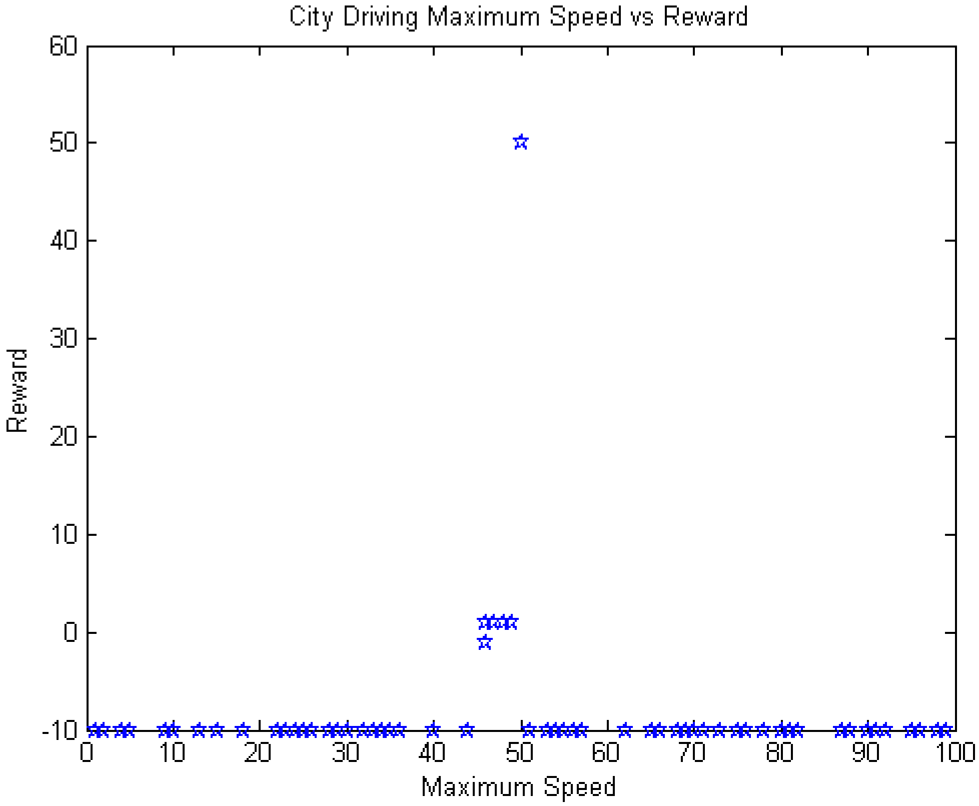


Figure 7.6 City Driving Maximum Speed vs Reward

Figure 7.7 shows the reward received by the agent in each simulation cycle. From the figure, it can be seen that the agent is punished in most simulation cycles. If figures 7.4, 7.6 and 7.7 are visually combined as a single figure, one can see that the agent is consistently punished for choosing the wrong maximum speed value, the agent eventually gets rewarded when it makes the correct choice.

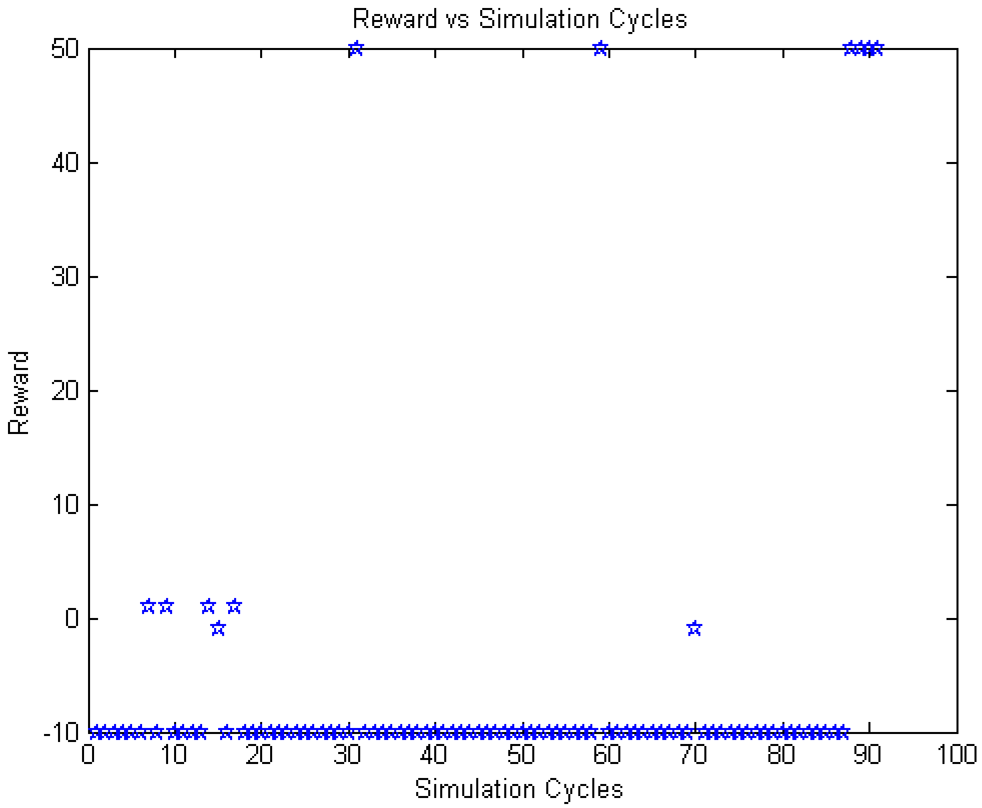


Figure 7.7 City Driving Rewards vs Simulation Cycles

In training the agent to learn the appropriate maximum speed attribute for the freeway driving context, the agent starts with the maximum speed value provided by the SME, i.e. 50m/h as shown in figure 7.8. A total of 71 simulation cycles were used in training the agent to learn the correct maximum speed value in the freeway context. It uses the learning strategy described in chapter 6, i.e., the agent randomly selects a maximum speed value during the first 20 simulation cycle, during the next 20 simulation cycles, i.e. from 21 to 40, it uses the value that provided the most reward or based on the reward the agent received in the previous simulation cycle, it directs the learning efforts of the agent. The agent randomly chooses the maximum speed value between 41 and 60 simulation cycles and from the 61st simulation cycle, the agent settles for the maximum speed value that produced the most reward in all 60 simulation cycles. As stated in Chapter 6, the choice of training the agent this way is to incorporate some form of direction in the agents learning. A random approach in training the agent will eventually converge as used in learning the city driving context. The choice of 0 to 20, 21 to 40, etc were chosen at random in even units, a choice of 0 to 10, 11 to 50, etc. could have been chosen as well.

The SME defined the maximum speed attribute as 50 miles/hr for the freeway driving context. After 71 simulation cycles, the agent learnt the appropriate maximum speed attribute for freeway driving context to be 75 miles/hr. As can be seen from figure 7.8, the maximum speed attribute converges to the correct value after 60 simulation cycles.

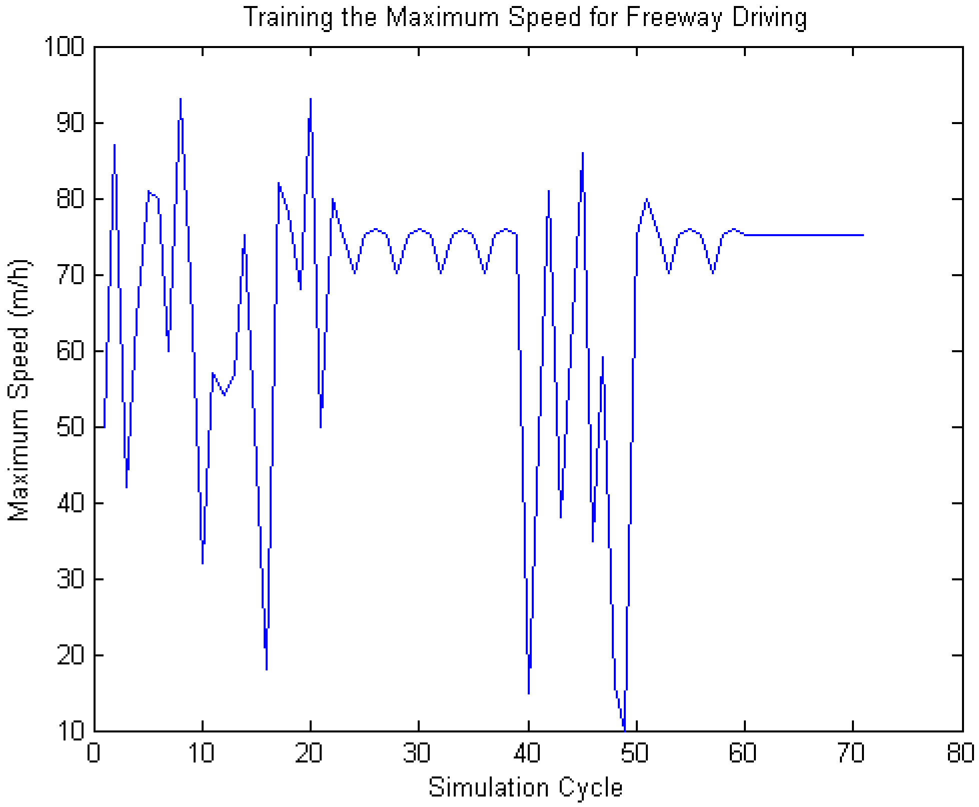


Figure 7.8 Training Maximum Speed Attribute for Freeway Driving

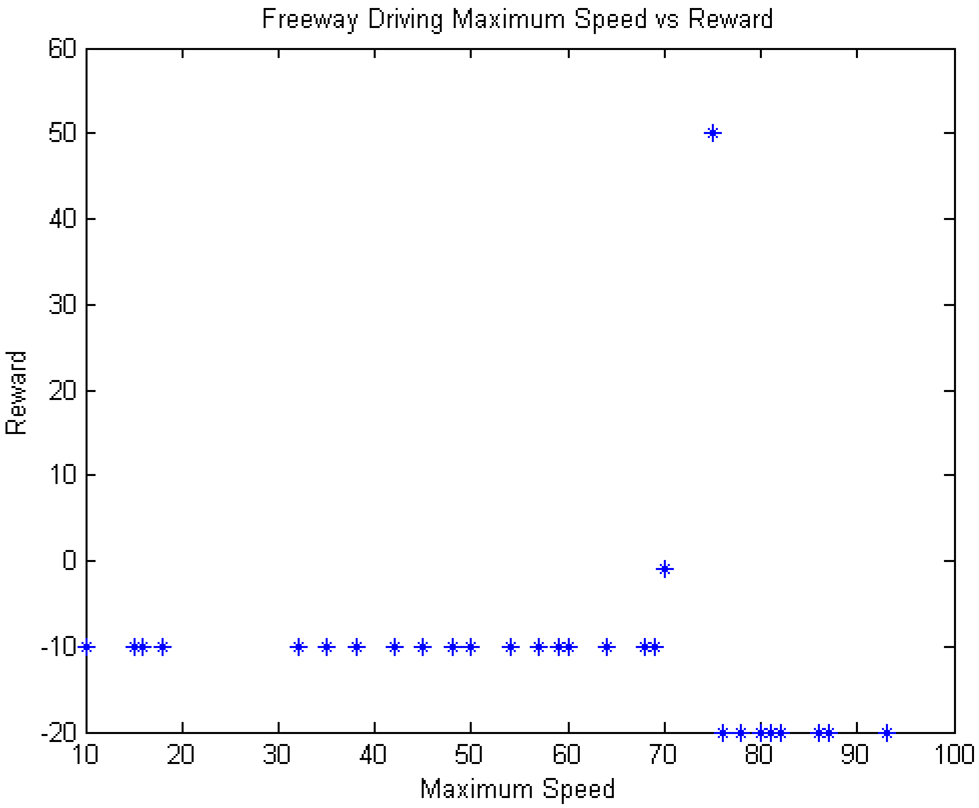


Figure 7.9 Freeway Driving Maximum Speed vs Reward

Figure 7.9 shows the reward received by the agent at a given maximum speed. From the figure, it can be seen that the agent received the maximum reward when the maximum speed value was 75 m/h. In figure 7.10, we can see the rewards received by the agent in each simulation cycle. The same analysis and description carried out on the city driving figures apply to the freeway driving figures too.

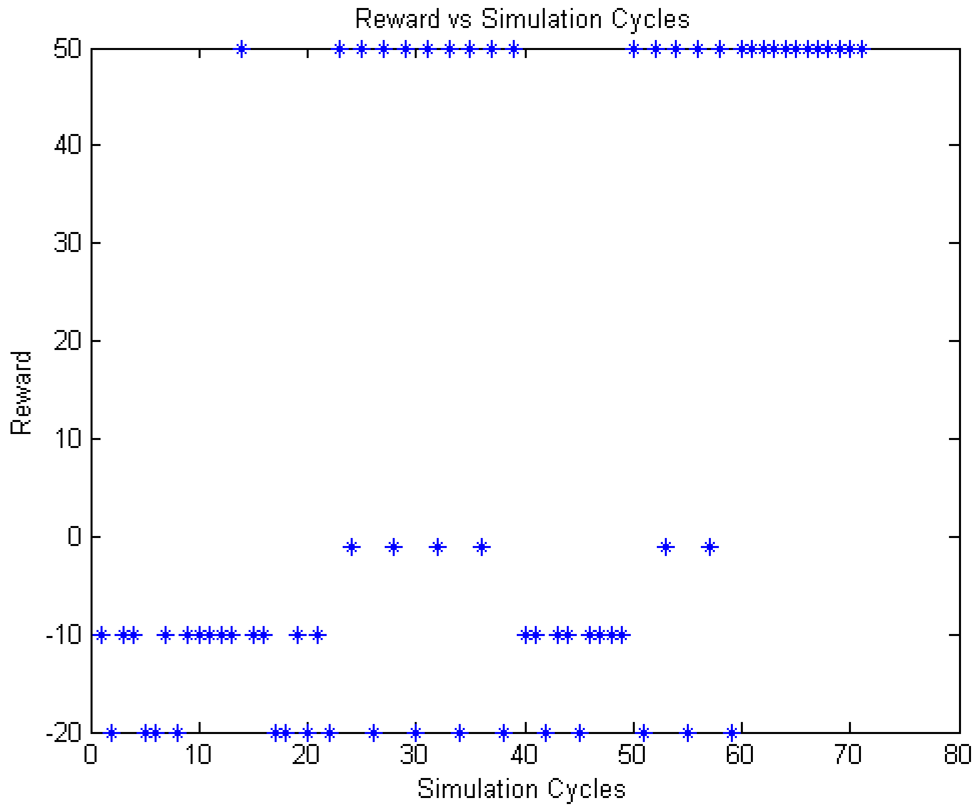


Figure 7.10 Freeway Driving Rewards vs Simulation Cycles

Figures showing the training outcomes for the other contexts – parking_lot driving, dirt driving, intersection driving, traffic_light driving, ramp driving and submarine target track are shown in Appendix A. The explanations provided for the training figures for city and freeway driving also applies to the other contexts.

7.3 Experiment Descriptions and Results

Experiment 1.0

One of the environments is used in this experiment. Four routes are traversed and the mission goal is for the agent to make an intelligent decision on the tactically optimal route to its destination when being controlled by the base contexts or the enhanced contexts.

The objective of this experiment is to show the dangers of using incorrect knowledge in decision making. The base and enhanced agents have a mission goal of finding the tactically optimal route to their destination, although a trivial mission goal that can be achieved using simple search algorithms, the objective of the experiment emphasizes the overall importance of the new technique in decision making. The destination on each route is different. The mission goal of both agents can be translated to finding the tactically optimal route to an emergency hospital[54]. The base agent uses the knowledge provided by the SME in determining the fastest route, whereas the enhanced agent uses the knowledge learnt during its training to find the fastest route.

The environment, the agents reward table, the global and local fact bases are all initialized. The experiment is performed on each agent at different times with the environmental conditions remaining the same. Upon starting the simulation, the agent randomly chooses a route to traverse. In this experiment, the agents are trained to learn the tactically optimal route. Recall that the base agent was previously trained to learn the actual maximum speed attribute on each route, thus making it an enhanced agent. With the knowledge of the actual maximum speed on any given route in its environment, the enhanced agent traverses the routes noting the time it takes to get to the destination (end of the route). This simulation cycle is repeated multiple times for both the base and enhanced agent. The elapsed times to their destinations are noted. The learning strategy defined in the previous chapter is used, i.e. both agents initially chooses a route randomly and then based on the rewards they receive, they adjust their route choices in subsequent simulation cycles. When the agents choice of route converges, i.e. a given route

[54] Note that there can be different emergency hospitals in a vicinity and finding the fastest route to one of them can help save a life. Also note that both agent drivers are akin to regular citizens that are expected to obey all traffic laws irrespective of the emergency situation and are different from a fire truck or other official cars that may not obey traffic laws in emergency situations.

consistently produces the most reward when chosen as the tactically optimal route, a decision is made. A comparison is then carried out on the fastest route chosen by both agents.

The definition of all routes traversed in this experiment is presented in Table 7.4 and figure 7.11

Table 7.4 Definition of Routes 1 through 4

ROUTE_ID	1	1	1	2	2
ROAD_ID	1	2	3	1	2
ROAD_NAME	PARKING_LOT	CITY	DIRT	CITY	FREEWAY
DESCRIPTION	PARKING_LOT	CITY DESC	DIRT DESC	CITY DESC	FREEWAY DRIVING
ROAD_LENGTH	0.2	0.4	0.5	0.35	0.75
ANGLE	5	85	26	30	15
ROAD_TYPE	PARKING_LOT	CITY	DIRT	CITY	FREEWAY
TRAFFIC	0	1	0	1	0
INTERSECTION	1	0	0	1	0
MAXSPEED	15	50	30	50	75

ROUTE_ID	3	3	4	4	
ROAD_ID	1	2	1	2	
ROAD_NAME	CITY	FREEWAY	FREEWAY	CITY	
DESCRIPTION	CITY DESC	FREEWAY DESC	FREEWAY	CITY DESC	
ROAD_LENGTH	0.7	0.1	1	0.8	
ANGLE	15	35	5	85	
ROAD_TYPE	CITY	FREEWAY	FREEWAY	CITY	
TRAFFIC	1	0	0	0	
INTERSECTION	0	0	0	0	
MAXSPEED	50	75	75	50	

