Genetic Context Learning – Automatic behavior modeling from observed human performance

Hans Fernlund, Sven E. Eklund
Dalarna University
Computer Science Department
78188 Borlänge, Sweden
hfe@du.se, sek@du.se

Avelino J. Gonzalez
University of Central Florida
School of Electrical Engineering and Computer Science
Orlando, FL 32816-2450
gonzalez@pegasus.cc.ucf.edu

Keywords:
Simulated agents, Learning by Observation, Context-Based Reasoning, Genetic Programming

ABSTRACT: *To create a realistic environment, some simulations require simulated agents with human behavior pattern. Creating such agents with realistic behavior is tedious and time consuming work. This paper describes a new approach that automatically builds human behavior models for simulated agents by observing human performance. With an automatic tool that builds human behavioral agents, the development cost and effort could dramatically be reduced. This research synergistically combines Context-Based Reasoning (CxBR), a paradigm especially developed to model tactical human performance within simulated agents, with the Genetic Programming machine learning algorithm able to construct the behavior knowledge in accordance to the CxBR paradigm. This synergistic combination of well-documented AI methodologies has resulted in a new approach that very effectively and automatically builds simulated agents with human behavior. This algorithm was exhaustively tested with five different simulated agents created by observing the performance of five humans driving an automobile simulator. The agents show, not only the capabilities to automatically learn and generalize the behavior of the human observed, but they also captured some of the personal behavior patterns observed among the five humans. Furthermore, the agents exhibited a performance that was at least as good as agents developed manually by a knowledge engineer*.

1. Introduction

Building human behavior models is very complex and time-consuming. Extracting and processing the knowledge from the subject matter expert is a very intricate task. To get the expert to express the behavior in an articulate way, analyze the information and later implement it in the model are time consuming and clear sources of misinterpretation. It is almost impossible to develop a mathematical formalism of human behavior, and the cost and effort to build good models can be very high. In the real world, there often exist problem domains where the knowledge might be incomplete, imprecise or even conflicting. Often, the models are built on inflexible doctrines. This can cause the entities to behave "too perfectly" without human similarities [1]. It has also been shown that the manual routines taught by the experts, are not necessarily the routines used by the experts themselves [2].

The use of a learning system that could automatically extract knowledge and construct a behavior model could reduce the problems mentioned above. If the system would be able to observe a human's behavior and automatically build a behavior model the development complexity would be dramatically reduced.

This paper presents an approach to building human behavior models automatically. The approaches employs Context-Based Reasoning (CxBR) and Genetic Programming (GP) to implement a methodology called Learning by Observation, which will learn the behavior of a human merely by observation.

2. Learning by Observation

Inspired by how humans and other mammals learn, the machine learning community has developed a number of theories on learning by observation, applied in various areas. The interest in this research is to investigate learning of human behavior by observation. The intent is not only to use the observations to learn, but also to learn the behavior of the observed entity. A number of advantages could be gained by using learning by observation in lieu of traditional knowledge acquisition and development methods. The development cost and complexity could be reduced and the models might be able to capture behavior patterns that would not have been found otherwise. The research described in this paper defines learning by observation as follows:

> *The agent adopts the behavior of the observed entity only through the use of data collected through observation.*

To create a behavioral model with vide variety of human features there must exist an efficient modeling framework. CxBR was proposed by Gonzalez and Ahlers [3] to have many of these features. Using CxBR as a framework satisfies the prerequisites for implementation of human behavior. If the model is to be created automatically, the framework needs to be equipped with a learning paradigm that will work in conjunction with the framework without disturbing the supported human features. The learning paradigm used in this research is GP. We now briefly discuss CxBR and GP.

3. Context-Based Reasoning

Gonzalez and Ahlers [5] presented Context-Based Reasoning (CxBR) as a modeling technique that can efficiently represent the behavior of humans in intelligent software agents. Later results showed that it is especially well-suited to modeling tactical behavior. CxBR is based on the idea that:

- A recognized situation calls for a set of actions and procedures that properly address the current situation.
- As a mission evolves, a transition to another set of actions and procedures may be required to address the new situation.
- Things that are likely to happen while under the current situation are limited by the current situation itself.

CxBR encapsulates tactical knowledge into hierarchically-organized contexts the knowledge about appropriate actions and/or procedures as well as compatible new situations.

Mission Contexts define the mission to be undertaken by the agent. While it does not control the agent per se, the Mission Context defines the scope of the mission, its goals, the plan, and the constraints imposed (time, weather, rules of engagement, etc). The Major Context is the primary control element for the agent. It contains functions, rules and a list of compatible next Major Contexts. Identification of a new situation can now be simplified because only a limited number of all situations are possible under the currently active context. Sub-Contexts are abstractions of functions performed by the Major Context which may be too complex for one function, or that may be employed by other Major Contexts. This encourages re-usability. Sub-Contexts are activated by rules in the active Major Context. They will de-activate themselves upon completion of their actions.

One and only one specific Major Context is always active for each agent, making it the sole controller of the agent. When the situation changes, a transition to another Major Context may be required to properly address the emerging situation. For example, an automobile may enter an interstate highway, requiring a transition to an **Interstate-Driving** Major Context. Transitions between contexts are triggered by events in the environment – some planned, others unplanned. Expert performers are able to recognize and identify the transition triggers quickly and effectively. Rules that monitor the environment for those triggers are called Sentinel Rules.

CxBR is a very intuitive, efficient and effective representation technique for human behavior. For one, CxBR was specifically designed to model tactical human behavior. As such, it provides the important hierarchical organization of contexts. A full description of CxBR can be found in [3].

4. Genetic Programming

Genetic Programming (GP) is derived from Genetic Algorithms. Both are stochastic search algorithms. The search process looks for the best suitable program that will solve the problem at hand. The target system for the GP could be a CPU, a compiler, a simulation or anything else that could execute the pre-defined instructions, from now on referred to as a program. GP evolves source code representing a program that can

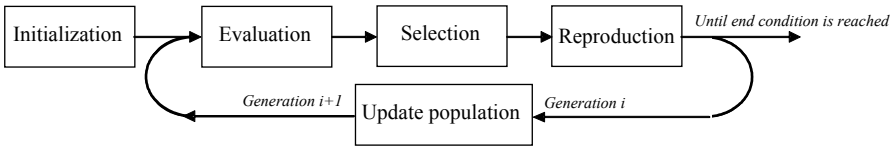

Figure 1: Evolution of GP

address a specific problem. This makes it very suitable for use together with CxBR. GP can build complete software programs that support the internal construction of the CxBR (i.e. the context-base).

Evolving a program with GP is described in Figure 1. To make GP work, some basic requirements must be satisfied. First, we need to have a set of individuals (i.e. programs that represent different solutions to the problem). Furthermore, all the individuals need to be evaluated in some manner as to what degree they are able to solve the problem. The features of the individuals with better suitability would preferably be preserved and survive, or breed new individuals to the next generation. The next GP step would be to evolve the individuals (i.e. reproduction) in some manner to preserve the "good" features and develop even better individuals. The most common genetic operators are crossover and mutation. They support the development and evolution of the individuals.

The criteria for stopping the evolutionary process can be a maximum number of evaluations made, a maximum number of generations evolved, the fitness reaching a certain level, or other measurable criteria given. When the evolutionary process is finished in GP, there then exists a program that will solve the problem to a certain degree.

Since GP builds source code, it could be used to incorporate knowledge in any context level or in any instances within CxBR where human behavior is encoded. This means that we could choose to implement learning in any specific part of CxBR and construct the knowledge therein.

5. Towards automation

Human behavior within CxBR can be categorized in two groups: action rules and sentinel rules. Rules, in this case describes knowledge containers. These containers can contain production rules, functions, operators and complex data structures. The action rules describe the action of the agent within the specific context. At the Major Context level, the action set tends to be more a collection of Sub Contexts and less of other functions, rules or variables. At lower context levels, the action set is less composed of Sub Contexts and at the lowest context level, there are no Sub Context calls. At the level of any Context, the sentinel rules determine whether their own Context will be active for that specific context level. The structure of these sentinel rules are similar in all contexts, at all context levels. Each context has its own set of sentinel rules that determine whether this context should still remain active or if it should turn over the control to another context at the same level or at a higher level. This can be viewed as state transition rules where each state (i.e. context) has its own transition table. Experience has shown that three or at most four, levels of contexts, including the Mission Context, are normally sufficient.