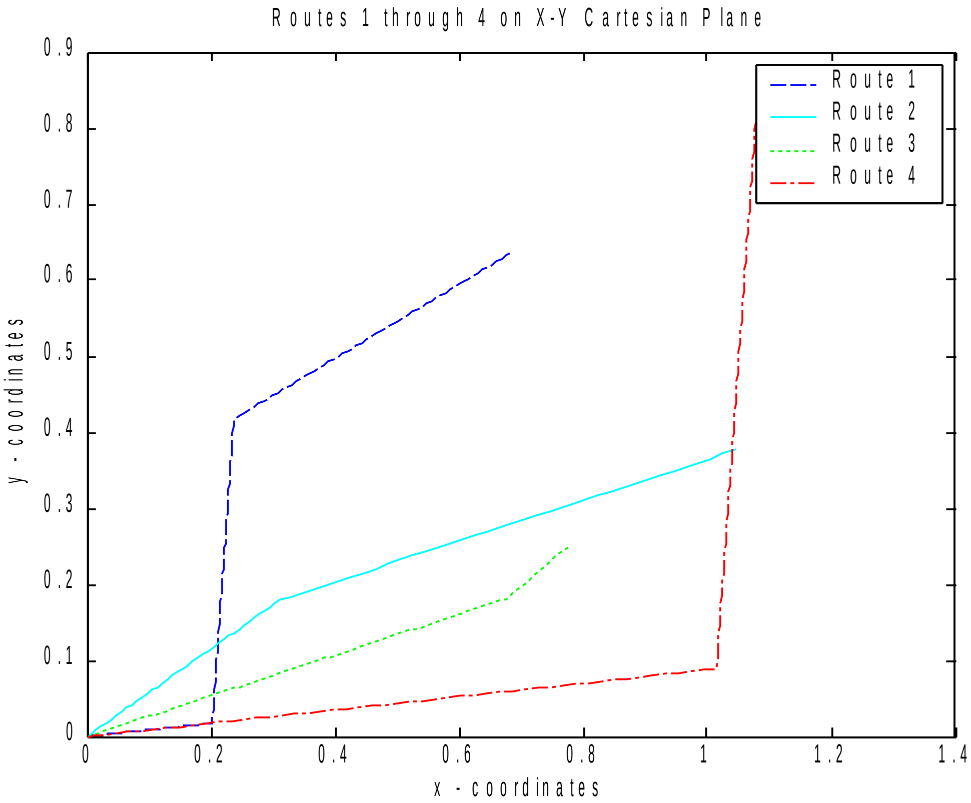


Figure 7.11 Pictorial Representations of Routes 1 through 4

In this experiment, the agent randomly selected a route and traversed that route until it arrived at its destination. Upon arrival, the time it took the agent to arrive at its destination is recorded in the reward table and a reward assigned to the agent based on the current elapsed time and the previous elapsed time. If the agent elapsed time of the agent on the current route is more than the elapsed time on the previous route, the agent is punished for taking the current route, on the other hand if the elapsed time on the current route is less than that of the previous route, the agent is rewarded and the route is noted. If there is no change in elapsed time between the previous and current routes, the agent is neither punished nor rewarded. A snippet of the reward procedure is shown in figure 7.12

```
PROCEDURE "REWARD"
  ( run_id IN pls_integer, TRAINING_MODE IN VARCHAR2, curr_rte_id IN pls_integer)
  IS
  rwd_cnt           pls_integer;
  prev_sessionid    pls_integer;
  prev_time         float := 0.0;
  ctx_reward        pls_integer;
  curr_time         float := 0.0;
  prev_rte_id       pls_integer;

BEGIN

Get the maximum session id in the reward table

   select max(session_id) into prev_sessionid
   from rwd
   where description = TRAINING_MODE;

--         Get the run_time and route id from reward table for the previous session

   select run_time, route_id into prev_time, prev_rte_id
   from rwd where session_id = (select max(session_id)
                  from rwd
                  where description = TRAINING_MODE);

 --        Get the total number of records in the reward table for the current training mode

  select count(*) into rwd_cnt
   from rwd a
   where a.description = TRAINING_MODE;

   --      Get the total elapsed time it took for the agent to arrive at its destination for the current
simulation run

   SELECT SUM(G.ELAPSED_TIME) INTO curr_time
   FROM GFB G WHERE G.SESSION_ID = RUN_ID;

   if rwd_cnt < 20 then   -- if the total number of simulation runs is less than 20,
      if prev_rte_id = curr_rte_id then   -- if the previous route id is the same as the current route
id randomly chosen
        ctx_reward := 0;  -- set the reward to 0
      else
         if prev_time > curr_time then   -- if the previous elapsed time is greater than the current
elapsed time
           ctx_reward := 10;   -- set the reward for using this route to 10
         else
           ctx_reward := -10;  -- else set the reward for using this route to - 10
         end if;
       end if;
   else
       if prev_rte_id = curr_rte_id then
         ctx_reward := 1;
       else
         if prev_time > curr_time then
           ctx_reward := 10;
         else
```

```
           ctx_reward := -10;
         end if;
       end if;
    end if;


   insert into rwd ( session_id, description, run_time,  route_id, reward)
      values (run_id, TRAINING_MODE, curr_time, curr_rte_id,ctx_reward);

   COMMIT;
EXCEPTION
   WHEN OTHERS THEN
   ----DBMS_OUTPUT.PUT_LINE(dbms_utility.format_error_backtrace);
   SIMULATION_RUNS( Run_ID, 'REWARD', 'Fail','Error=>'||
substr(dbms_utility.format_error_backtrace,1,200)  );
   RAISE;

END;
```

Figure 7.12 Procedure for rewarding the agent for choosing a faster route

Table 7.5 shows a snippet of the rewards table

Table 7.5 Snippet of Reward Table

SESSION_ID	DESCRIPTION	RUN_TIME	ROUTE_ID	REWARD
853	TIME	75.35	2	10
854	TIME	74.95	2	0
855	TIME	75.09	2	0
856	TIME	176.24	1	-10
857	TIME	177.75	1	0
858	TIME	74.66	2	10
859	TIME	79.71	3	-10
860	TIME	79.67	3	0
861	TIME	80.66	3	0
862	TIME	79.81	3	0

Table 7.6a Sum of Rewards when using the Original and Enhanced Contexts

ROUTE_ID	**SUM of REWARDs using Original Contexts**	**SUM of REWARDs using Enhanced Contexts**
1	-68	-60
2	185	16
3	-9	173
4	-59	-100

Table 7.6a shows how the agent decides what route is the tactically optimal route. Figure 7.13 is a pictorial representation of Table 7.6a. Both agents make their decisions based on the sum of the rewards they receive during the simulation. As can be seen from Table 7.6a and figure 7.13, the sum of the rewards for the base and enhanced agents differ on all routes. The base agent was consistently rewarded during the simulation for choosing route 2 because the elapsed time to the destination in route 2 was the smallest. At the end of the simulation the agent received a total of 185 points when it chose route 2, hence the base agent made the decision that route 2 was the tactical optimal route. On the other hand, the enhanced agent chose route 3 as the tactical optimal route.

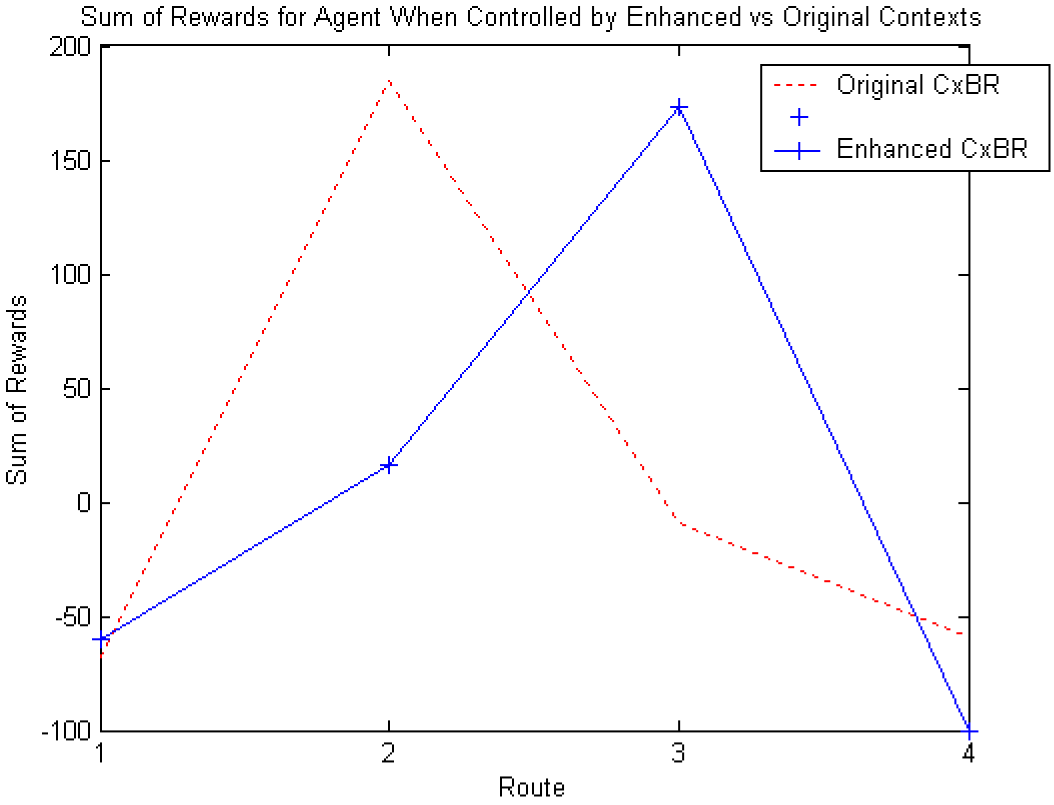


Figure 7.13 Sum of Rewards

Table 7.6b Average Run Time on Each Route

Route ID	Average Run Time (secs) using Original Contexts	Average Run Time (secs) using Enhanced Contexts
1	164.35	145.98
2	75.3	63.41
3	80.31	57.35
4	130.6	106.72

In the route selection experiment, the agent chooses route 2 as the tactically optimal route to its destination when it is controlled by the base CxBR as shown in figure 7.13. On the other hand when the agent is controlled by the enhanced CxBR, the agent chooses route 3 as being the tactically optimal route to its destination. This experiment shows that incomplete knowledge or misrepresentations about a situation in a context could lead to the agent making wrong choices. A note should be made that in this experiment, the maximum speed value for the same contexts in the enhanced and original CxBR model had the same values, with the exception of the CITY driving context. The maximum speed value for the CITY driving context of the original CxBR model was reverted back to 35 m/h whereas the enhanced model remained at 50 miles per hour. Both agents made intelligent decisions on the tactically optimal route to their destination based on the information available in their contexts.

Table 7.6b shows the average run times of the agent on each route when controlled by the original and enhanced contexts. From the average run times, it can be seen that the tactically optimal route is actually route 3 as determined by the enhanced CxBR agent. The base agent made an incorrect choice. In reality the tactically optimal route is route 3.

Experiment 2.0

The objective of this experiment is to show the effects of using incorrect or wrong information on an agent's performance in a given task. A comparison of the performance of the enhanced CxBR agent to that of the base CxBR agent is carried out. The performance is compared in terms of the elapsed time to arrive at their destinations while traversing the same routes. The shorter the elapsed time to the agent's destination, the better the agent's performance. Both agents traverse the three routes separately. The agents traverse each route five times, traversing each route five times was chosen to present enough data for analyzing the results[55], the agents could have also traversed each route 7 times or 28 times, etc. with no significant difference in the final results. The agents encounter different road segments in each route as described earlier in the chapter. The time it takes the agent to arrive at the destination for each simulation run before and after the agent's enhancement is presented below. Note that the agent had previously been trained. The actual comparison in all experiments is between the already enhanced agent and the base agent.

[55] There was no difference in the overall result when the results of 1 or 2 or 3 simulation cycles were used

Table 7.7 Elapsed Time of Original CxBR agent and the Enhanced CxBR agent.

ROUTE_ID	ELAPSED_TIME (Original CxBR)[56]	ELAPSED_TIME (Enhanced CxBR)
1	1197.44	940.23
1	1197.92	938.83
1	1198.45	938.8
1	1198.17	939.09
1	1197.33	938.33
2	1040.25	790.53
2	1036.34	788.83
2	1033.3	792.2
2	1033.33	792.98
2	1034.42	788.65
3	1270.17	762.64
3	1273.1	766.58
3	1006.95	767.09
3	1011.79	765.47
3	1006.51	763.44

[56] For experiment 1, the behavior of the original CxBR was modified to observe all environmental / simulation constraints such as stopping at intersections, stopping at red traffic light, etc. This was done to present a uniform test bed for both the original CxBR agent and its enhanced counterpart.

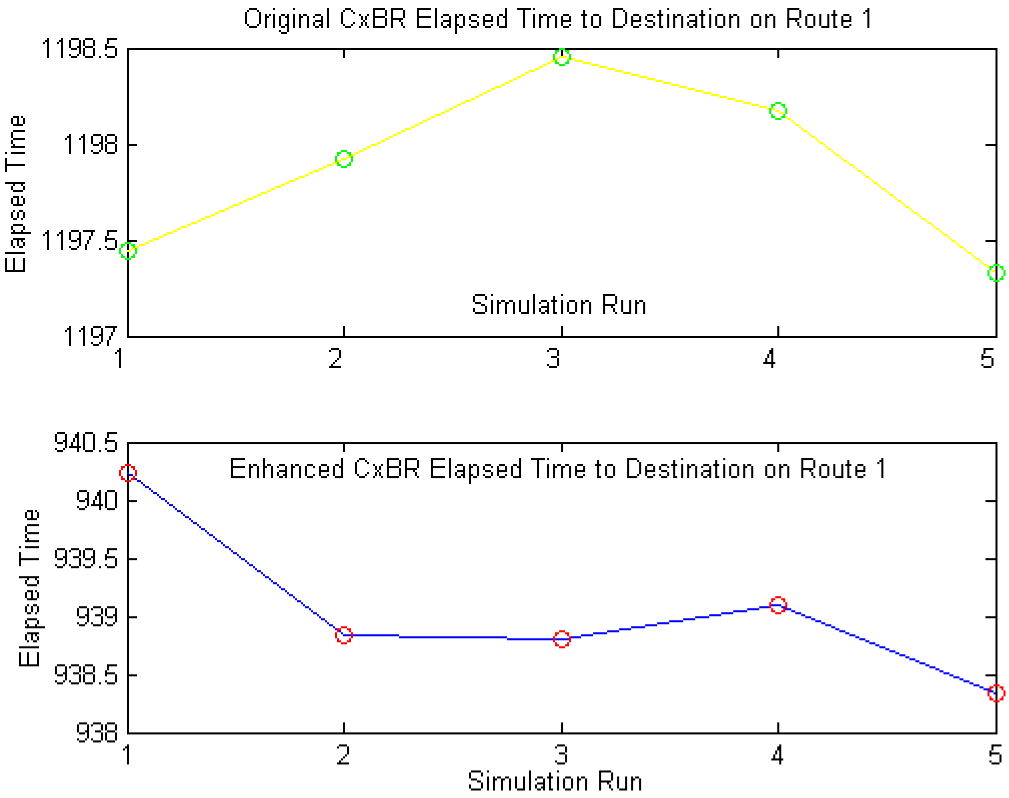


Figure 7.14 Elapsed Time to Destination on Route 1

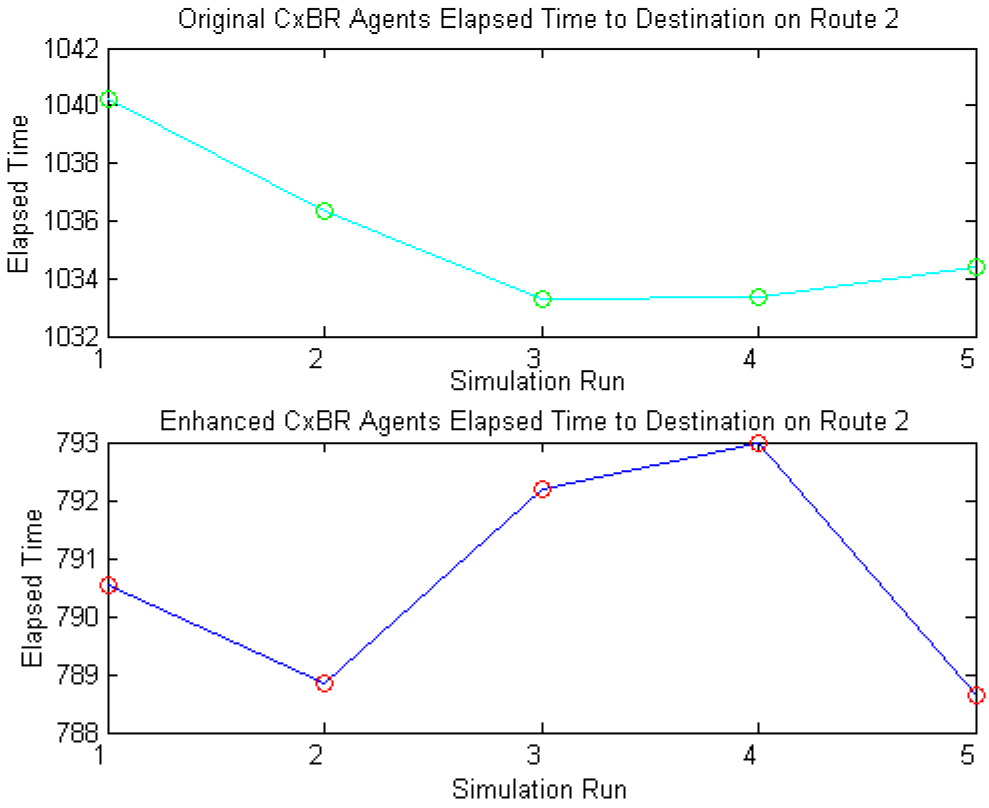


Figure 7.15 Elapsed Time to Destination on Route 2

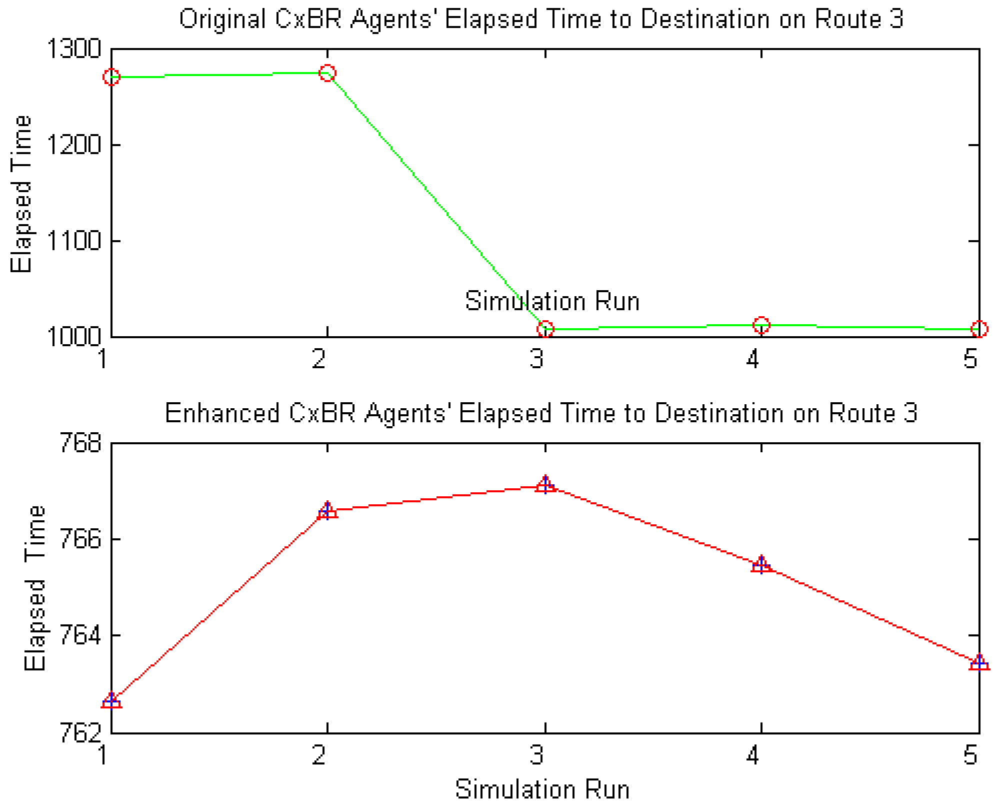


Figure 7.16 Elapsed Time to Destination on Route 3

Figures 7.14 to 7.16 show the elapsed time to destinations of the base CxBR agent and its enhanced counterpart on all three routes for the five simulation runs. The fluctuations in elapsed time on each route for each agent is due to the various events in the environment, for example, the traffic light color being different on each simulation run, hence the agent might stop at a red light on one simulation run and maintain its current speed past a green light on another simulation run.

Recall the hypotheses stated in Chapter 3, (3.2):

> *Reinforcement learning can be used to automatically and efficiently enhance a tactical agent's behaviour from the experience gained by the interaction of the agent with its environment. Additionally, based on the mission goals, these agents*

will perform better than the agents developed from knowledge acquired from experts.

The enhanced agent was enhanced using reinforcement learning, proving this hypothesis. We show this quantitatively as follows:

From table 7.4, according to Mason, et. al. [199] the difference d is:

d = Elapsed Time for Enhanced Agent (x) – Elapsed Time for Original Agent (y)

The null hypothesis is the original CxBR agent will perform as well as the enhanced CxBR agent at the minimum.

$$H_0 : \mu_d \geq 0$$
$$H_a : \mu_d < 0$$

Where μ_d is the mean of the differences between the enhanced CxBR agent and the original CxBR agent, H_0 is the null hypothesis and H_a is the alternate hypothesis.