From the discussion above, we can conclude that if we want to incorporate learning into CxBR, the learning paradigm must be able to learn the applicable behavior in a specific context (i.e. action rules) and also the appropriate context switches (i.e. sentinel rules). The type of learning in those two knowledge containers is usually quite different. To learn specific action patterns, the learning regularly becomes somewhat of a regression problem where the model is trying to minimize the discrepancies to the human's performance. When it comes to choose the right context for the current situation, the learning process is more of a classification problem. A learning paradigm that can adapt well to those different learning problems is GP.

Instead of creating the contexts by hand, we use the GP process to build the knowledge within the contexts. The GP's evolutionary process provides the CxBR frame with appropriate context's action rules and sentinel rules. This new approach to model human behavior is called Genetic Context Learning (GenCL). The individuals in the genetic population are components of the context base and a simulator is used to simulate an individual's behavior. The behavior from the simulator is then compared with the human performance, and a fitness measure is established to evaluate the models appropriateness (see Figure 2).

The evolutionary process will strive to minimize the discrepancies between the performances of the contexts created by GP and the human performance. The human data is the observed human performance, or rather appropriate parts of the performance selected by the observer module. The features of CxBR and GP show that their combination could be a feasible approach to learning by observation. Through this synergistic combination, a system could be constructed that, by observing a human, is able to build a context base for simulated entities that exhibit human behaviors.

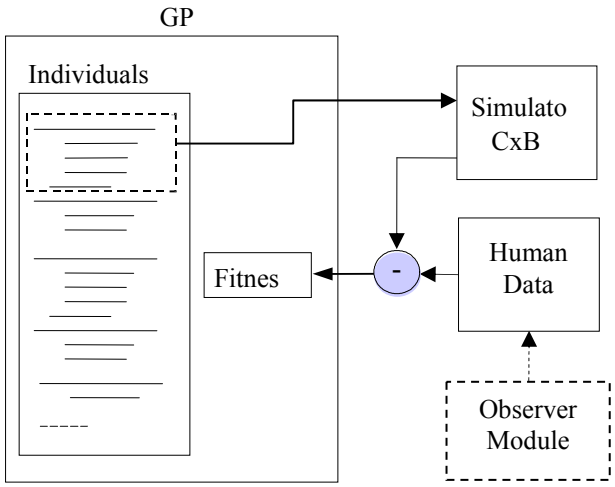

Figure 2: Genetic Context Learning

The Observer Module in Figure 2 has not yet been implemented. In the results described later the objective was to investigate the feasibility of integrating CxBR and GP to achieve our stated objectives. The intended use of the observational module is to select appropriate data to be used in the learning process. Since the computational task was too complex to employ the complete data set this has been done manually so far.

A positive feature of using GP as the learning algorithm is that it preserves many of the features of CxBR that makes it a good paradigm to model human behavior. The set of tools GP uses to store the knowledge learned is the same tools as a programmer or knowledge engineer would have used in developing the knowledge base - source code statements. The agents are able to express their action in a way that is understandable to humans. This implies that the knowledge is interpretable if we want to include communicative features within the agents. It is easy to interpret source code and convert it to written language.

6. Experiments

The first experiments with this new approach to learning by observation were conducted to model human car driving behaviors. These models were designed to store the knowledge in contexts and be applied to simulated agents. Five different drivers were used to drive a commercial driving simulator in city traffic. Training data was collected from 30 minutes of driving from each driver in a realistic environment. The drivers returned to the driving simulator four months later and data were collected for validation purposes. The objective was to determine whether the new approach could model the drivers' behavior during normal operating conditions merely from its observation, i.e., learning by observation. Hence, no single scenario was repeated during the runs. The aim of these first experiments was to test the validity of the learning paradigm that combines CxBR and GP. The experiments were organized so that 1) the knowledge within the different context (i.e. action knowledge) and 2) the knowledge that determines the appropriate context to activate (i.e. situational awareness) were learned. If the model should cover the basic city driving, it must be able to handle traffic lights and intersections, besides normal driving on a straight road segment. The prerequisite for the learning is described in Figure 3.