Table 7.8 Differences in Elapsed Time

ROUTE_ID	[E]LAPSED_TIME (Original CxBR) (y)	ELAPSED_TIME (Enhanced CxBR)(x)		*d = x-y*	*d*^{2}
1	1197.44	940.23		-257.21	66156.9841
1	1197.92	938.83		-259.09	67127.6281
1	1198.45	938.8		-259.65	67418.1225
1	1198.17	939.09		-259.08	67122.4464
1	1197.33	938.33		-259	67081
2	1040.25	790.53		-249.72	62360.0784
2	1036.34	788.83		-247.51	61261.2001
2	1033.3	792.2		-241.1	58129.21
2	1033.33	792.98		-240.35	57768.1225
2	1034.42	788.65		-245.77	60402.8929
3	1270.17	762.64		-507.53	257586.7009
3	1273.1	766.58		-506.52	256562.5104
3	1006.95	767.09		-239.86	57532.8196
3	1011.79	765.47		-246.32	60673.5424
3	1006.51	763.44		-243.07	59083.0249
			Sum	***-4261.78***	***1326266.283***

To get the average difference in elapsed time we use the formula below [199], where *n* is the total number of simulation runs.

$$\bar{d} = \frac{\sum d}{n} = \frac{-4261.78}{15} = -284.1186667$$

Therefore, the average reduction in elapsed time is *284.1186667seconds*

The standard deviation [199] is :

$$s_d = \sqrt{\frac{\sum d^2 - \frac{\left(\sum d\right)^2}{n}}{n-1}} = \sqrt{\frac{1326266.283 - \frac{(-4261.78)^2}{15}}{15-1}} = 90.79609493$$

Using the t-test,

$$t = \frac{\bar{d}}{s_d / \sqrt{n}} = \frac{-284.1186667}{90.79609493 / \sqrt{15}} = -12.11931929$$

[E]

The results of the t-test above give a probability (p-value) of 0.0, thus there is a 0% probability that the null hypothesis is rejected in error. The null hypothesis is rejected, since the mean of the differences is less than 0 and the results show a 100% confidence that the enhanced CxBR agent will out-perform the base CxBR agent when there is incomplete and/or incorrect knowledge acquired from a subject matter expert.

An argument can be made against the validity of the results and conclusion of experiment 2. The argument will be that in this experiment, the enhanced agent learnt a maximum speed for each route that is higher than what the SME provided, and as such the enhanced agent is expected to move faster and thus have a shorter elapsed time on each route. For example, the maximum speed value for city driving provided by the SME is 35m/h whereas the agent learnt the correct value was 50m/h; Driving at 50m/h rather than 35m/h will definitely provide a shorter elapsed time on any route. The question is what will be the impact in the performance of the agent, if the SME had provided 50m/h and the agent learnt the actual speed was 35m/h? The argument can be made that the base agent driving at 50m/h will have a shorter elapsed time to destination than the enhanced agent driving at 35m/h. This argument is incorrect as the base agent moving at a faster speed will be stopped multiple times by the police and thus there will be delays and punishments for the base agent. These delays will lead to a larger elapsed time for the base agent.

Experiment 3.0

The objective of this experiment is to compare the behavior of the enhanced CxBR agent and that of the base CxBR agent. This experiment shows the effects of the SME providing an incorrect process or omitting a process on a given task. In this experiment, the process omitted by the SME is the process of stopping at a red traffic light or at an intersection with a stop sign and also the process of decelerating when the traffic light color is yellow and the agent is approaching the traffic light; in other words, the SME omits some of the core behavioral attributes of a typical car driver. Recall that there is a base agent and an enhanced agent. The behavior of both agents is compared in terms of the agents' actions and speed at intersections and traffic lights while both agents traverse the same routes. Both agents traverse three routes separately. The agents traverse each route five times as in experiment 2. The agents encounter different road segments in each route as described earlier. The driving speed of both agents at intersections, red and green traffic lights are presented.

Table 7.9 and figure 7.17 show a typical pattern of the enhanced and base agents' speed when approaching a traffic light. Table 7.9 and figure 7.17 contain information for only one traffic light. It can be seen that the enhanced CxBR agent starts to reduce its speed when the traffic light color changes from green to yellow and subsequently to red. On the other hand, the base CxBR agents' speed remained constant; it was utilizing the speed defined for the major context, CITY driving, even though it was being controlled by the TRAFFIC_LIGHT context. The pattern in table 7.9 and figure 7.17 was consistent in all simulation runs to test the behavior of the base and enhanced agents in all traffic lights.

Table 7.9 The Pattern of Enhanced CxBR Speed vs Original CxBR Speed

AgentsPosition	**LIGHT_COLOR**	**Enhanced CxBR Speed**	**Original CxBR Speed**
Before Traffic Light	GREEN	49.00	34.00
	GREEN	49.00	34.00
	YELLOW	29.00	34.00
	YELLOW	9.00	34.00
	YELLOW	4.00	34.00
	YELLOW	3.00	34.00
	RED	2.00	34.00
	RED	1.00	34.00
At Traffic Light	RED	0	34.00
After Traffic Light	GREEN	20.00	34.00

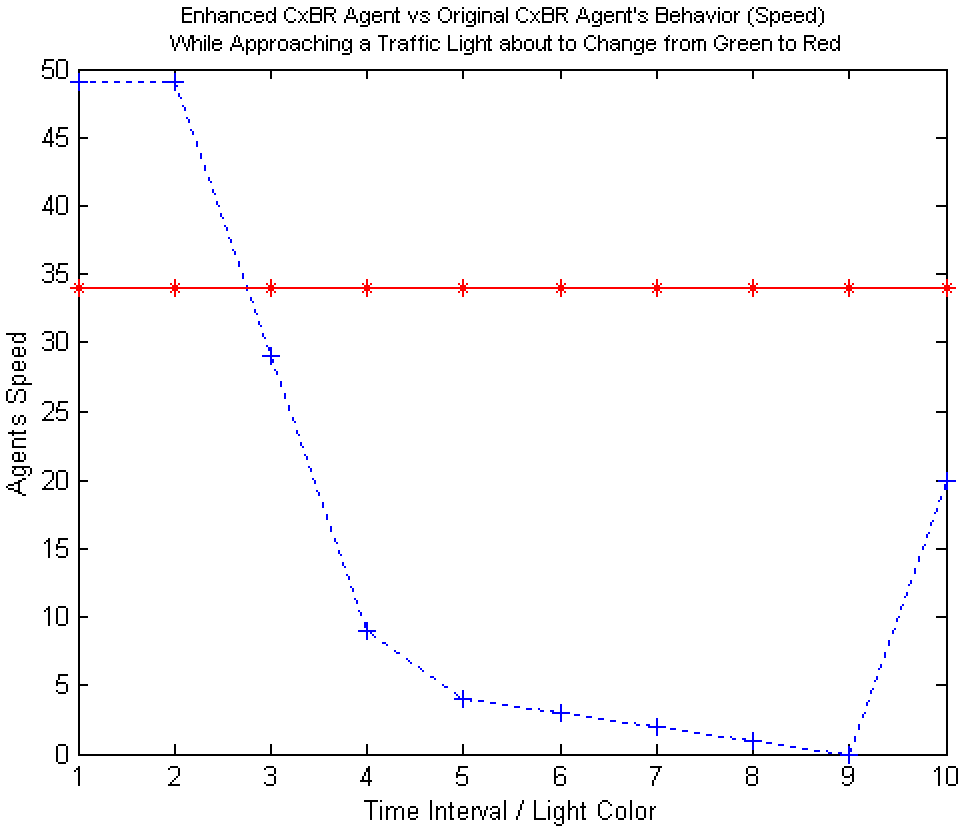


Figure 7.17 Enhanced CxBR agent vs Base CxBR agent Speed at a Traffic Light

In analyzing and computing the differences in behavior between the enhanced CxBR agent and the base CxBR agent, a snapshot of the agents' speed at the traffic light and at intersections was carried out. This is shown in Tables 7.10 and 7.11

In Table 7.10, the route_id shows the route number, intersection_position is the position where the intersection occurs on the route. Five records are shown, each representing a run through the simulation for route_ids 2 and 3. Note that there are no intersections on route 1.

Table 7.10 Agents' Speed at Intersection

ROUTE_ID	INTERSECTION POSITION	SPEED at INTERSECTION (Base Agent)	SPEED at INTERSECTION (Enhanced Agent)
2	2.5	34.00	0
2	2.5	34.00	0
2	2.5	34.00	0
2	2.5	34.00	0
2	2.5	34.00	0
3	0.2	10	0
3	0.2	10	0
3	0.2	10	0
3	0.2	10	0
3	0.2	10	0

In Table 7.11, note that there are no traffic lights on route 1 and there are two traffic lights on route 2, hence the denotation 2A and 2B to differentiate between both traffic lights on route 2. T.L Position denotes the position the traffic light on the route and T.L. Color denotes the color of the traffic light when both agents pass it. The behavior of the agent at the traffic light for each simulation run can be seen from table 7.11 by the speed at which it passes the traffic light based on the traffic light color, for example, on the first simulation run, 2A, when the traffic light color was yellow, the base agent went through at a speed of 34 m/h, its maximum speed, whereas the enhanced agent went through the same traffic light at a speed of 29 m/h because it had already started applying the brakes. On the second run through the first traffic light of route 2, i.e. 2A, the traffic light color is green and both agents pass through the traffic light at their maximum speed. On the third run, the traffic light color is red; the base agent cruises pass the light at its maximum speed whereas the enhanced agent stopped at the traffic light.

Table 7.11 Agents' Speed at Traffic Light

ROUTE_ID[57]	T.L POSITION	T.L COLOR	SPEED at T.L. (Base Agent)	SPEED at T.L. (Enhanced Agent)
2A	1.5	YELLOW	34.00	29.00
2A	1.5	GREEN	34.00	49.00
2A	1.5	RED	34.00	0
2A	1.5	RED	34.00	0
2A	1.5	RED	34.00	0
2B	4	GREEN	34.00	49.00
2B	4	GREEN	34.00	49.00
2B	4	RED	34.00	0
2B	4	RED	34.00	0
2B	4	RED	34.00	0
3	0.9	YELLOW	34.00	4.00
3	0.9	RED	34.00	0
3	0.9	RED	34.00	0
3	0.9	RED	34.00	0
3	0.9	GREEN	34.00	49.00

Recalling the hypothesis and steps utilized in analyzing the performance from experiment 1, it is hypothesized that the enhanced agent will behave better than the original CxBR agent. The measurement of behavior in this dissertation is restricted to the speed both agents exhibit at and near intersections and traffic lights.

From tables 7.7 and 7.8, according to Mason, et. al. [199] the difference d[58] is:

d = Speed for Enhanced Agent (x) – speed for Original Agent (y)

[57] In route 2, there are two traffic light positions, hence the connotation 2A & 2B

[58] d is the difference in speed between both agents at the traffic_light or intersection

The null hypothesis is the base CxBR agent will behave as well as the enhanced CxBR agent at the minimum.

$$H_0 : \mu_d \geq 0$$
$$H_a : \mu_d < 0$$

Where μ_d is the mean of the differences between the enhanced CxBR agent and the base CxBR agent, H_0 is the null hypothesis and H_a is the alternate hypothesis.

Table 7.12 Differences in Speed at Intersections for all Routes

ROUTE_ID	**INTERSECTION POSITION**	**SPEED m/h at INTERSECTION (Base Agent)(y)**	**SPEED m/h at INTERSECTION (Enhanced Agent)(x)**		**d = x - y**	**d^2**
2	2.5	34.00	0		-34.00	1156.0
2	2.5	34.00	0		-34.00	1156.0
2	2.5	34.00	0		-34.00	1156.0
2	2.5	34.00	0		-34.00	1156.0
2	2.5	34.00	0		-34.00	1156.0
3	0.2	10	0		-10	100
3	0.2	10	0		-10	100
3	0.2	10	0		-10	100
3	0.2	10	0		-10	100
3	0.2	10	0		-10	100
				sum	***-220.00***	***6280.00***

To get the average difference in the agents' speed at the intersection we use the formula below [199], where *n* is the total number of simulation runs.

$$\bar{d} = \frac{\sum d}{n} = \frac{-220.003}{10} = \underline{\underline{-22.0003}}$$

Therefore, the average difference in speed between the enhanced and base agents at intersections is 22.0003

The standard deviation [199] is :

$$s_d = \sqrt{\frac{\sum d^2 - \frac{\left(\sum d\right)^2}{n}}{n-1}} = \sqrt{\frac{6280.19 - \frac{(-220.003)^2}{10}}{10-1}} = 12.64937$$

Using the t-test,

$$t = \frac{\overline{d}}{s_d/\sqrt{n}} = \frac{-22.0003}{12.64937/\sqrt{10}} = -5.49996$$

The results of the t-test above give a probability (p-value) of 0.0 from p-value tables, thus there is a 0% probability that the null hypothesis is rejected in error. The null hypothesis is rejected since the mean of the differences is less than 0 and the results show a 100% confidence that the enhanced CxBR agent behaves better than the base CxBR agent at intersections when the SME provides incomplete or incorrect knowledge or the knowledge represented by the knowledge engineer in the model is incomplete and/or incorrect.

Table 7.13 Differences in Speed at Traffic Lights on all Routes

ROUTE_ID	T.L POSITION	T.L COLOR	SPEED at T.L. (Original CxBR)(y)	SPEED at T.L. (Enhanced CxBR)(x)		d = x - y	d^2
2A	1.5	YELLOW	34.00	29.00		-5.00	25.00
2A	1.5	GREEN	34.00	49.00		15.00	225.00
2A	1.5	RED	34.00	0		-34.00	1156.04
2A	1.5	RED	34.00	0		-34.00	1156.04
2A	1.5	RED	34.00	0		-34.00	1156.04
2B	4	GREEN	34.00	49.00		15.00	225.00
2B	4	GREEN	34.00	49.00		15.00	225.00
2B	4	RED	34.00	0		-34.00	1156.04
2B	4	RED	34.00	0		-34.00	1156.04
2B	4	RED	34.00	0		-34.00	1156.04
3	0.9	YELLOW	34.00	4.00		-30.00	900.01
3	0.9	RED	34.00	0		-34.00	1156.04
3	0.9	RED	34.00	0		-34.00	1156.04
3	0.9	RED	34.00	0		-34.00	1156.04
3	0.9	GREEN	34.00	49.00		15.00	225.00
					sum	***-281.00***	***12229.37***

To get the average difference in the agents' speed at traffic lights we use the formula below [199], where *n* is the total number of simulation runs.

$$\overline{d} = \frac{\sum d}{n} = \frac{-281.0048}{15} = -18.733653$$

Therefore, the average difference in speed between the enhanced and original agents at traffic lights is 18.733653

The standard deviation [199] is :

$$s_d = \sqrt{\frac{\sum d^2 - \frac{\left(\sum d\right)^2}{n}}{n-1}} = \sqrt{\frac{12229.37192 - \frac{\left(-281.0048\right)^2}{15}}{15-1}} = 22.30491$$

Using the t-test,

$$t = \frac{\overline{d}}{s_d / \sqrt{n}} = \frac{-18.733653}{22.30491 / \sqrt{15}} = -3.25288$$

The results of the t-test above give a probability (p-value) of 0.0, thus there is a 0% probability that the null hypothesis is rejected in error. The null hypothesis is rejected because the mean of the differences is less than 0 and the results show a 100% confidence that the enhanced CxBR agent behaves better than the original CxBR agent at traffic lights and intersections when there are mistakes in the knowledge acquired from a subject matter expert.

Experiment 4.0

The objective of this experiment is to test the agent's performance when the SME omits a task in a process, or the entire process itself during the knowledge acquisition process. In this experiment, the SME omitted describing a route; thus during the execution of the experiments, the agents encounter an unknown route. The performance of the agent is compared when it is controlled by the base CxBR contexts versus when it is controlled by the enhanced CxBR contexts. The same environmental conditions from previous experiments apply in this experiment. The agent traverses the same routes under the same conditions when controlled by either contexts (base and enhanced). The performance of the agent is measured in terms of achieving the mission goal and the elapsed time to arrive at its destinations. This experiment is carried out when one of the road segments in a route is unknown to the agents.

The CxBR agent and the enhanced CxBR agent move from the starting point of each route to the end point. There are three routes which the agents must traverse. The objective of this experiment is for the agent when controlled by either the base or enhanced contexts to merely arrive at the final destination when a road segment in one of the routes is unknown.

Table 7.14 below shows the elapsed time to arrive at the destination when an unknown road segment is introduced.

Table 7.14 Elapsed time to destination with introduction of an unknown road segment

ROUTE_ID	ELAPSED_TIME (Original CxBR)	ELAPSED_TIME (Enhanced CxBR)
4	Unmatched Context	644.88
4	Unmatched Context	645.46
4	Unmatched Context	644.75
4	Unmatched Context	644.29
4	Unmatched Context	644.77

As can be seen from table 7.14, the agent was unsuccessful in its mission when it was controlled by the base CxBR contexts. The reason is, it encountered an unknown and undefined situation which didn’t have a context defined and as such the mission goal was not accomplished because an exception was raised and the base agent remained in the same position (road segment) endlessly hence it couldn’t arrive at its destination. On the other hand, when the agent was controlled by the enhanced agent, the mission goal was accomplished, this is because during the training phase, the agent learnt of the new road segment and learnt the appropriate actions and attributes of this road segment. Based on the information learnt during training the agent was successful in its mission.

Because the agent was successful in its mission when controlled by the enhanced contexts and unsuccessful in its mission goals when controlled by the base CxBR contexts, we could conclude that the agent when controlled by the enhanced CxBR contexts out-performed the agent when controlled by the base CxBR contexts because of its successful completion of its mission goals.

Experiment 5.0

This experiment is a modification of the experiment performed by Gonzalez & Ahlers [200] in the submarine warfare domain. Detailed description of the contexts, etc. can be read from [200]. In [200], Gonzalez and Ahlers stated that the submarines had "…static slots (defined as those whose values will not change during the simulation)…., examples are maximum speed, quite speed, maximum depth, …..". Note that some aspects of the description of this experiment will not suffice in reality, for example, the angle of dive of the submarine being greater than -10 degrees may not be feasible in reality. Irrespective of the accuracy of the description of a submarine mission portrayed in this experiment, the concept nevertheless is valid. This experiment only shows the extension of the enhancement technique to other domains.

In this experiment, it is assumed that the maximum depth provided by the SME is 2376 ft (0.45 miles[59]), meaning the maximum depth of the body of water in which the submarine can descend to. Note that it is assumed that there are no constraints on the maximum depth of water the submarine is designed to descend to. The mission goal is for the submarine to track an enemy's submarine and return to sector.

Four simulation runs are carried out to compare whether the mission goal is accomplished by the base agent and the enhanced agent. The choice of running four

[59] Depth is not typically measured in miles. In the graphs of experiment 5, it is the authors preference to measure depth in miles

simulations was taken to have enough data to analyze the results. The static enemy submarine[60] will be referred to as the target.

Table 7.15 describes the simulation parameters; the angle of dive refers to the angle which the base agent and enhanced agent dive into the water. Xtarget and Ytarget are the x-y coordinates of the stationary enemy submarine. The maximum speed for base agent and enhanced agent submarines is 11.5 m/h (10 knots). It is assumed both submarines will attain this speed within seconds of starting the simulation and when being controlled by the contexts used in this example. The context hierarchy and the transitions of the contexts are shown in figures 7.18 and 7.19.