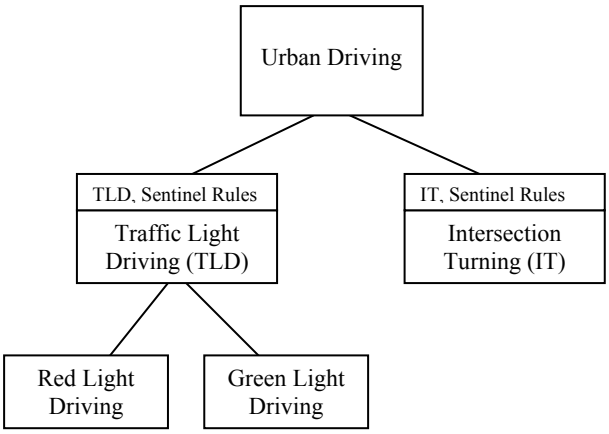

Figure 3: The initial empty Context base

This context base will initially be empty and contain no knowledge. Even though, the emty context base will give the GP a structure to evolve knowledge within. The action within the context **Urban-Driving**, the Sub-Contexts **Traffic-Light-Driving** and **Intersection-Turning** and the Sub-Sub-Contexts **Red-Light-Driving** and **Green-Light-Driving** need to be evolved by the GP algorithm. Additionally, the rules controlling the activation of contexts (i.e. Sentinel

Rules) need to be evolved by the GP. The environment for the experiments was set up to ensure that the behavioral patterns of the drivers were neither predictable nor trivial. An example of this unpredictable behavior is when a traffic light changes from green to yellow and then to red. If this change takes place at an appropriate distance from the car, the drivers will make a decision on whether to stop when the light turns yellow or if they continue and pass the light while it still is yellow. The distance to trigger this diverse behavior among people seemed to be when the car is 30 meter prior to the light when driving in city traffic. The aim of these experiments was to show the validity of this new approach to automatically build context knowledge from observation. Hence, the aim is not to build the perfect model of a driver but to model an individual driver. So, five different driver models were built. If one of the drivers was a poor driver, there will be a model created of a poor driver. In other words, the experiments succeed when the deviation between the driver observed and his particular model is very small.

To make the learning feasible, the data was reduced from 5000 usable samples per driver to less than 350 samples. These samples need to be complete enough to represent all possible situations within city driving with different state changes of traffic light, intersections and normal city driving. The training data consist of the observed driver's speed, distance to an intersection or a traffic light, two Boolean variables indicating the presence of intersections and traffic lights and the state of the traffic light if it is present.

The data were partitioned to contain similar amount of data from the different scenarios to be used in training. This was done to ensure that the search pressure would not favor any particular situation. The data points were selected randomly from some typical scenarios (e.g. within 100 meters before a traffic light or an intersection). The scenarios are further selected to complete the behavior in the context to be learned. As an example, if Traffic-Light-Driving is to be learned, data from several different scenarios (e.g. stopping at light turning red, running yellow lights, passing traffic lights when making an intersection turn, etc.) need to be included in the training data. The data points for each scenario were recorded at a constant rate (e.g. 0.4 seconds) so the algorithm knows when to stop the simulation and compare the individual's performance with the human's performance.

Note that the agent is unaware of all traffic rules and regulations and the only way of learning is to mimic the driver's behavior. As an example, the agent is never told to stop at a red light but it will learn this by observing the driver's behavior. A complete description of the experiments, the data used and the learning algorithm can be found in Fernlund [3].

7. Results

To evaluate the results from the experiments, four different evaluation criteria were defined:

- Learning capabilities
- Generalization
- Long term reliability
- Utility test

The first set of tests, *Learning capabilities*, simply measures how well the agent has learned the driver's behavior. This is simply the deviation between the agents' and the drivers' output when fed with the same data used during the training of the agents.

In this research we evolved five autonomous agents. Therefore, it is necessary to test their ability to perform in normal environment (i.e. running in a simulation). It is not sufficient to only test input vectors and compare them with the anticipated output vectors since this will not tell us much about the agents' accumulated errors and their long term reliability. We need to ensure that the agents are autonomous and that the agents' performances in this simulated environment actually are comparable to the drivers' performances even after minutes, hours and days. Except for the Learning Capabilities evaluation, the results are gathered when the agents operate autonomously within the simulated environment. At each simulator cycle, the performances of the agents are compared to the behavior of the drivers at the same position. The deviation in performance could primarily be measured in speed and time deviations. Since the comparison is made at specific locations, it would have taken the agent and the driver some time to get there from the start of the simulation. Hence, there will always be a time deviation between the agent and the corresponding driver at a certain location in the simulation environment, making time an inadequate standard for comparison.