Table 7.15 Location of Target on X-Y Plane

SIMULATION RUN	ANGLE of DIVE	XTARGET	YTARGET
1	-89	0.387580639	-0.379818702
2	-77	0.375877751	-0.704649047
3	-45	0.508713655	-0.739862961
4	-83	0.110872406	-0.47510347

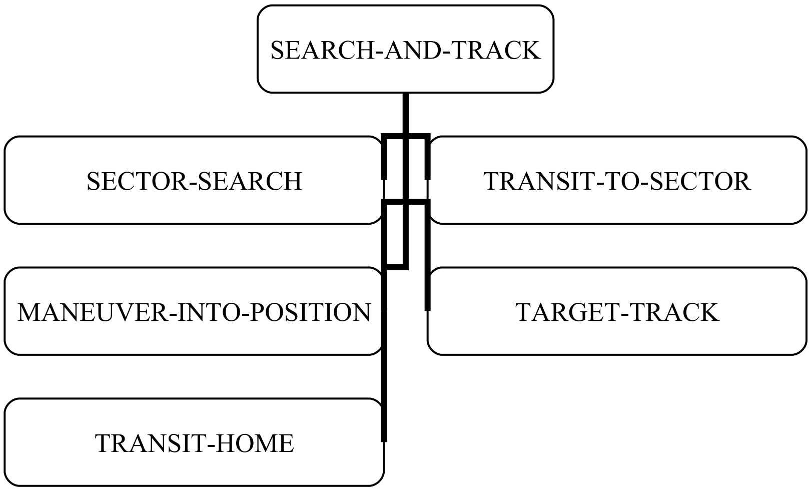


Figure 7.18 Search and Track Mission Context Topology [200]

[60] Enemy submarines are not static in reality, they move around.

The flow of events for the search-and-track mission is as follows: The flow of events starts from the default context (transit-to-sector). The flow is clockwise.

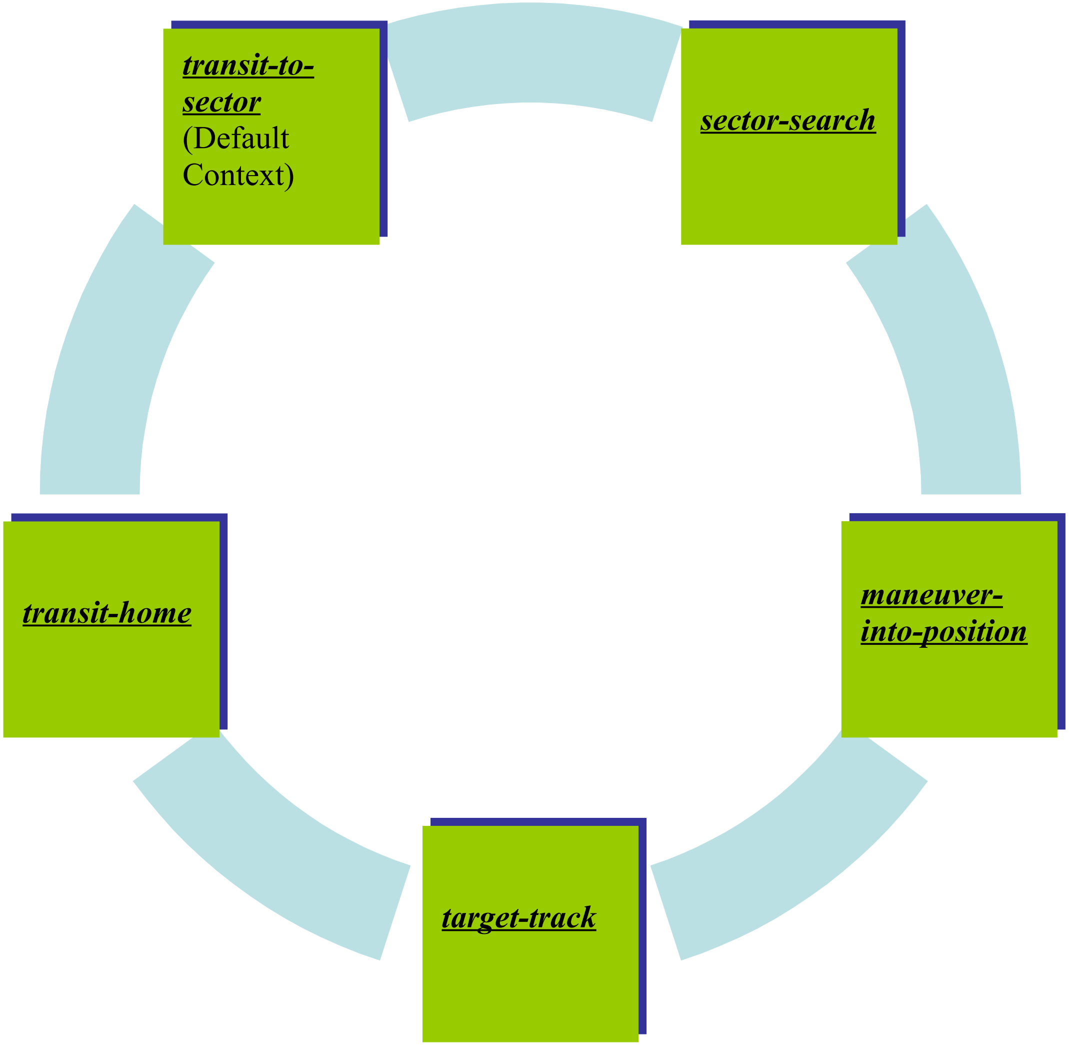


Figure 7.19 Context Transition

Table 7.16 Comparison of Base Agent and Enhanced Agent Mission Success

Simulation Run	Maximum Depth defined by SME	Base Agent (Successful or Not)	Enhanced Agent (Successful or Not)
1	-0.45 miles (2376 ft)	Success	Success
2	-0.45 miles (2376 ft)	Unmatched Context	Success
3	-0.45 miles (2376 ft)	Unmatched Context	Success
4	-0.45 miles (2376 ft)	Unmatched Context	Success

Table 7.16 shows the result of the base agent and enhanced agent in achieving the mission goal. In the first simulation run, it can be seen that both agents are successful in achieving the mission goal; this is because the target (enemy submarine) is higher than the maximum depth defined by the SME. The base agent couldn't achieve the mission goal in simulation runs 2 through 4 because the target was below the maximum depth defined by the SME. On the other hand, because the enhanced agent was trained to learn the actual maximum depth of the body of water, it was able to achieve its mission goal. The path of both agents in simulation runs 1 through 4 is shown in figures 7.20 to 7.23.

Figure 7.20 highlights the contexts on the graph and figures 7.21 to 7.23 shows the direction of the agents motion.

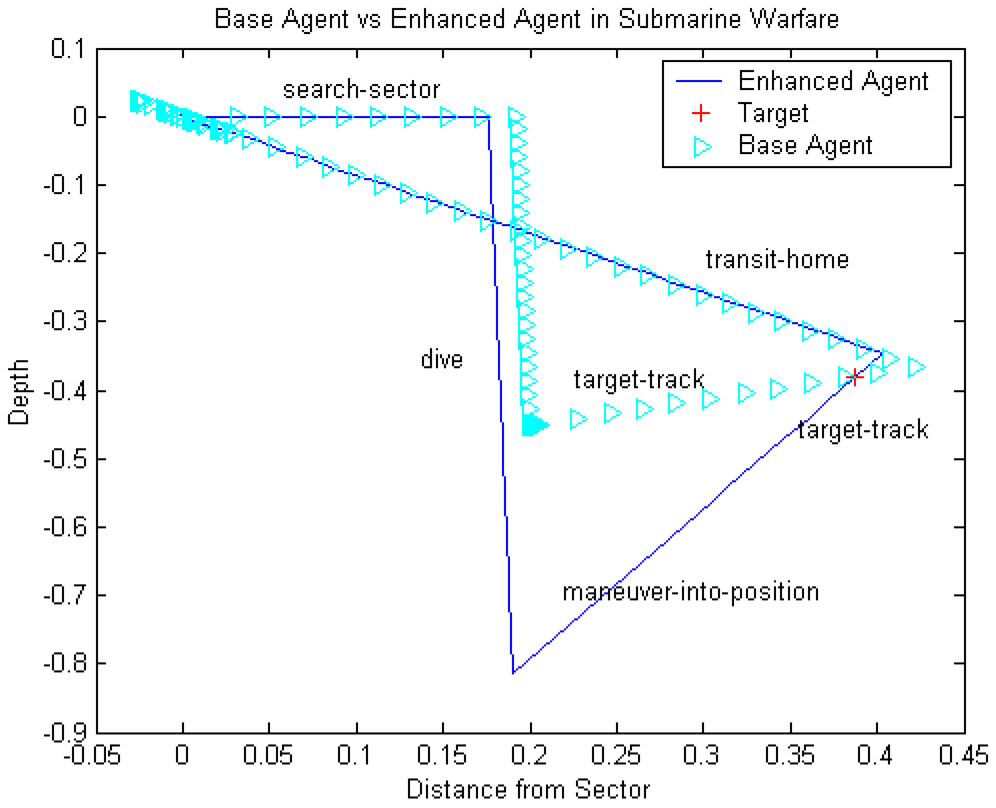


Figure 7.20 Base Agent vs Enhanced Agent both are successful in their mission goal

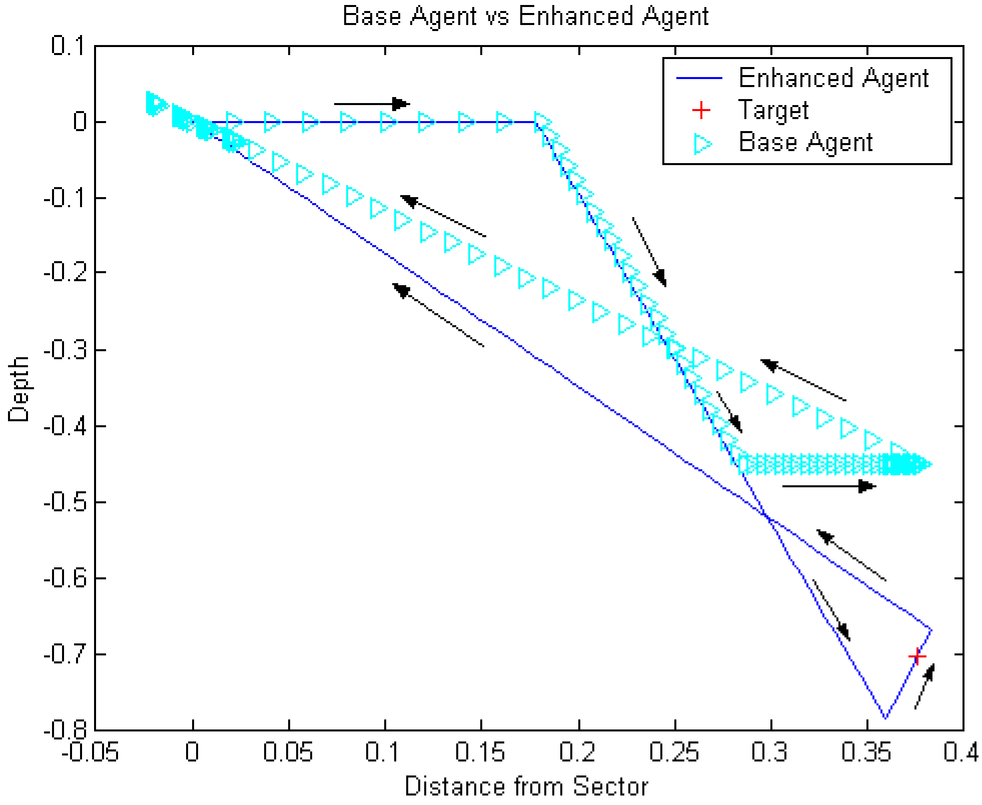


Figure 7.21 Base Agent vs Enhanced Agent with base agent unsuccessful and enhanced agent successful in their mission goal

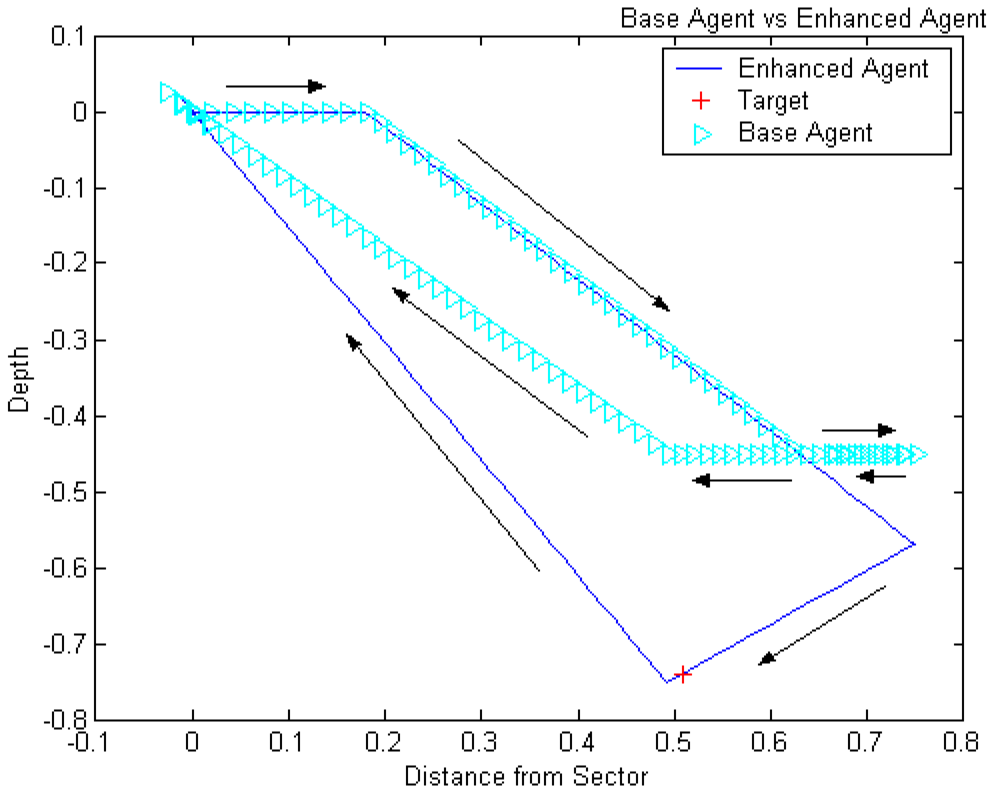


Figure 7.22 Base Agent vs Enhanced Agent with base agent unsuccessful and enhanced agent successful in their mission goal

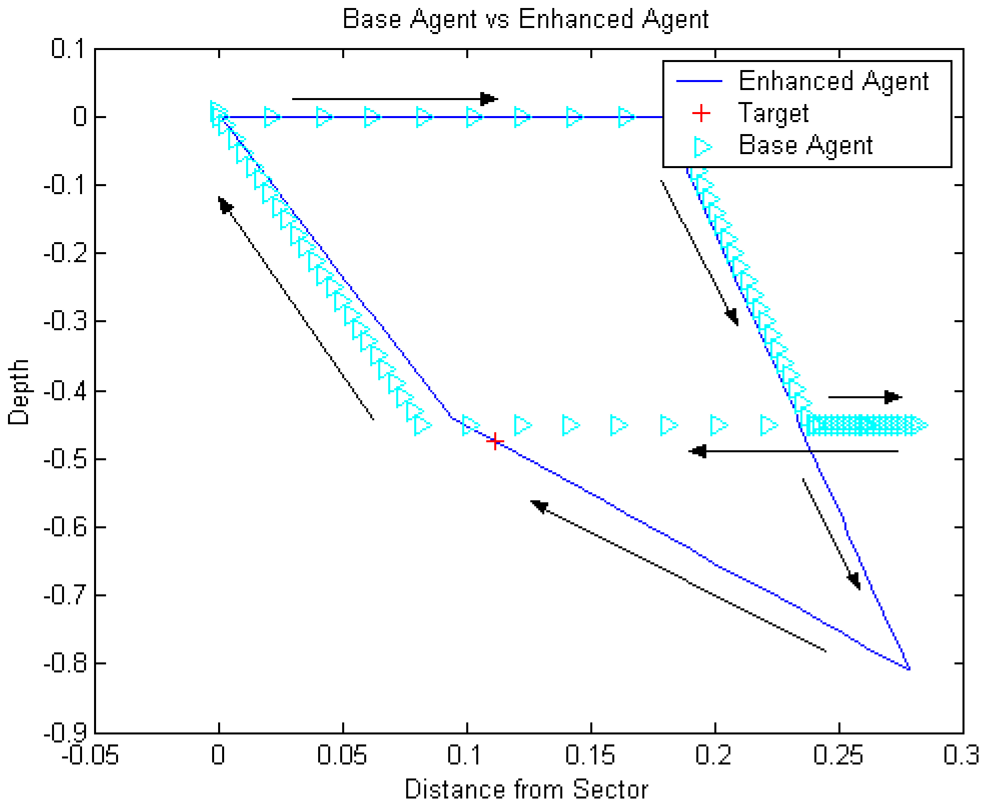


Figure 7.23 Base Agent vs Enhanced Agent with base agent unsuccessful and enhanced agent successful in their mission goal

7.4 Summary of Results

In this section, a summary of the results of all experiments performed is presented. Recall the objectives for performing the experiments highlighted at the beginning of the chapter. Experiments 1 and 2 highlighted scenarios where the SME provided wrong information and as such the agent was destined to make wrong decisions and perform poorly on a given task. In Experiment 1, the agent was supposed to complete a simple mission of choosing the fastest route to a destination. When the agent utilized the information provided by the agent (base contexts), it incorrectly chose the wrong route as being the fastest to the destination. On the other hand, when the agent utilized the enhanced contexts, it made the correct choice. Thus, experiment 1 shows that the enhancement technique leads to an agent making the right choices in a tactical situation. In experiment 2, the performance of the agent was put to test on a mission goal of arriving at a destination with a minimum elapsed time. In experiment 2, the maximum speed value provided by the SME on the various road segments in a route were incorrect. When the agent used the information provided by the SME (base contexts), the elapsed time to its destination was larger than when the agent used the enhanced information. In experiment 2, the SME provided a smaller maximum speed value for the various road segments; the agent learnt that the actual maximum speed values for the road segments were larger. It was noted in experiment 2 that the outcome of the result will remain the same even if the SME had provided a larger maximum speed value and the agent learnt that the correct maximum speed value was smaller, because the agent using the SME provided information (base agent) will be stopped by police and delayed continuously throughout the journey. Experiment 2 thus shows that the performance of the agent using the enhanced information is better than the agent using the base information.

Experiment 3 shows when the SME provided an incorrect process; the SME failed to tell the agent to stop at red traffic lights and intersections. It was shown that the enhanced agent behaved better than the base agent at intersections and red traffic lights. Experiment 4 showed when the SME omitted information about a road segment in a route. The enhanced agent had previously learnt the attributes of the road segment and thus when it encountered it during the execution phase, it was able to achieve the mission goal. On the other hand, the base agent was unable to achieve the mission goal because it lacks learning capabilities. Experiment 5 showed the application of the enhancement technique on a different domain. A submarine warfare domain was used. In this experiment, the submarine was supposed to track an enemy's submarine that was stuck at a given depth, i.e. the submarine was stationary. The SME provided the depth of water at which the submarine could not go any deeper. When the enemy submarine was at a lower depth than that provided by the SME, the base agent was unable to achieve its mission goal, on the other hand, the enhanced agent achieved its mission goal.

7.5 Chapter Summary

In this chapter, the experiments and their results were described. Also, conclusions were made on the effectiveness of the enhancement technique based on the results of the experiments. The results of the experiments show that the agent when controlled by enhanced contexts out-performs an agent controlled by the original contexts in known and unknown situations. The quality of the agents' behavior was also shown to be better after the contexts were enhanced. On the other hand, the reliability of the agents' behavior was unchanged between the enhanced contexts and base contexts. This was because of the consistency in decision making of the CxBR technique, hence the behavior of the agent will always be consistent in the same situation. The usefulness of the enhancement technique was also shown in a decision making situation, where the agent had to choose the fastest route to its destination, the enhanced agent chose the fastest route based on the enhanced attributes of the contexts whereas the original agent choose the fastest route based on the context attributes. In tactical situations, making the right decisions at every point, could lead to a successful mission.

Tables 7.17 & 7.18 summarizes the results from all experiments.

Table 7.17 Quantitative Summary of Results

Exp. No.	Exp. Description	Original CxBR Ratio (O)	Enhanced CxBR Ratio (E)	Difference in Ratios (E-O)
1	Find Tactical Optimal Route	0	1	1
2	Agent Performance	0	1	1
3.1	Agent Behavior at Intersection	0	1	1
3.2	Agent Behavior at T.L	0.4	1	0.6
4	Agent in Unknown Situation	0	1	1
5	Submarine Agent	0.25	1	0.75

Ratios = Number of Successful Runs / Total Number of Runs
In Experiment 1, Original CxBR Ratios = 0 /1 = 0; Enhanced CxBR Ratios = 1/1 =1;
Experiment 2, Original CxBR Ratios = 0 /15 = 0; Enhanced CxBR Ratios = 15/15 =1;
Experiment 3a, Original CxBR Ratios = 0 /10 = 0; Enhanced CxBR Ratios = 10/10 =1;
Experiment 3b, Original CxBR Ratios = 6 /15 = 0.4; Enhanced CxBR Ratios = 15/15 =1;
Experiment 4, Original CxBR Ratios = 0 /5 = 0; Enhanced CxBR Ratios = 5/5 =1;
Experiment 5, Original CxBR Ratios = 1 /4 = 0.25; Enhanced CxBR Ratios = 4/4 =1;

Table 7.17 shows the number of successes for each experiment for the original CxBR and enhanced CxBR agent. The table provides a quantitative view of the success rate of an experiment for both agents. The ratio has been defined as the number of successful runs of a given experiment versus the total number of runs, for example, in experiment one, where the agent’s goal is to find the tactically optimal route, the original CxBR agent didn’t have a successful run, whereas the enhanced agent was successful; the total number of independent experiment runs made in making the decision was 1. On the other hand, in experiment two, there were 15 independent runs; the original CxBR agent was unsuccessful in each run, whereas the enhanced CxBR agent was successful in all 15 runs.

The results shown in Table 7.17 above clearly indicate that the enhanced agent performed in a superior manner to the original agent in each of the tests. This is further explained in Table 7.18 below.

Table 7.18 Summary of Results

Experiment Number	Mission Goal	Purpose of Experiment	Outcome of Experiment
1	Find the tactically optimal route. There were four routes with different destinations, akin to finding the tactically optimal route to an emergency hospital.	To show the effects of using incorrect knowledge in decision making.	The experiment was successful. When the agent used the incorrect knowledge provided by the SME, it chose a wrong route as the tactically optimal route. After the SME knowledge was enhanced, the agent chose the actual tactically optimal route. This is shown in Table 7.17
2	Arrive at the destination on the various routes in the fastest time.	To compare the performance of the enhanced agent and base agent when the SME provides incorrect knowledge	The experiment was successful. The enhanced agent outperformed the base agent. This is shown in Table 7.17
3	Arrive at the destination on the various routes in the fastest time.	To compare the behavior of the enhanced agent and base agent when procedural knowledge provided by the SME is incorrect.	The experiment was successful. The enhanced agent behaved appropriately at intersections and traffic lights. This is shown in Table 7.17
4	Arrive at the destination on the various routes.	To compare the performance of the enhanced agent and base agent when the SME omits a process needed to achieve its mission goal.	The experiment was successful. The enhanced agent was successful in its mission goal while the base agent failed to achieve its mission goal. This is shown in Table 7.17
5	Track an enemy submarine and return to sector	To show the enhancement technique can be used in other domains	The experiment was successful. The enhanced agent was able to track the enemy submarine and return to sector, as shown in Table 7.17

CHAPTER 8 SUMMARY, CONCLUSIONS AND FUTURE WORKS

In this chapter, a summary of what this research was all about is provided as well as conclusions on the results obtained, reasons for the choices made, layouts and strategies for future works with this research as the foundation are provided.

8.1 Summary

In this section, four questions pertinent to this investigation are addressed. These are:

1) What was this investigation all about?
2) What was done during the investigation?
3) How was it done?
4) Why was it done (the various choices)?

8.1.1 What Was Investigated

This research investigated some techniques used in representing human behavior models as described in Chapters 1 and 2. During the investigation, it was noted that most HBR techniques suffer from the limitations inherent in the way knowledge is acquired i.e. the total dependence of knowledge acquisition and representation on subject matter experts. Usually, the experts determine what actions to perform in a given situation and how the agent should behave in all situations as perceived by the expert. In some situations, however, the SME might not know the optimal actions to perform, or might not know how to describe an implicit action, hence that situation may not be properly represented. In other situations, the interpretation of the SME actions by a knowledge engineer may be incorrect or there may be some mistakes in the way a given action or attribute is represented from the way it was intended. Additionally, how do you reconcile differences in different expert opinions for the same situation under the same circumstances? In other words, the information represented in the HBR application is incorrect.

It was noted that the fact that the SME provides the knowledge that determine the behaviors of an agent isn't wrong. What was wrong is the inability of these models to be improved beyond the SME's level of competence or for these models to be improved beyond the mistakes or omissions in the way knowledge for a given situation is represented. An investigation into creating a technique that enhances a HBR model based on the agent achieving the overall mission goal was carried out.

A system where the acquired knowledge - the actions and thus the behavior of the agent can be enhanced based on the mission goal, irrespective of the SME's imparted knowledge was developed. This was achieved by placing the model developed with the SME's knowledge in a simulator and exposing the model to situations imagined and not imagined by the SME. The model was run multiple times until the knowledge acquired from the SME was modified to address these new situations. The model was thus enhanced to perform better, based on the mission goal. Furthermore, this research showed that the CxBR technique can be greatly improved by incorporating the RL technique in it.

8.1.2 What Was Done During the Investigation

After it was determined that most HBR models lacked a mechanism for enhancing the knowledge being represented, a technique utilizing reinforcement learning was hypothesized to do this. Reinforcement learning is a machine learning strategy that assigns rewards (positive or negative) as an agent (simulated or live) interacts with its environment (immediate or distant). Context-Based Reasoning was the HBR technique of choice used in this investigation and Reinforcement Learning was synergistically incorporated within CxBR. To prove the hypothesis, experiments where performed with the agent placed in different situations and a comparison between the original agent and the enhanced agent towards a mission goal was carried out.

8.1.3 How The Investigation Was Done

The experiments to test the hypothesis was carried out in an automobile driving simulation test bed. Some of the constraints on using the CxBR technique were relaxed, for example 'hard-coding' the relationship between contexts (which is the traditional method for representing knowledge using the Context-Based Reasoning technique) in the compatible context segment. Moreover, the context-based reasoning framework developed by Norlander [124] and previously used by others in CxBR simulations was replaced by database table structures in the definition of contexts. Nevertheless, the context hierarchy of having a major context, sub-context, etc. was still maintained in these table structures. The context definitions, context attributes, and context actions were all placed in different database tables with identifiers relating a given context definition to its attributes and actions. Contexts representing the different road types encountered to the best knowledge of the SME were hand-created. The attributes of the various road segments and actions available in the road segments were defined. The maximum speed attribute was of particular interest in this investigation as most of the experiments performed were based on this attribute. The sentinel rules, inference engine, etc. of the CxBR model were developed. The new technique that incorporates reinforcement learning within CxBR and code to implement the new technique were also developed.

The rewards associated with the new technique were defined and the agent was placed in a simulator.

The agent went through a training phase and an execution phase. During the training phase, the maximum speed attribute was trained to reflect the actual maximum speed value of the environment in each road segment. Upon completion of the training phase, the optimal maximum speed for each road segment was learnt. The learnt maximum speed attribute was a correct representation of what was in the environment during training. Also during training, the agent encountered a situation (road segment) which had no context defined for it. The agent was able to learn the attributes of the new road segment, create a context to represent this road segment and learn the optimal maximum speed for this road segment.

After the training phase was complete, the agents' performance and behavior were compared when it was controlled by the enhanced contexts versus when it was controlled by the original contexts.

8.1.4 Why Various Choices Were Made

Various choices pertinent to this investigation were made for different reasons. For example, why was the CxBR technique selected as the HBR paradigm of choice in performing the experiments? Why was an automobile driving simulation test bed used and not a flight simulation test bed? Why was the maximum speed attribute the attribute of choice for training? There are so many questions that could arise from the choices made in this investigation, an attempt will be made to answer most of them.

The choice of CxBR as the modeling technique of choice is based on its modular design and its ability to prune down the search space of the agents' actions to only the relevant actions for any given situation. The ease of use of CxBR has also been established and its flexibility towards modeling any situation has also been established.

The choice of an automobile driving simulation test bed and the subsequent choice of learning the appropriate maximum speed for the road segments in a route is based on an assumption by the author that most people can easily relate to driving and modifying their speed to obey the maximum speed signs as they approach different road segments. The example of learning the appropriate maximum speed can be easily understood by researchers in different domains and the technique presented can thus be utilized in the domain of choice of the readers. Some examples of application of this technique in other domains is presented in the future works section of this chapter.

8.2 Conclusions

In comparing the performance and behavior of the agent when it was controlled by the enhanced CxBR contexts versus when it was controlled with the original CxBR contexts, the agent when controlled by the enhanced CxBR contexts outperformed the agent when it was controlled by the original CxBR contexts based on the SME definition of the original CxBR contexts. The snapshot measurements of behavior of both agents also show the enhanced agent behaving better than the base CxBR agent at traffic lights and intersections. A conclusion can be made that the CxBR enhancement technique introduced in this research enhances contexts that make an agent perform and behave better than when the same agent is being controlled by the original CxBR contexts.

Some observations were made during the course of this investigation. It was noticed that while training the agent, the rate of learning is directly proportional to the value of reward chosen for the test domain. It was noticed that if the difference in value between reinforcing a positive behavior and a negative behavior is small, it takes a longer time for the agent to learn. For example, if one assigned a reward value of '+1' to the agent for choosing the correct maximum speed value in a given context and assigned the value of '-1' for choosing a maximum speed value that exceeds that of the environment, it took a much longer time for the learning process to converge to the correct maximum speed value. Whereas if one had provided a reward value of '+50' for choosing the correct maximum speed value in a given context and assigned a reward value of '-50' for choosing a maximum speed value that exceeds that of the environment, after a few iterations of the agent in its environment, it became apparent what the correct maximum speed value should be.

Also noted was the confidence level in determining that the performance and behavior of the enhanced agent is better than the base agent. This can be attributed to the knowledge acquired in the base CxBR contexts. Conversely, if the knowledge acquired from a SME is absolutely correct and matches all expectations in the environment, there will probably be no difference in the performance of the agent that goes through the enhancement process and that of the original agent. In other words, assuming the maximum speed value in the enhanced contexts were the original values provided by a SME, if these contexts were to go through the enhancement process, there will be nothing to enhance because the maximum speed values already represent what is optimal in the real world. In cases like this, the enhancement process will not show any improvement in performance or behavior but rather, act as a validation mechanism to validate the knowledge from the SME. This point was observed because as a side experiment, an attempt was made to enhance the already enhanced CxBR agent. A conclusion could be made that at a minimum the enhancement technique will produce contexts (agents) that are exactly like the original contexts (agents), hence there are no known disadvantages to using the enhancement technique to attempt to enhance HBR models, other than the time it takes the model to go through the enhancement process.

8.3 Future Research

Although the results obtained from experiments in the automobile driving and in the submarine warfare domains were positive, it would be desirable to more extensively test the application of this technique in other domains. As earlier described, the choice of applying this technique in determining the optimal maximum speed for a given road segment is to provide an example to which most people could relate. A conceptual approach to applying this technique in other domains is presented here and it is the hope of the author that other researchers will test the technique in different domains. Of particular interest is using this technique in correctly identifying purchasing patterns of people or correctly identifying an online search based on the search keyword and the context in which that keyword is used.

Bookstores or movie rental stores typically like to suggest accompanying books or movies after a book purchase or movie rental has been made. To determine this, books are categorized and based on historical data, the pattern of relationships between different categories and different books are presented to buyers. The mechanism for relating the different categories is still a topic that needs further investigations because in some cases, customers don't like the books or movies that were recommended to them. It will be nice to present the enhancement technique to determine the relationships between the various categories (contexts). The enhancement technique can modify and learn new relationships between the various contexts and will eventually provide optimal and accurate book/movie suggestions to potential buyers who might want to purchase a related book. The mission goal in this type of experiment will be to minimize the root square error associated with recommending the wrong book or movie to a customer.

It would also be desirable to provide an online training mechanism to this technique. Currently the contexts are trained offline and then modified based on the optimal value of a variable learnt during training. What if after training a new optimal value emerges, i.e., a new maximum speed value for a road segment is put in place by law? The technique should be able to recognize this and learn in real time what the new value is.

Although the CxBR technique was used as the HBR paradigm of choice in evaluating the enhancement technique, it will be good to see how implementing the enhancement technique with other HBR paradigms will be. Will the results be as encouraging if the enhancement technique is used with other HBR paradigms?

It is also of interest that this technique be embedded in a robot to test its effectiveness. Tests in simulations have shown positive encouragements, but it will be good to test it in the real world using robots to evaluate its ease of use in the real world.

APPENDIX A

TRAINING RESULTS

Table A.1 City Driving Context Training

SESSION_ID	CTX_NAME	RUN_TIME	MAX_SPEED	REWARD
101	CITY	1196.71	35	-10
102	CITY	1045.69	88	-10
104	CITY	309.53	78	-10
105	CITY	318.56	70	-10
106	CITY	414.95	32	-10
107	CITY	474.24	24	-10
108	CITY	354.75	49	1
109	CITY	1348.39	5	-10
110	CITY	358.81	47	1
111	CITY	864	9	-10
112	CITY	339.59	56	-10
113	CITY	337.72	57	-10
114	CITY	456.11	26	-10
115	CITY	361.61	46	1
116	CITY	361.69	46	-1
117	CITY	613.35	15	-10
118	CITY	356.68	48	1
119	CITY	1627.7	4	-10
120	CITY	341.72	55	-10
121	CITY	323.61	66	-10
122	CITY	309.44	78	-10
123	CITY	327.43	65	-10
124	CITY	311.45	76	-10
125	CITY	299.61	91	-10
126	CITY	296.44	95	-10
127	CITY	672.1	13	-10
128	CITY	397.07	36	-10
129	CITY	552.51	18	-10
130	CITY	343.58	54	-10
131	CITY	464.47	25	-10
132	CITY	352.63	50	50
133	CITY	441.03	28	-10
134	CITY	343.67	54	-10
135	CITY	473.46	24	-10
141	CITY	3009.81	2	-10
142	CITY	306.54	82	-10
143	CITY	441.02	28	-10
144	CITY	379.8	40	-10
145	CITY	295.47	96	-10
146	CITY	299.36	91	-10
147	CITY	367.73	44	-10
148	CITY	294.47	98	-10
149	CITY	317.42	71	-10
150	CITY	300.4	90	-10
151	CITY	320.54	68	-10
152	CITY	341.75	55	-10

153	CITY	474.15	24	-10
154	CITY	426.97	30	-10
155	CITY	326.85	65	-10
156	CITY	800.52	10	-10
157	CITY	296.44	95	-10
158	CITY	318.78	70	-10
159	CITY	350.53	51	-10
160	CITY	298.39	92	-10
161	CITY	315.5	73	-10
162	CITY	441.31	28	-10
163	CITY	307.64	80	-10
164	CITY	307.56	81	-10
165	CITY	351.89	50	50
166	CITY	341.9	55	-10
167	CITY	405.98	34	-10
168	CITY	294.33	99	-10
169	CITY	307.5	81	-10
170	CITY	320.59	68	-10
171	CITY	302.34	87	-10
172	CITY	410.18	33	-10
173	CITY	484.28	23	-10
174	CITY	345.54	53	-10
175	CITY	320.61	68	-10
176	CITY	361.59	46	-1
177	CITY	495.34	22	-10
178	CITY	320.51	68	-10
179	CITY	302.56	87	-10
180	CITY	295.55	96	-10
181	CITY	339.52	56	-10
182	CITY	301.48	88	-10
183	CITY	299.33	91	-10
184	CITY	330.64	62	-10
185	CITY	5764.8	1	-10
186	CITY	319.56	69	-10
187	CITY	5776.65	1	-10
188	CITY	434.32	29	-10
189	CITY	312.53	75	-10
190	CITY	298.55	92	-10
191	CITY	349.75	51	-10
192	CITY	464.33	25	-10
193	CITY	299.7	90	-10
195	CITY	354.21	50	50
196	CITY	351.84	50	50
197	CITY	351.6	50	50
198	CITY	352.12	50	50

Table A.2 Freeway Driving Context Training

SESSION_ID	CTX_NAME	RUN_TIME	MAX_SPEED	REWARD
200	FREEWAY	426.81	50	-10
201	FREEWAY	290.72	87	-20
202	FREEWAY	481.64	42	-10
203	FREEWAY	355.66	64	-10
204	FREEWAY	304.78	81	-20
205	FREEWAY	307.67	80	-20
206	FREEWAY	372.05	60	-10
207	FREEWAY	279.56	93	-20
208	FREEWAY	338.03	69	-10
209	FREEWAY	596.2	32	-10
210	FREEWAY	386.02	57	-10
211	FREEWAY	401.06	54	-10
212	FREEWAY	386.08	57	-10
213	FREEWAY	320.58	75	50
214	FREEWAY	457.27	45	-10
215	FREEWAY	976.84	18	-10
216	FREEWAY	301.52	82	-20
217	FREEWAY	311.66	78	-20
218	FREEWAY	341.7	68	-10
219	FREEWAY	279.41	93	-20
220	FREEWAY	427.06	50	-10
301	FREEWAY	320.16	75	50
302	FREEWAY	307.39	80	-20
303	FREEWAY	320.48	75	50
304	FREEWAY	335.58	70	-1
305	FREEWAY	320.53	75	50
306	FREEWAY	317.63	76	-20
307	FREEWAY	320.24	75	50
308	FREEWAY	334.73	70	-1
309	FREEWAY	320.58	75	50
310	FREEWAY	317.59	76	-20
311	FREEWAY	319.64	75	50
312	FREEWAY	334.87	70	-1
313	FREEWAY	320.56	75	50
314	FREEWAY	317.76	76	-20
315	FREEWAY	320.12	75	50
316	FREEWAY	334.74	70	-1
317	FREEWAY	320.51	75	50
318	FREEWAY	317.66	76	-20
319	FREEWAY	320.42	75	50
320	FREEWAY	1124.67	15	-10
321	FREEWAY	436.14	48	-10
322	FREEWAY	305.59	81	-20
323	FREEWAY	519.55	38	-10
324	FREEWAY	372.03	60	-10

325	FREEWAY	293.41	86	-20
326	FREEWAY	555.35	35	-10
327	FREEWAY	376.78	59	-10
328	FREEWAY	1084.08	16	-10
329	FREEWAY	1626.17	10	-10
330	FREEWAY	319.81	75	50
331	FREEWAY	306.83	80	-20
332	FREEWAY	320.61	75	50
333	FREEWAY	335.67	70	-1
334	FREEWAY	319.65	75	50
335	FREEWAY	316.69	76	-20
336	FREEWAY	320.5	75	50
337	FREEWAY	335.6	70	-1
338	FREEWAY	320.5	75	50
339	FREEWAY	317.54	76	-20
340	FREEWAY	320.61	75	50
341	FREEWAY	320.49	75	50
342	FREEWAY	320.5	75	50
343	FREEWAY	320.51	75	50
344	FREEWAY	319.69	75	50
345	FREEWAY	320.63	75	50
346	FREEWAY	320.49	75	50
347	FREEWAY	320.43	75	50
348	FREEWAY	319.72	75	50
349	FREEWAY	320.5	75	50
350	FREEWAY	319.68	75	50

Table A.3 Parking_Lot Driving Context Training

SESSION_ID	CTX_NAME	RUN_TIME	MAX_SPEED	REWARD
425	PARKING_LOT	92.44	60	-20
426	PARKING_LOT	126.72	17	-20
427	PARKING_LOT	106.48	29	-20
428	PARKING_LOT	117.55	21	-20
429	PARKING_LOT	91.42	68	-20
430	PARKING_LOT	88.42	96	-20
431	PARKING_LOT	88.54	90	-20
432	PARKING_LOT	99.5	40	-20
433	PARKING_LOT	94.44	56	-20
434	PARKING_LOT	99.37	41	-20
435	PARKING_LOT	99.12	40	-20
436	PARKING_LOT	87.86	96	-20
437	PARKING_LOT	92.58	63	-20
438	PARKING_LOT	100.53	38	-20
439	PARKING_LOT	109.53	26	-20
440	PARKING_LOT	100.44	39	-20
441	PARKING_LOT	442.07	3	-10
442	PARKING_LOT	88.83	95	-20
443	PARKING_LOT	153.87	10	-1