Generalization measures how well the agent can handle new situations not seen in the training data. The agent operates in the simulated environment as described above and is compared with the recorded driver's performance. In this case, however, the driver's recorded performance was not seen by the agent during its training phase.

During *Long term reliability* the agent is tested in a variety of situations during a long term scenario to examine if its behavior is consistent and stable.

The *Utility test* compared the performance of two of the agents built automatically through GenCL against two manually developed agents. The comparison was related to fidelity, not to development time or effort.

7.1 Learning Capabilities

When testing learning capabilities of our evolved agents the same data used during training was fed into the learned model and the output is then compared to that of the corresponding driver (See Table 1).

	Speed deviation [km/h]		Speed
	RMS	Std. Dev.	Correlation
Driver A/Agent A	1.92	2.29	0.988
Driver B/Agent B	2.03	3.07	0.983
Driver C/Agent C	1.85	1.79	0.990
Driver D/Agent D	1.69	1.92	0.989
Driver E/Agent E	3.81	7.84	0.852

Table 1: Training error

In order to determine how well the agents learned their tasks, inputs were presented to the agents and their corresponding outputs were compared to the actual drivers' cars (i.e. the agent is treated as a black-box). The output from the car and the agent is the speed at a specific moment. The data samples used in this test are approximately 150 samples for each agent from the training data. Since no simulation takes place, there will be no discrepancies in time or distance and the only output variable is speed. The speed deviation is low and the correlation is high. A high correlation, close to one, means that the agent increases and lowers its speed in the same manner as the real driver. Hence, GenCL show good learning capabilities.

7.2 Generalization

To measure generalization, the agent is put to operate in the simulated environment and compare its performance with the recorded driver's behavior in the situation the driver experienced during the validation run. The validation environment refers to the new environment the drivers experienced during their second simulator run. During the drive, the agent will pass five traffic lights that change from green to red and one that changes from red to green.

	L 1	L 3	L 4	L 5	L 6	L 7
Driver A /Agent A	R/R	OK	S/S	R/R	R/R	S/R
Driver B /Agent B	R/R	OK	S/S	R/R	R/R	R/R
Driver C /Agent C	S/S	OK	S/S	S/S	S/S	S/S
Driver D /Agent D	R/R	OK	S/S	R/R	S/R	S/R
Driver E /Agent E	R/R	OK	S/S	S/R	S/R	S/R

Table 2: Qualitative comparison of the drivers / agents performance. S stands for stop and R for running the light while it's still yellow. OK indicates that the agent performs in accordance to the driver at the light turning green

First the qualitative behavior at the different traffic lights was examined. Table 2 shows the agents behavior at the traffic lights. Here it shows that agent A runs light 7 while driver A stops at that light. Agents B and C performs exactly as drivers B and C, respectively. Agent D, however, runs both light 6 and light 7 but driver D actually stops at those two lights. If we look at agent E and driver E, we can see that the behavior often differs at the lights. Driver E actually performs very differently at the two simulator runs. In the first run he was very reckless and ran more yellow lights than the other drivers. On the other hand, while in the validation environment, he was very careful and stopped at almost every light.

	Speed dev. [km/h]		Time dev. [s]		Speed
	RMS	Std.Dev	RMS	Std.Dev	Correlation
Agent A	7.47	7.44	1.47	1.47	0.880
Agent B	7.14	6.19	2.56	1.75	0.896
Agent C	7.12	7.11	3.60	2.80	0.926
Agent D	10.5	9.23	9.10	6.78	0.712
Agent E	17.0	12.0	38.4	30.3	0.550

Table 3. Speed and time deviation during the validation testing

If we look at the speed and time deviations of the validation run in table 3, the A, B and C agents performs very well. However, a slightly worse performance can be observed from agent D. Agent E, on the other hand, is not performing well at all. That agent E is not performing well in comparison to driver E is easy explained by the irregular behavior of driver E between the original training run and the subsequent simulation run. When the stopping behavior at a traffic light differs between the driver and the agent, the time deviations will be affected rather dramatically after the light, since the time for the light to turn green will be added to the time deviation. Speed correlation will also

be affected since differences at those specific traffic lights will indicate negative correlations.