444	PARKING_LOT	88.41	95	-20
445	PARKING_LOT	111.5	25	-20
446	PARKING_LOT	153.89	10	-1
447	PARKING_LOT	226.03	5	-10
448	PARKING_LOT	153.66	10	1
449	PARKING_LOT	129.71	15	50
450	PARKING_LOT	129.59	16	-20
451	PARKING_LOT	129.58	15	50
452	PARKING_LOT	153.78	10	-1
453	PARKING_LOT	129.7	15	50
454	PARKING_LOT	129.6	16	-20
455	PARKING_LOT	129.67	15	50
456	PARKING_LOT	153.71	10	-1
457	PARKING_LOT	129.59	15	50
458	PARKING_LOT	129.75	16	-20
459	PARKING_LOT	129.57	15	50
460	PARKING_LOT	153.67	10	-1
461	PARKING_LOT	129.7	15	50
462	PARKING_LOT	129.59	16	-20
463	PARKING_LOT	129.57	15	50
464	PARKING_LOT	153.85	10	-1
465	PARKING_LOT	94.42	53	-20
466	PARKING_LOT	98.89	42	-20
467	PARKING_LOT	92.56	66	-20
468	PARKING_LOT	108.55	27	-20
469	PARKING_LOT	89.39	86	-20
470	PARKING_LOT	93.42	57	-20
471	PARKING_LOT	99.53	39	-20
472	PARKING_LOT	92.42	64	-20
473	PARKING_LOT	184.81	8	-10
474	PARKING_LOT	104.5	32	-20
475	PARKING_LOT	129.64	15	50
476	PARKING_LOT	153.88	10	-1
477	PARKING_LOT	129.81	15	50
478	PARKING_LOT	129.57	16	-20
479	PARKING_LOT	129.58	15	50
480	PARKING_LOT	153.87	10	-1
481	PARKING_LOT	129.59	15	50
482	PARKING_LOT	129.58	16	-20
483	PARKING_LOT	129.59	15	50
484	PARKING_LOT	153.82	10	-1
485	PARKING_LOT	129.56	15	50
486	PARKING_LOT	129.58	15	50
487	PARKING_LOT	129.67	15	50
488	PARKING_LOT	129.62	15	50
489	PARKING_LOT	129.67	15	50
490	PARKING_LOT	129.61	15	50
491	PARKING_LOT	129.61	15	50
492	PARKING_LOT	129.83	15	50
493	PARKING_LOT	129.63	15	50

494	PARKING_LOT	129.47	15	50

Table A.4 Ramp Driving Context Training

SESSION_ID	CTX_NAME	RUN_TIME	MAX_SPEED	REWARD
355	RAMP	145.94	42	-20
356	RAMP	147.07	40	-20
357	RAMP	180.41	23	-10
358	RAMP	180.66	23	-10
359	RAMP	976.94	2	-10
360	RAMP	124.61	83	-20
361	RAMP	127.78	75	-20
362	RAMP	128.71	72	-20
363	RAMP	147.14	41	-20
364	RAMP	148.75	39	-20
365	RAMP	229.17	14	-10
366	RAMP	123.66	91	-20
367	RAMP	153.72	35	50
368	RAMP	126.62	77	-20
369	RAMP	149.76	38	-20
370	RAMP	143.69	44	-20
371	RAMP	176.62	24	-10
372	RAMP	123.61	90	-20
373	RAMP	149.74	38	-20
374	RAMP	129.92	68	-20
375	RAMP	153.88	35	50
376	RAMP	161.86	30	-1
377	RAMP	153.75	35	50
378	RAMP	152.82	36	-20
379	RAMP	153.83	35	50
380	RAMP	161.8	30	-1
381	RAMP	154	35	50
382	RAMP	152.78	36	-20
383	RAMP	153.73	35	50
384	RAMP	162.75	30	-1
385	RAMP	153.83	35	50
386	RAMP	152.74	36	-20
387	RAMP	153.86	35	50
388	RAMP	162.77	30	-1
389	RAMP	153.74	35	50
390	RAMP	152.89	36	-20
391	RAMP	153.79	35	50
392	RAMP	161.8	30	-1
393	RAMP	153.92	35	50
394	RAMP	152.71	36	-20
395	RAMP	183.86	22	-10
396	RAMP	131.65	64	-20
397	RAMP	124.69	86	-20
398	RAMP	124.58	86	-20
399	RAMP	137.8	52	-20
400	RAMP	161.8	30	-1

401	RAMP	121.61	99	-20
402	RAMP	171.06	26	-10
403	RAMP	183.9	22	-10
404	RAMP	169.6	27	-10
405	RAMP	153.71	35	50
406	RAMP	147.72	40	-20
407	RAMP	153.77	35	50
408	RAMP	161.87	30	-1
409	RAMP	153.74	35	50
410	RAMP	152.98	36	-20
411	RAMP	153.75	35	50
412	RAMP	162.77	30	-1
413	RAMP	153.77	35	50
414	RAMP	152.72	36	-20
415	RAMP	153.85	35	50
416	RAMP	153.78	35	50
417	RAMP	154.1	35	50
418	RAMP	153.63	35	50
419	RAMP	153.48	35	50
420	RAMP	153.91	35	50
421	RAMP	153.7	35	50
422	RAMP	153.78	35	50
423	RAMP	153.8	35	50
424	RAMP	153.83	35	50

Table A.5 Dirt Driving Context Training

SESSION_ID	CTX_NAME	RUN_TIME	MAX_SPEED	REWARD
502	DIRT	126.03	96	-20
503	DIRT	131.98	72	-20
504	DIRT	171.78	27	-1
505	DIRT	145.31	44	-20
506	DIRT	975.76	2	-10
507	DIRT	124.28	77	-20
508	DIRT	126.99	81	-20
509	DIRT	125.89	87	-20
510	DIRT	146	41	-20
558	DIRT	329.94	8	-10
559	DIRT	135.81	63	-20
560	DIRT	133.11	68	-20
561	DIRT	126.82	72	-20
562	DIRT	123.73	90	-20
563	DIRT	125.26	90	-20
564	DIRT	174.8	25	-1
565	DIRT	130.03	61	-20
566	DIRT	126.52	89	-20
567	DIRT	136.97	63	-20
568	DIRT	126.28	90	-20
569	DIRT	168.14	27	-1

570	DIRT	176.03	25	-1
571	DIRT	171.07	27	1
572	DIRT	167.31	28	1
573	DIRT	172.03	29	1
574	DIRT	162.22	30	50
575	DIRT	162.53	31	-20
576	DIRT	168	30	50
577	DIRT	179.53	25	-1
578	DIRT	166.27	30	50
579	DIRT	157.79	31	-20
580	DIRT	168.17	30	50
581	DIRT	176.27	25	-1
582	DIRT	169.16	30	50
583	DIRT	162.29	31	-20
584	DIRT	165.86	30	50
585	DIRT	177.94	25	-1
586	DIRT	166.47	30	50
587	DIRT	159.19	31	-20
588	DIRT	164.39	30	50
589	DIRT	137.06	67	-20
590	DIRT	130.03	66	-20
591	DIRT	142.01	53	-20
592	DIRT	131.02	76	-20
593	DIRT	173.05	25	-1
594	DIRT	1845.58	1	-10
595	DIRT	137.14	56	-20
596	DIRT	132.34	71	-20
597	DIRT	144.61	42	-20
598	DIRT	126.41	89	-20
599	DIRT	165.4	30	50
600	DIRT	178.68	25	-1
601	DIRT	162.39	30	50
602	DIRT	158.02	31	-20
603	DIRT	165.08	30	50
604	DIRT	174.13	25	-1
605	DIRT	164.48	30	50
606	DIRT	161.33	31	-20
607	DIRT	166.1	30	50
608	DIRT	179.25	25	-1
609	DIRT	169.34	30	50
610	DIRT	164.06	30	50
611	DIRT	165.02	30	50
612	DIRT	164.03	30	50
613	DIRT	162.16	30	50
614	DIRT	165.07	30	50
615	DIRT	162.04	30	50
616	DIRT	165.14	30	50
617	DIRT	166.28	30	50
618	DIRT	161.61	30	50
619	DIRT	167.27	30	50

620	DIRT	165.85	30	50
621	DIRT	157.04	30	50
622	DIRT	162.17	30	50
623	DIRT	163.25	30	50
624	DIRT	169.5	30	50
625	DIRT	169.38	30	50
626	DIRT	161.99	30	50
627	DIRT	170.23	30	50
628	DIRT	161.1	30	50
629	DIRT	163.11	30	50
630	DIRT	164.05	30	50
631	DIRT	164.33	30	50
632	DIRT	163.61	30	50
633	DIRT	166.03	30	50
634	DIRT	164.22	30	50
635	DIRT	167.11	30	50
636	DIRT	164.24	30	50
637	DIRT	163.21	30	50
638	DIRT	163.08	30	50
639	DIRT	166.23	30	50
640	DIRT	162.1	30	50
641	DIRT	164.98	30	50
642	DIRT	169.08	30	50
643	DIRT	167.22	30	50
644	DIRT	166.28	30	50
645	DIRT	166.37	30	50
646	DIRT	164.55	30	50

APPENDIX B

TRAINING GRAPHS

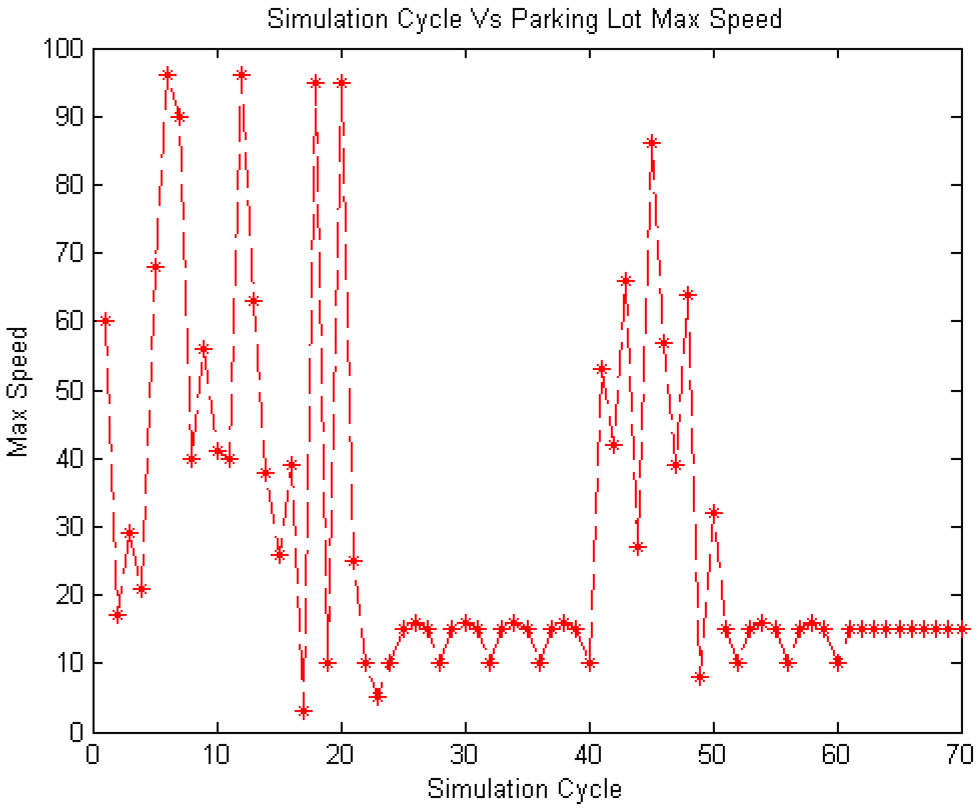


Figure B.1 Training Maximum Speed for Parking Lot Driving

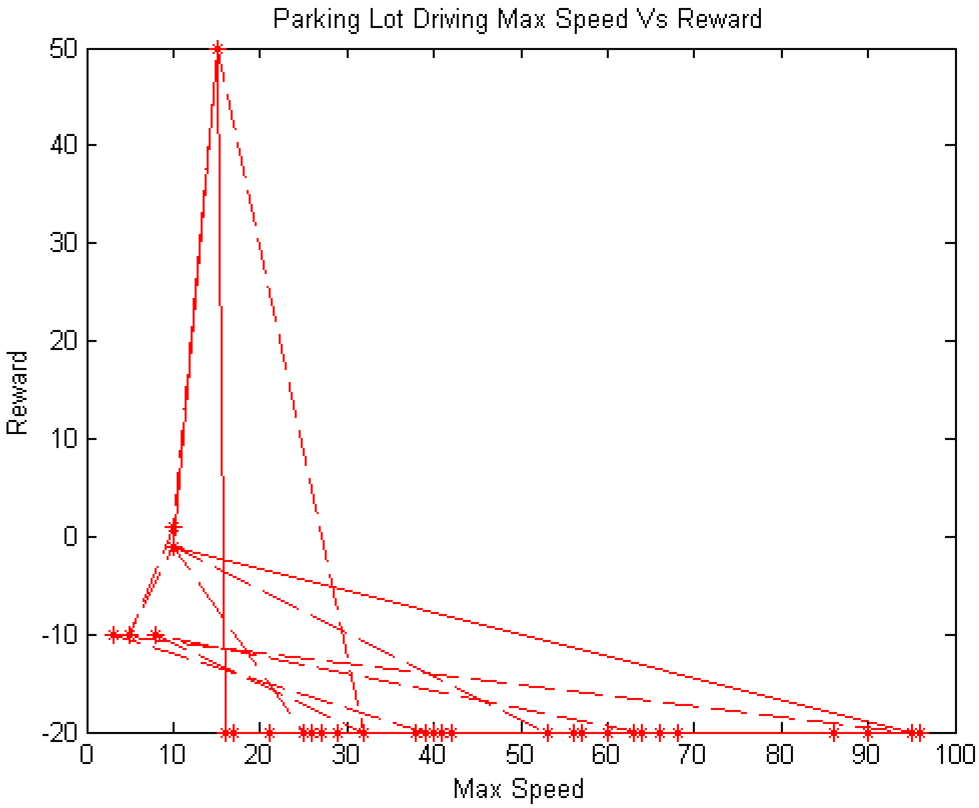


Figure B.2 Parking Lot Driving Maximum Speed Vs Reward

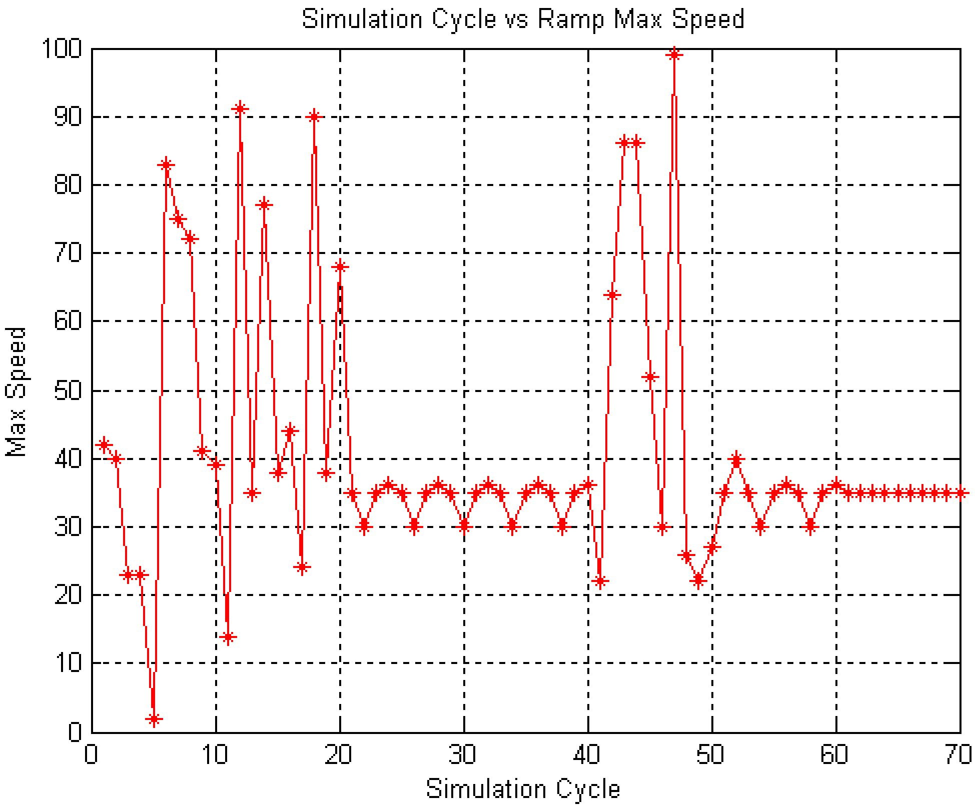


Figure B.3 Training Maximum Speed for Ramp Driving

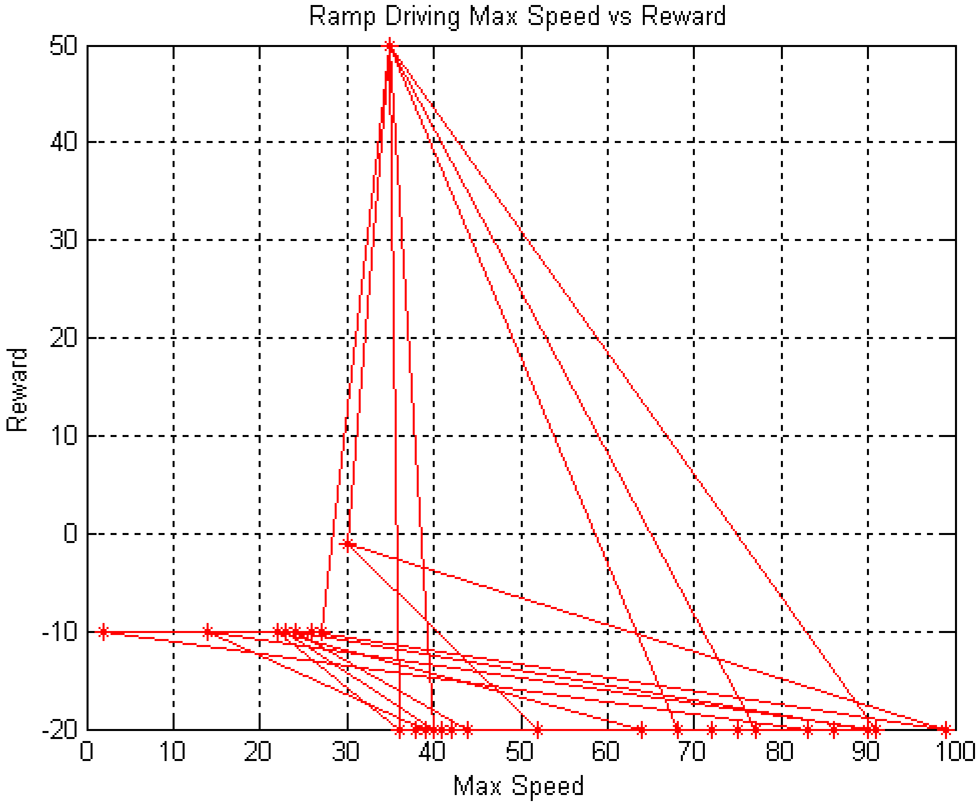


Figure B.4 Ramp Driving Maximum Speed Vs Reward

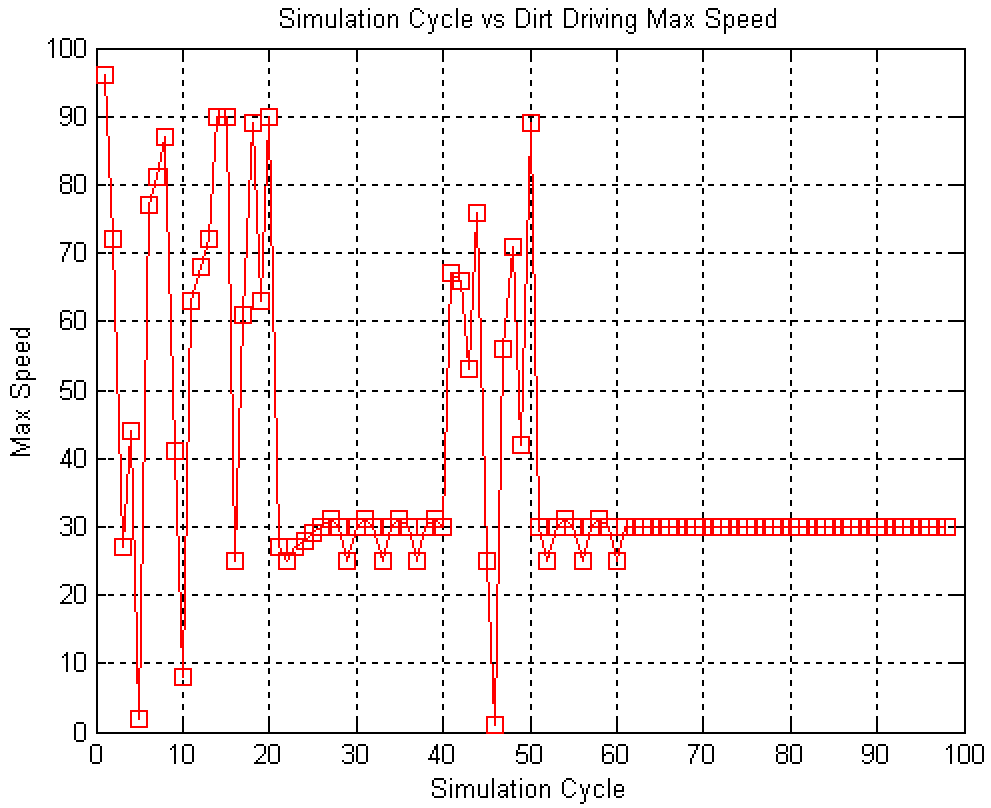


Figure B.5 Training Maximum Speed for Dirt Driving

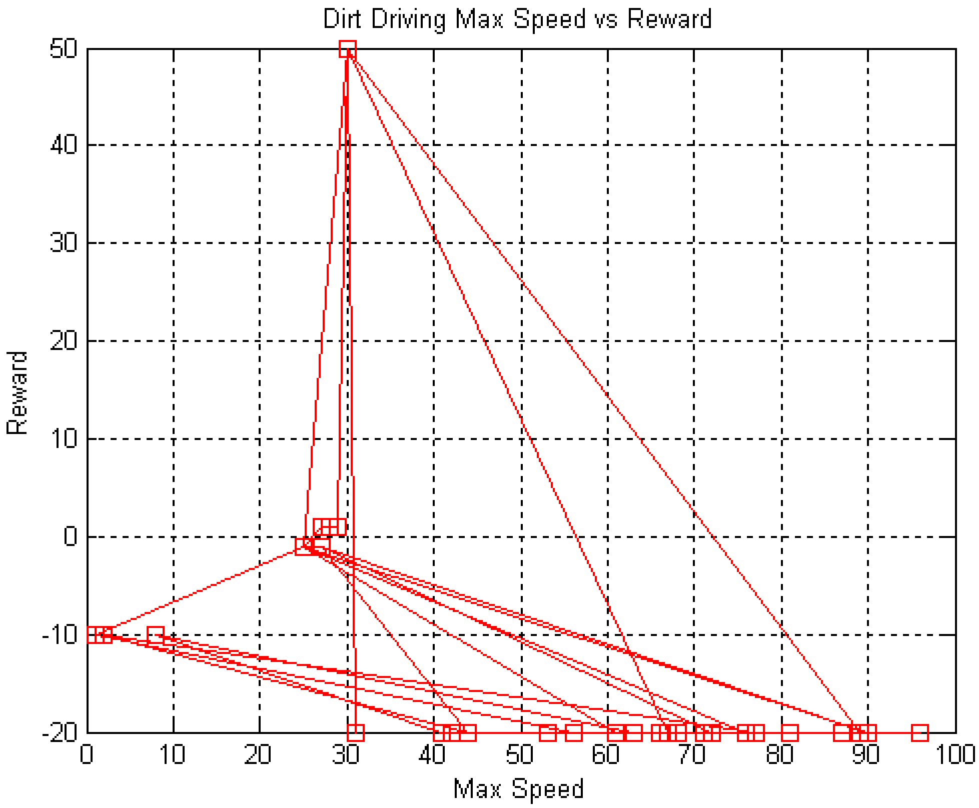


Figure B.6 Dirt Driving Maximum Speed Vs Reward

REFERENCES

[1] Oxford Dictionary; www.dictionary.com

[2] Kokinov, B. (1997). A Dynamic Theory of Implicit Context. In Proceedings of the 2nd European Conference on Cognitive Science, April 9-11, Manchester, UK

[3] Lenox, T., Payne, T., Hahn, S., Lewis, M. & Sycara, K. (1999). MokSAF: How should we support teamwork in human-agent teams? CMU Technical Report. CMU-RI-TR-99-31

[4] Gonzalez, F. G., Grejs, P. & Gonzalez, A. J. (2000). Autonomous Automobile Behaviour through Context-Based Reasoning. In Proceedings of the 12th International Florida Artificial Intelligence Research Society Conference, pp. 2-6, May 22, Orlando, Florida.

[5] Zachary, W., Ross, L., & Weiland, M. (1991). COGNET and BATON: An integrated approach for embedded user models in complex systems. In Proceedings of International Conference on Systems, Man, and Cybernetics, (Vol. 1, p. 689). New York, NY: IEEE.

[6] Turner, E. H. & Turner, R. M. (1991). A Schema-based Approach to Cooperative Behaviour. In Proceedings of the Thirteenth Annual Conference of the Cognitive Science Society.

[7] Gonzalez, A. J. & Ahlers, R. (1994). A Novel Paradigm for Representing Tactical Knowledge in Intelligent simulated Opponents. In Proceedings of the 7th international conference on Industrial and engineering applications of artificial intelligence and expert systems, pp. 515-523, Austin, Texas.

[8] Geffner H., Modelling Intelligent Behaviour: the Markov Decision Approach. Invited talk Iberamia 98, Lect. Notes in AI 1484, H. Coelho (Ed), pp 1--12, Springer, 1998

[9] Gonzalez, A. J., Georgiopoulos, M., DeMara, R. F., Henninger, A. E. & Gerber, W. (1998). Automating the CGF Model Development and Refinement Process by Observing Expert Behaviour in a Simulation. In Proceedings of the 7th Conference on Computer Generated Forces and Behaviour Representation, July, Orlando, Florida.

[10] Fernlund, H. & Gonzalez, A. (2002). An Approach Towards Building Human Behaviour Models Automatically by Observation. In Proceedings of the 1st Swedish–American Workshop on Modeling and Simulation (SAWMAS-2002), Orlando, Florida.

[11] http://www.mesa.afmc.af.mil/html/ambr.htm

[12] Michael van Lent, Ryan McAllinden, Paul Brobst, Barry G. Silverman, Kevin O'Brien, Jason Cornwell, Enhancing the Behavioral Fidelity of Synthetic Entities with Human Behavior Models. Proceedings of the 2003 Conference on Behavior Representation in Modeling and Simulation

[13] Sukthankar G., Hodgins J., Mandel M., Sycara K., Modeling Physical Variability for Synthetic MOUT Agents. In Proceedings of Behavior Representation in Modeling and Simulation Conference (BRIMS 2004)

[14] Wray R. E., Laird J. E., Variability in Human Behavior Modeling for Military Simulations, BRIMS 2003

[15] Lundin, M., Simulating the Effects of Mental Workload on Performance in Tank Crew.

[16] Castro J., Gonzalez A. J., Gerber W. J., "Design and Implementation of CITKA, a Context Based Tactical Knowledge Acquisition System." SAWMAS-2002

[17] Gonzalez A. J., Gerber W. J., Castro J., Automated Acquisition of Tactical Knowledge through Contextualization. BRIMS 2002

[18] Douglas Pearson, John E. Laird, Redux: Example-Driven Diagrammatic Tools for Rapid Knowledge Acquisition, Proceedings of Behavior Representation in Modeling and Simulation, 2004, Washington, D.C.

[19] Kim J., Gil Y., Interactive Acquisition of Behavior Models. Proceedings of the 2003 Conference on Behavior Representation in Modeling and Simulation (BRIMS), 2003.

[20] Bolton A., Buff W., Campbell G., Faster, Cheaper and "Just As Good"? A Comparison of the Instructional Effectiveness of Three HBRs that Vary in Development Requirements.

[21] Henninger A. E., and Gonzalez A. J.,: "Automated Acquisition Tool for Tactical Knowledge" Proceedings of the 10th Annual International Florida Artificial Intelligence Research Symposium, pp. 307-311, May 1997.

[22] Gonzalez, A. J., "Validation of Human Behavioral Models". FLAIRS Conference 1999: 489-493

[23] Rainer Knauf, Ilka Philippow, Avelino J. Gonzalez, Klaus P. Jantke: The Character of Human Behavior Representation and Its Impact on the Validation Issue. FLAIRS Conference 2001: 635-639

[24] Harmon S. Y., Hoffman C. W. D., Gonzalez A. J., Knauf R., Barr V. B. Validation of Human Behavior Representations

[25] Russell S., Norvig P. Artificial Intelligence: A Modern Approach. Englewood Cliffs, NJ. Prentice-Hall, Inc, 1995

[26] Foner, L. N., "What's An Agent, Anyway? A Sociological Case Study," MIT Media Lab, Boston, Technical Report, Agents Memo 93-01, 1993.

[27] Franklin, S., Graesser, A., “Is It An Agent or Just A Program?: A Taxonomy for Autonomous Agents” Proceedings of the Third International Workshop on Agent Theories, Architectures, and Languages, Spinger-Verlag, 1996.

[28] Saeki, S., Gonzalez, A. J., “Competing Context Concept: Experimental Results”, Proceedings of the 9th Computer Generated Forces and Behavioral Representation Conference. Orlando Fl. 2000

[29] Easterbrook, S. M., “Coordination Breakdown: How Flexible is Collaborative Work?” In P. Thomas (ed) CSCW: Requirements and Evaluation, Pp91-106. London: Springer-Verlag (1996)

[30] Gore, B. F., “Human Performance Cognitive-Behavioral Modeling: A Benefit For Occupational Safety”, In B. Chase & W. Karwowski (Eds.), International Journal of Occupational Safety and Ergonomics (JOSE) 2002, Vol., 8, No.3, 339-351

[31] Picard, R. W., “Toward Agents that Recognize Emotion”, M.I.T Media Laboratory Perceptual Computing Section Technical Report No. 515, Appeared: Actes Proceedings IMAGINA, March 1998, pp. 153-165, Monaco.

[32] Archer, R., Lavine, N., Goldberg, S., “Using Human Performance Models to Train Tomorrow’s Soldiers for the Objective Force” Report of Micro Analysis & Design Inc. and the US Army Research Institute, 2001.

[33] Laird, J. E., Newell, A. and Rosenbloom, P. S., “Soar: An Architecture for General Intelligence”, Artificial Intelligence, 33(1), 1987, pp. 1-64.

[34] Kokinov B., Dynamics and Automaticity of Context: A Cognitive Modeling Approach, P. Bouquet et al. (Eds.): Context’99, LNAI 1688, pp. 200-213

[35] Zibetti E., Hamilton E., Tijus C., The Role of Context in Interpreting Perceived Events as Actions, P. Bouquet et al. (Eds.): Context’99, LNAI 1688, pp. 200-213

[36] Turner, R. M. (1997). Context-Mediated Behavior for Intelligent Agents. International Journal of Human-Computer Studies, 3(48), 307-330

[37] Sowa, J. (1999). Knowledge Representation: Logical, Philosophical, and Computational Foundations. New York, PWS Publishing Co

[38] Dey, A., Abowd, G. & Salber, D.(1999). A Context-based Infrastructure for Smart Environments. In Proceedings of the 1st International Workshop on Managing Interactions in Smart Environments (MANSE '99), pp. 114-128, December 13-14, Dublin, Ireland

[39] Gonzalez, A. J. & Ahlers, R. (1995). Context-based representation of intelligent behaviour in simulated opponents. In Proceedings of the 5th Conference on Computer Generated Forces and Behavioural Representation, pp. 53-62

[40] Brezillon, P. (2002). Modeling and Using Context: Past, Present and Future.
http://www.lip6.fr/reports/lip6.2002.010.html
http://ftp.lip6.fr/lip6/reports/2002/lip6.2002.010.pdf

[41] Gonzalez, A. J. & Ahlers, R. (1999). Context-based Representation of Intelligent Behaviour in Training Simulations. Transactions of the Society for Computer Simulation International, 15(4), 153-166

[42] Gonzalez, A. J. & Saeki, S. (2001). Using Contexts Competition to Model Tactical Human Behavior in a Simulation. CONTEXT 2001, pp. 453-456

[43] Saeki, S., & Gonzalez, A. J. (2000). Soft-coding the Transitions between Contexts in CGF's: The Competing Context Approach. In Proceedings of the Computer Generate Forces and Behaviour Representation Conference, Orlando, FL, May 17, 2000.

[44] Turner, R. M. (1998). Context-Mediated Behavior for AI Applications. In Proceedings of the 11th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, IEA/AIE-98, Vol. 1, pp. 538-545, June 1-4, Castell, Spain.

[45] Picard, R. W. Toward Computers that Recognize and Respond to User Emotions. IBM Systems Journal, Vol. 39, Number ¾, Page 705. MIT Media Laboratory, 2000.

[46] Davis, D. N. (2000) Agents, Emergence, Emotion and Representation, IEEE International Conference on Industrial Electronics, Control and Instrumentation (IECON2000) Nagoya, Japan 2000.

[47] Davis, D. N. Modelling Emotion in Computational Agents (2000) http://www2.dcs.hull.ac.uk/NEAT/dnd/papers/ecai2m.pdf

[48] McCauley, T. L., "Implementing Emotions in Autonomous Agents." Master of Science Thesis, University of Memphis, August 1999

[49] Kort B., Reilly R. Analytical Models of Emotions, Learning and Relationships: Towards an Affect-sensitive Cognitive Machine

[50] Giordano, J.C.; Reynolds, P.F.; Brogan, D.C., "Exploring the Constraints of Human Behavior Representation" Proceedings of the 2004 Winter Simulation Conference. R .G. Ingalls, M. D. Rossetti, J. S. Smith, and B. A. Peters, eds. Pages 893- 901

[51] Tenney, Y. J., Diller, D. E., Pew, R. W., Godfrey, K., Deutsch, S. The AMBR Project: A Case-Study in Human Performance Model Comparison.

[52] Pew, Richard W. and Mavor, Anne S. (Eds.) 1998. Modeling Human and Organization Behavior: Application to Military Simulations. National Research Council. National Academy Press, Washington DC.

[53] Mitchell, T. (1997). Machine Learning. McGraw-Hill

[54] Dietterich, T. G. (2003). Machine Learning. In Nature Encyclopedia of Cognitive Science. Macmillan, London

[55] Dietterich, T. G. (1997). Machine Learning Research: Four Current Directions. AI Magazine, 18(4), 97-136

[56] Dietterich, T. G. (2003). Learning and Reasoning.

[57] Christodoulou, C. & Georgiopoulos, M. (2000). Applications of Neural Networks in Electromagnetics. Artech House Inc.

[58] Sidani , T. A.: " Learning Situational Knowledge through Observation of Expert Performance in a Simulation-based Environment" Doctoral Dissertation, Dept. of Electrical and Computer Engineering, University of Central Florida, Orlando, Fl December 1994

[59] Merlo, A., Schotter, A., “Learning by Not Doing: An Experimental Investigation of Observational Learning”. New York University, May 2000

[60] Huitt, W., Hummel, J., “Observational (Social) Learning”, August 1997 - http://home.utm.utoronto.ca/~n_h/ppoint-6.htm
http://chiron.valdosta.edu/whuitt/col/soccog/soclrn.html

[61] Gurney, K., “Computers and Symbols versus Nets and Neurons”, Dept. of Human Sciences, Brunel University Uxbridge, Middx.
http://www.cs.pdx.edu/~bart/cs510games-summer2000/papers/nets.pdf

[62] Krose, B., Patrick van der Smagt, “An Introduction to Neural Networks” 8th Ed

[63] Sutton, R. and Barto, A. Reinforcement Learning: An Introduction. Cambridge, MA: MIT Press, 1998.
http://www.cs.ualberta.ca/%7Esutton/book/ebook/the-book.html

[64] Tesauro, G. (1995). Temporal difference learning and td-gammon. Communications of the ACM, 38(3):58-68.

[65] Sutton, R. (1999). Reinforcement learning: Past, Present and Future. Springer-Verlag pp. 195-197.

[66] Pollack J. B., Blair A. D. Why did TD-Gammon Work? Computer Science Dept., Brandeis University

[67] R. Crites and A. Barto. (1998) "Elevator Group Control Using Multiple Reinforcement Learning Agents". Machine Learning 33, pp. 235-262. (81045 bytes)

[68] Crites, R., and Barto, A. (1995). Improving elevator performance using reinforcement learning. Advances In Neural Information Processing Systems 8. D. Touretzky, M. Mozer, and M. Haselmo, ed. MIT Press, Cambridge, MA.

[69] Singh, S., Bertsekas, D. “Reinforcement Learning for Dynamic Channel Allocation in Cellular Telephone Systems”.

[70] W. Zhang and T. G. Dietterich. Solving combinatorial optimization tasks by reinforcement learning: A general methodology applied to resource-constrained scheduling. Journal of Artificial Intelligence Reseach, 2000.

[71] Gosavi, A., Bandla, N., & Das, T. (2002). Airline seat allocation among multiple late classes with overbooking. IIE Transactions 34:9, 729-742.

[72] Sutton, R. S. “Temporal Credit Assignment in Reinforcement learning”. PhD thesis, University of Massachusetts, Amherst, MA, USA, May 1984.

[73] Sutton, R. S. “Learning to Predict by the Method of Temporal Differences”. Machine Learning, 3:9-44, 1988.

[74] Watkins, C. J. “Learning from Delayed Rewards”. PhD thesis, Kings College, Cambridge, England, May 1989.

[75] Harmon, M. E., Harmon, S., “Reinforcement Learning: A tutorial”. Wright Laboratories, OH.

[76] Kaelbling, L. P., Littman, M. L., Moore, A. W., “Reinforcement Learning: A Survey” AI Access Foundation and Morgan Kaufmann Publishers, 1996.

[77] Whitehead, S. D., Sutton, R. S., Ballard D. H., “Advances in Reinforcement Learning and Their Implications for Intelligent Control” IEEE Trans. On Systems, Man, and Cybernetics, pg 1289 - 1297, 1990

[78] Gosavi, A, 2003. Simulation-Based Optimization Parametric Optimization Techniques and Reinforcement Learning, KluwerAcademic Publishers, Boston, Massachusetts, USA.

[79] Cybenko, G., Gray, R. & Moizumi, K. (1997). Q-Learning: A Tutorial and Extensions. Mathematics of Neural Networks, Kluwer Academic Publishers, Boston/London/Dordrecht

[80] Keerthi, S. & Ravindran, B. (1995). A Tutorial Survey of Reinforcement Learning. Indian Academy of Sciences, Sadhana.

[81] Pratt, E. J. (1999). Elephants Don't Play Backgammon Either. MIT Leg Laboratory, 545 Technology Square, Cambridge, MA 02139

[82] Brooks, R.A., "The Role of Learning in Autonomous Robots", Proceedings of the Fourth Annual Workshop on Computational Learning Theory (COLT '91), Santa Cruz, CA, Morgan Kaufmann Publishers, August 1991, pp. 5–10.