The results for agent A are good even when its misbehavior at one of the traffic lights is kept in the comparison. The misbehavior of agent D is more interesting to analyze since agent D's behavior is not good. An analysis of the training data shows that there was a lack of richness in D's training data. It shows that driver D actually stops at all lights abut to turn red if he is going to make a turn in the intersection where the light is located. By inspecting the code of the evolved agent D it shows that the agent found this relationship and will only stop if it will make a turn. In the validation data set driver D stopped at light 6 and 7 where he was going straight.

The correlation between the agents' speed and the drivers' speed could provide another quantitative measure of the agents' performance. By comparing the correlation between all the drivers and agents in the validation environment, we can investigate whether the agents have been able to capture driver specific features. Table 4 shows the correlation between the drivers and the agents. Note that the table is not symmetric because each agent operates in the environment and which data to compare to depends on its behavior. This means that the same data is not used when agent x is compared to agent y as when agent y is compared to agent x. If the coefficient is close to one, the correlation between the variables is high. Conversely, if the value is low there is a low correlation between the variables. A specific row in table 4 shows the correlation between a specific agent and the five different drivers' data. The correlation is calculated over the entire test run and is based on a sample set between 1,500 – 2,300 samples.

	Driver A	Driver B	Driver C	Driver D	Driver E
Agent A	**0.879**	0.840	0.831	0.708	0.667
Agent B	0.819	**0.896**	0.711	0.690	0.540
Agent C	0.853	0.644	**0.926**	0.857	0.913
Agent D	0.859	0.853	0.694	0.717	0.602
Agent E	0.794	0.855	0.738	0.675	0.550

Table 4: Speed correlation between the agents and the drivers

To make a fair comparison of the correlation values in table 4, we need to know how the different training data sets correlate to each other and also how the different validation data sets correlate to each other. The only data set without high correlation (above 0.8) to any other set is the data set for driver A, but in the validation set driver A have a high correlation to driver D.

Agents A, B and C show the best correlation to the correct driver while agent D and E correlates better to other drivers. The result becomes even more interesting when the poor performance of agent E (inconsistency of driver E) and agent D (lack of richness in training data) could be explained. All other agents are able to capture the individual behavior of their corresponding driver even if it exist high correlation between the different training sets or the validation sets.

7.3 Long term reliability test

The long term reliability test was conducted to investigate whether the agents exhibit consistent behavior even after substantial amount of time in a simulation run. Given that GP will produce code not accessed during the training phase, and that the use of conditional statements that introduce discontinuities, the test of long term reliability is very important.

Here the five agents were allowed to operate within the simulated environment for 40 minutes, pass more then 60 traffic lights and 25 intersections. Now the agents were exposed to a variety of traffic light scenarios where none were the same as the other. To be able to compare their stability and long term reliability, their behavior was recorded when the traffic light ahead of them was either yellow or red, since this is one of the occurrences where different behavior can be detected.

If an agent was not be stable and invokes **Intersection-Turning** when making a turn, the agent will approach the turn to fast and will actually end up beside the road. If this was the case, the agent would be stuck since no recovery algorithm is implemented for the agent to find a way back to the road. Hence, the fact that all the agents were still running after 40 minutes prove robustness in terms of **Intersection-Turning**.

Since the traffic lights now change their states at different distances (i.e. the lights are time scheduled and not related agents distance) and agents might approach the lights at different speeds, it is difficult to make an exhausted statistical analysis of their behavior. Anyhow, table 5 shows a simple compilation of the agents' behavior when they approach lights that is either yellow or red. Two different events occur: the light turns from green to red or from red to green.

	Light turning Red			Light turning Green
	Stop	Avg.Dist	Std.Dev	Correct behavior
Agent A	20/20	34.7	12.9	20/20
Agent B	22/22	8.04	1.95	22/22
Agent C	25/25	5.89	1.03	8/8
Agent D	31/34	4.50	1.31	6/6
Agent E	22/22	13.5	0.551	11/11

Table 5: Agents' long term behavior

A qualitative measure could be performed of the agents' action when the lights turn red. The stopping column in table 4 shows how many lights the agents stop at, compared to the total number of lights passed turning red. All the agents, except agent D, stop at all lights turning red. Agent D runs three lights when they turn red late (i.e. the light actually turns red before the agent pass the light). Investigating the results more it shows that if the lights turn red when the agent is further away than 27 meters the agent will stop and the occasions where the lights turn red when the agent is closer than 23 meters the agent will run it. Even if the agent some times runs the light, it is consistent and acts the same in similar situations.