[83] Brezillon, P. "Context Dynamic and Explanation in Contextual Graphs"

Brézillon, P. (2003) Context dynamic and explanation in contextual graphs. In: Modeling and Using Context (CONTEXT-03), P. Blackburn, C. Ghidini, R.M. Turner and F. Giunchiglia (Eds.). LNAI 2680, Springer Verlag (http://link.springer.de/link/service/series/0558/tocs/t2680.htm). pp. 94-106.

[84] Brooks, R. A., "How To Build Complete Creatures Rather Than Isolated Cognitive Simulators", Architectures for Intelligence, K. VanLehn (ed), Erlbaum, Hillsdale, NJ, Fall 1989, pp. 225–239.

[85] Brooks, R.A., "Integrated Systems Based on Behaviors", SIGART Bulletin (2:4), August 1991, pp. 46–50

[86] Maes, P., Brooks, R. A. "Learning to Coordinate Behaviors"

[87] Kokinov, B., Yoveva, M. (1996). "Context Effects on Problem Solving." In Proceedings of the Eighteenth Annual Conference of the Cognitive Science Society. Hillsdale, NJ:Erlbaum.

[88] Brézillon, P. (2005) Role of context in social networks. Proceeding of the 18th International FLAIRS Conference, Invited Special Track "AI for Social Networks, Social Networks in AI", Miami, Florida (To Appear)

[89] Stensrud, B. S., Barrett, G. C., Gonzalez, A. J. Context-Based Reasoning: A Revised Specification. FLAIRS 2004

[90] Kokinov, B. "The Cognitive Architecture DUAL". http://www.socsci.uci.edu/~apetrov/proj/dual/

Kokinov, B. (1994). The Context-Sensitive Cognitive Architecture DUAL. In: Proceedings of the 16th Annual Conference of the Cognitive Science Society. Erlbaum, Hillsdale, NJ.

[91] Kokinov, B. "The Cognitive Model AMBR" http://www.nbu.bg/cogs/personal/kokinov/ambr_i.html

[92] Lorins, P., Brezillon, P., Gonzalez, A. J. (2004). "Context-Based Decision Making: Comparison of CxBR and CxGs Approaches." DSS2004 Conference Proceedings

[93] Kokinov, B. "A Dynamic Approach to Context Modeling." Proceedings of the IJCAI-95 Workshop on Modeling Context in Knowledge Representation and Reasoning. LAFORIA 95/11, 1995

[94] Kokinov, B., Grinberg, M. “Simulating Context Effects in Problem Solving with AMBR”

[95] Kokinov, B., Hadjiilieva, K., Yoveva, M. “Is a Hint Always Useful in Problem Solving? The influence of Pragmatic Distance on Context Effects”

[96] Zibetti, E., Quera, V., Beltran, F. S., Tijus, C. “Contextual Categorization: a Mechanism Linking Perception and Knowledge in Modeling and Simulating Perceived Events as Actions.” Modeling and Using Contexts (pp. 395-408). Berlin: Springer (2001)

[97] Pomerol, J., Brezillon, P. “Context Proceduralization in Decision Making”

[98] Sidani, A., Gonzalez, A. J. “A Framework for Learning Implicit Expert Knowledge through Observation”

[99] Arritt, R. P. and Turner, R. M. “Context-Sensitive Weights for a Neural Network”. In Proceedings of the Fourth International and Interdisciplinary Conference on Modeling and Using Contexts, pages 29 – 39, 2003

[100] Ciskowski, P.”Context-Dependent Neural Nets in Contextual Modelling”. Wroclaw University of Technology, Institute of Engineering Cybernetics, Wroclaw, Poland.

[101] Anderson, J. R., “ACT: A Simple Theory of Complex Cognition” American Psychologist, 51, 355-365. April 1996

[102] Anderson, J. R., Bothell, D., Byrne, M. D., Douglass, S., Lebiere, C., and Qin, Y. “An Integrated Theory of Mind” Psychological Review 111, (4). 1036-1060 (2004)

[103] Laird, J. E., Congdon, C. B., Coulter, K. J., “The Soar User’s Manual, Version 8.2” First Ed. June 1999

[104] Laird, J. E., “The Soar 8 Tutorial” Jan. 2003

[105] “Wearable Behavior Acquisition” http://vismod.media.mit.edu/vismod/demos/learninghumans/behavior.htm

[106] Fernandez-Breis, J. T., Valencia-Garcia, R., Martinez-Bejar, R., Cantos-Gomez, P., “A Context-driven Approach for Knowledge Acquisition: Application to a Leukaemia Domain”

[107] Kass, R., Finin, T., “Rules for the Implicit Acquisition of Knowledge About the User” AAAI-87 Proceedings

[108] Flournoy, R. D., “Leveraging Human Behavior Modeling Technologies to Strengthen Simulation-Based C2 System Acquisition” The MITRE Corporation April 2002

[109] http://www.epistemics.co.uk/Notes/63-0-0.htm

Knowledge Engineering and Management: The CommonKADS Methodology, Schreiber et al., 2000, MIT Press. (Chapter 8)

[110] A.J. Gonzalez, P. J. Drewes and W. Gerber, "Interpreting Trainee Intent in Real Time in a Simulation-based Training System", Transactions of the Society for Computer Simulation International, Vol. 17, No. 3, September 2000, pp. 135-147.

[111] Chen, L., Chan, C. W., “Ontology Construction from Knowledge Acquisition”

[112] Simon, G., “Knowledge Acquisition and Modeling for Corporate Memory: Lessons Learnt from Experience” Proceedings of the 10th Knowledge Acquisition for Knowledge-based Systems Workshop (KAW'96), pp. 41/1-41/18, 1996

[113] Zachary, W. W., Ryder, J. M., Hicinbothom, J. H., "Cognitive Task Analysis and Modeling of Decision Making in Complex Environments". J. Cannon-Bowers & E. Salas (Eds.), Decision making under stress: Implications for training and simulation. Washington, DC: American Psychological Association

[114] Ryder, J. M., Weiland, M. Z., Szczepkowski, M. A., Zachary, W. M., "Cognitive Engineering of a New Telephone Operator Workstation Using COGNET" International Journal of Industrial Ergonomics 22 (1998)

[115] Ritter, F. E., Shadbold, N. R, Elliman, D., Young, R. M, Gobet, F., Baxter, G. D., "Techniques for Modeling Human Performance in Synthetic Environments: A Supplementary Review" June 2002

[116] Young, R. M., Lewis, R. L., "The Soar Cognitive Architecture and Human Working Memory" A. Miyake & P. Shah (Eds), Models of Working Memory: Mechanisms of Active Maintenance and Executive Control. New York: Cambridge University Press. Nov. 1997

[117] Chong, R. S., Kieras, D. E., "Modeling with Perceptual, Memory and Motor Constraints: An EPIC-Soar Model of a Simplified Enroute Air Traffic Control Task" Aug. 2000

[118] Nason, S., Laird, J. E., "Soar-RL: Integrating Reinforcement Learning with Soar"

[119] Balkenius, C., and Morén, J. (2000). A computational model of context processing. In Meyer, J-A., Berthoz, A., Floreano, D., Roitblat, H. L., Wilson, S. W. (Eds.), From Animals to Animats 6: Proceedings of the 6th International Conference on the Simulation of Adaptive Behaviour. Cambridge, MA: MIT Press.

[120] Balkenius, C. (2003). Cognitive processes in contextual cueing. In Schmalhofer, F., Young, R. M., and Katz, G. (Eds.), Proceedings of the European Cognitive Science Conference 2003 (pp. 43-47). Mahwah, NJ: Lawrence Erlbaum Associates.

[121] Balkenius, C., and Winberg, S. (2004). Cognitive modeling with context sensitive reinforcement learning. In Proceedings of AILS '04. Lund: Dept. of Computer Science.

[122] Hollnagel, E. (1993) Human reliability analysis: Context and control. London: Academic Press

[123] http://www.ida.liu.se/~eriho/COCOM_M.htm

[124] Norlander, L., “A Framework for Efficient Implementation of Context-Based Reasoning in Intelligent Simulation”, Masters Thesis, Dept. of Electrical and Computer Engineering, University of Central Florida, Orlando, FL., 1998

[125] 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. 2000.

[126] Henninger, A. E., “ATTacK: Automated Acquisition Tool for Tactical Knowledge” Masters Thesis, University of Central Florida, Orlando, FL. 1996

[127] Viet C. Trinh, Brian S. Stensrud, Avelino J. Gonzalez, “Implementation of a Prototype Context-Based Reasoning Model onto a Physical Platform” SAWMAS 2004

[128] Roberto C. Sanchez, Avelino J. Gonzalez, “Improving Computational Efficiency in Context-Based Reasoning Simulations” SAWMAS 2004

[129] Bonzon, P. (2000). Contextual Learning: Towards Using Contexts to Achieve Generality. Vol. 10, pp. 127-141, Kluwer Academic Publishers.

[130] Lebiere, C., “Introduction to ACT-R 5.0 Tutorial”, 24th Annual Conference Cognitive Science Society” http://act-r.psy.cmu.edu/tutorials/

[131] Kukla, R., Kerridge, J. and Willis, A. (2003). Application of Context Mediated Behaviour to a multi-agent pedestrian flow model (PEDFLOW). Transportation Research Board Annual Meeting, Washington, 82nd Annual Meeting

[132] Brezillon, P., “Context-based Modeling of Operators’ Practices by Contextual Graphs”. 14th Mini Euro Conference, Luxembourg, May 5-7, 2003

[133] Brezillon, P., Pasquier, L., Pomerol, J., “Modelling Decision Making with Context-Based Reasoning and Contextual Graphs. Application in Incident Management on a Subway Line”.

[134] Kieras, D. E., Meyer, D. E., (1997) “An Overview of the EPIC Architecture for Cognition and Performance with Application to Human-Computer Interaction.” Human-Computer Interaction, 12, 391-438.

[135] Byrnes, R. B., “The Rational Behavior Model: A Multi-Paradigm, Tri-level Software Architecture for the Control of Autonomous Vehichles”, Phd Dissertation, Naval Postgraduate School, Monterey, California, March 1993.

[136] Holden, M. J., “Ada Implementation of Concurrent Execution of Multiple Tasks in the Strategic and Tactical Levels of the Rational Behavior Model for the NPS

Phoenix Autonomous Underwater Vehicle (AUV)", Masters Thesis, Naval Postgraduate School, Monterey, California, September 1995.

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

[138] Fernlund, H., "Evolving Models from Observed Human Performance", Phd Dissertation, University of Central Florida, Orlando, FL., Spring 2004.

[139] Parr, R., Russell, S. (1998), "Reinforcement Learning with Hierarchies of Machines." In Advances in Neural Information Processing Systems, Vol. 10, pp. 1043-1049 Cambridge, MA. MIT Press.

[140] Dietterich, T. G., "An Overview of MAXQ Hierarchical Reinforcement Learning"

[141] Barto, A. G., Mahadevan, S. "Recent Advances in Hierarchical Reinforcement Learning". Discrete Event Systems, Special issue on reinforcement learning, 13:41-77, 2003.

[142] Tesauro, G., "Practical Issues in Temporal Difference Learning"

[143] Hogkolan, K. T., "Teaching Robots Behavior Patterns by Using Reinforcement Learning: How to Raise Pet Robots with a Remote Control" Masters Thesis, Computer Science and Engineering Dept., NADA.

[144] Bridle, R., McCreath, E., "Learning a Transition model to Enhance the Performance of Reinforcement Learning"

[145] Dietterich, T. G. (2000), "Hierarchical Reinforcement Learning with the MaxQ Value Function Decomposition." Journal of Artificial Intelligence Research

[146] Kaelbling, L. P. (1993), "Hierarchical Learning in Stochastic Domains: Preliminary Results" In Proceedings of the Tenth International Conference on Machine Learning, pp. 167-173 San Francisco, CA. Morgan Kaufmann.

[147] Dayan, P., Hinton, G. (1993), "Feudal Reinforcement Learning." In Advances in Neural Information Processing Systems, 5, pp. 271-278. Morgan Kaufmann, San Francisco, CA.

[148] Hernandez-Gardiol, N., Mahadevan, S. "Hierarchical Memory-Based Reinforcement Learning"

[149] Lane, T., Kaelbling, L. P., "Toward Hierarchical Decomposition for Planning in Uncertain Environments"

[150] Bakker, B., Schmidhuber, J., "Hierarchical Reinforcement Learning Based on Subgoal Discovery and Subpolicy Specialization"

[151] Hengst, B., "Generating Hierarchical Structure in Reinforcement Learning from State Variables" In PRICAI 2000 Topics in Artificial Intelligence, pages 533–543,San Francisco, 2000. Springer.

[152] Hengst, B., "Discovering Hierarchy in Reinforcement Learning" PhD Thesis School of Computer Science and Engineering, University of New South Wales Australia, December 2003

[153] Hengst, B., "Variable Resolution Hierarchical RL"

[154] McGovern, A., Sutton, R. S., "Macro-Actions in Reinforcement Learning: An Empirical Analysis" Technical Report

[155] McGovern, A., Precup, D., Ravindran, B., Singh, S., Sutton, R. S., "Hierarchical Optimal Control of MDP's" Proceedings of the 10th Yale Workshop on Adaptive and Learning Systems.

[156] Munos, R., "Finite-Element Methods with Local Triangulation Refinement for Continuous Reinforcement Learning Problems." European Conference on Machine Learning, 1997 pp 170-182.

[157] Gang, W., Mahadevan, S., "Hierarchical Optimization of Policy-Coupled Semi-Markov Decision Processes" International Conference on Machine Learning (ICML-99)

[158] Singh, S. "Reinforcement Learning with a Hierarchy of Abstract Models" Proceedings of the 10th National Conference on Artificial Intelligence 1992.

[159] Thrun, S., Schwartz, A. "Finding Structure in Reinforcement Learning." Advances in Neural Information Processing Systems 7, Morgan Kaufmann, San Mateo (1995)

[160] Wiering, M., Schmidhuber, J., "HQ-Learning: Discovering Markovian Subgoals for Non-Markovian Reinforcement Learning" Technical Report IDSIA-95-96, October 1996

[161] Goel, S., Huber, M. "Subgoal Discovery for Hierarchical Reinforcement Learning Using Learned Policies" Proceedings of the 16th International FLAIRS Conference. Pp. 346-350, St. Augustine, FL 2003

[162] Barto, A. G., (2003) "Reinforcement Learning" In Handbook of Brain Theory and Neural Networks, Second Edition M.A. Arbib (Ed.), pages 963-968. Cambridge: MIT Press.

http://cogns.northwestern.edu/RL-handbook-proofs.pdf

[163] Reinforcement Theory: http://www.as.wvu.edu/~sbb/comm221/chapters/rf.htm

[164] Ratitch, B., Precup, D. (2002) "Characterizing Markov Decision Processes" In 13th European Conf. on Machine Learning (ECML'02), Helsinki, Finland.

[165] Agogino, A. K., Tumer, K. "Unifying Temporal and Structural Credit Assignment Problems" In Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems - Volume 2 (AAMAS'04) , pp. 980-987 New York City, New York, USA july 2004

[166] Wray, R., Jones, R. "How to Build Intelligent Interactive Agents Using Soar" BRIMS Conference, Scottsdale, AZ, May 12, 2003.

[167] Riedmiller, M. "Application of Sequential Reinforcement Learning to Control Dynamic Systems"

[168] Madden, M.G.M., Nolan, P.J. "Application of AI Based Reinforcement Learning to Robot Vehicle Control

[169] Kaelbling, L.P., Littman, M. L., Cassandra, A. R. "Planning and Acting in Partially Observable Stochastic Domains" Artificial Intelligence, Vol. 101, 1998

[170] Kimura, H., Kobayashi, S. "Reinforcement Learning for Continuous Action using Stochastic Gradient Ascent"

[171] Smart, W. D., Kaelbling, L. P., "Practical Reinforcement Learning in Continuous Spaces"

[172] MICHAEL T. ROSENSTEIN and ANDREW G. BARTO, "Supervised Actor-Critic Reinforcement Learning"

http://www-anw.cs.umass.edu/~mtr/papers/RosensteinM04b.pdf

[173] Mark Humphrys, “Action Selection methods using Reinforcement Learning” Phd Thesis Trinity Hall, Cambridge, June 1997 http://computing.dcu.ie/~humphrys/Notes/RL/how.q.html

[174] Nilsson, N. J., “Introduction to Machine Learning” 1996

[175] Glorennec, Pierre Yves, ”Reinforcement Learning: an Overview”, European Symposium on Intelligent Techniques, Aachen Germany.

[176] Barto, A.G., DIETTERICH, T. G., “Reinforcement Learning and its Relationship to Supervised Learning”

[177] http://www.funderstanding.com/observational_learning.cfm

[178] Singh, S. P., “Learning to Solve Markovian Decision Processes” PhD Dissertation, February 1994, University of Massachusetts

[179] Mahadevan, S., “Machine Learning for Robots: A Comparison of Different Paradigms”

[180] Sallans, B., “Reinforcement Learning for Factored Markov Decision Processes” Phd Dissertation, 2002, University of Toronto

[181] Dayan, P., “Unsupervised Learning” Appeared in Wilson, RA & Keil, F, editors. The MIT Encyclopedia of the Cognitive Sciences.

[182] Hasinoff, S.W., “Reinforcement Learning for Problems with Hidden State” September, 2003

[183] Konda, V. R., Tsitsiklis, J. N., “On Actor-Critic Algorithms” http://web.mit.edu/jnt/www/Papers/J094-03-kon-actors-pre.pdf

[184] Kretchmar, R. M., “Parallel Reinforcement Learning”

[185] Shapiro, D., Langley, P., Shachter, R., "Using Background Knowledge to Speed Reinforcement Learning in Physical Agents"

[186] Baird, L., Moore, A. "Gradient Descent for General Reinforcement Learning"

[187] Dietterich, T. G., Flann, N. S., "Explanation-Based Learning and Reinforcement Learning: A Unified View"

[188] Guo, M., Liu, Y., Malec, J., "A New Q-learning Algorithm Based on the Metropolis Criterion

[189] Wiering, M., "Fast Online Q(λ)"

[190] Peng, J., "Incremental Multi-Step Q-Learning"

[191] Dearden, R., Friedman, N., Russell, S., "Bayesian Q-learning"

[192] Wan A.D.M. and Braspenning P.J. (1997). Context and Generalization in a Multiple Goal State Reinforcement Learning Task. In: K. van Marcke & W. Daelemans (eds.), Proceedings of the Ninth Dutch Conference on Artificial Intelligence (NAIC '97), pp. 293-302.

[193] Bridle, R., McCreath, E., "Improving the Learning Rate by Inducing a Transition model" In proceeding of the Third International Joint Conference on Autonomous Agents and Multiagent Systems(AAMAS2004), Columbia University New York City, July 2004

[194] Schutte, P. C., "Definitions of Tactical and Strategic: An Informal Study". NASA/ TM-2004-213024, November 2004 http://techreports.larc.nasa.gov/ltrs/PDF/2004/tm/NASA-2004-tm213024.pdf

[195] Latorella, K., Chamberlain, J., "Tactical vs. Strategic Behavior: General Aviation Piloting in Convective Weather Scenarios", Proceedings of the Human Factors & Ergonomics Annual Meeting, Baltimore, MD., 2002 http://techreports.larc.nasa.gov/ltrs/PDF/2002/mtg/NASA-2002-46hfes-kal.pdf

[196] Petty, M.D., "Benefits and Consequences of Automated Learning in Computer Generated Forces Systems" Information and Security: An International Journal, Vol. 12, No.1, 2003, pg 63-74 http://cms.isn.ch/public/docs/doc_6934_259_en.pdf

[197] Gonzalez, A. J., Dankel, D. D., "The Engineering of Knowledge-Based Systems, Theory and Practice" Prentice Hall, NJ

[198] Fernlund, H, Gonzalez, A. J., "Using GP to Model Contextual Human Behavior – Competitive with Human Modeling Performance" http://www.cs.bham.ac.uk/~wbl/biblio/gecco2004/LBP015.pdf#search='gencl%20fernlund'

[199] Mason, R.D., Lind, D.A., Marchal, W.G., "Statistics, An Introduction", Duxbury Press, 4th Edition

[200] 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.