As the agents come to a stop at the red lights, a comparison could be made on their different stopping distances. Table 4 show that all the agents except agent A, stop at almost the same distance every time and therefore their standard deviation on the stopping distance is small. Agent A stops at different distances almost every time. The surprising fact is actually that the other four agents manage to generalize so well that they stop at approximately the same distance, even if the time of light change is different. All the training data presented to the agents during learning, lights changed their state when the driver was 30 meter prior to the light. Hence, all the agents stopped at consistently at the same distance during training (approximately thirty meters after the light turns from green to yellow). So, the most obvious is not for the agents to generalize as good as they have, but rather to stop in the vicinity of thirty meters after the light change from green to yellow.

The final observations on the agents' long term behavior are their behavior when approaching traffic lights turning green. Two observations can be made as the agents approaching a red light about to turn green. The first thing that institutes correct behavior of the agents is that they do not stop at the red light when they are far from the light. The other behavior to investigate is that they lower the speed as they get closer and that they pick up speed when the light turn green. The column that describes the correct behavior at a light turning green in table 4 compares the number of correct behaviors to the total numbers of lights turning green exposed to each agent. All the agents show a correct behavior all the time as they approach a red light about to turn green.

This test has shown that the agents show consistent and stable performance throughout the long term stability test. Four of the five agents even perform more consistent traffic light driving than could be expected regarding the training data presented during the learning phase.

7.4 Utility Test

This test aims to prove that the agents developed through automated observation are comparable to those created via the conventional manual means. Drivers C and D were interviewed as SME's by a knowledge engineer to capture their know-how. These drives were also observed in their performance, but this time by a knowledge engineer and not a learning system.

Furthermore, driver performance data was not logged in this experiment. The results show that the evolved agents performed equally well as the engineered agents. Table 6 below depicts this.

	Speed [km/h]		**Time [s]**		**Speed**
	RMS	**Std.Dev**	**RMS**	**Std.Dev**	**Corr.**
Engineered agent C vs. Driver C	7.94	7.81	4.35	4.35	0.894
Evolved agent C vs. Driver C	6.74	6.72	2.10	2.06	0.920
Engineered agent D vs. Driver D	8.83	8.88	9.55	9.01	0.852
Evolved agent D vs. Driver D	8.46	8.45	3.13	3.12	0.842

Table 6: Comparing evolved and engineered agents in the training environment

8. Conclusions

This research has shown that individualized human behavior models can be created automatically from observed human performance. All human behavior knowledge, except the context structure, in the agents created within this research was automatically created by the GenCL algorithm. The tests have also shown that those agents were able to generalize the knowledge and proved a stable performance with individual behavior patterns. Further, the algorithm

produced models competitive with models manually created by a knowledge engineer.

The results presented here have shown the ability of GP to produce knowledge in all the different parts of the CxBR's context base. GP has been able to evolve knowledge in the action rules of the contexts and also the knowledge in the sentinel rules. Using the structure of CxBR also improves the learning capabilities of GP. In this research, the CxBR gave GP a frame in which to conduct the learning.

9. References

[1] Henninger A., Gonzalez A., Gerber W. Georgiopoulos M., DeMara R. (2000) "On the Fidelity of SAFs: Can Performance Data Help?" Proceedings of the Inter-service/Industry Simulation and Education Conference, Orlando, FL, 2000.

[2] Deutsch, S. (1993) "Notes Taken on the Quest for Modeling Skilled Human Behavior" Proceedings of the Third Conference on Computer Generated Forces and Behavioral Representation, Orlando, FL, March 17-19, pp.359-365

[3] Fernlund, H. G, "Evolving Models from Observed Human Performance" Doctoral dissertation, Department of Electrical and Computer Engineering, University of Central Florida, Spring, 2004.

[4] Gonzalez A. J. and Ahlers, R. H. (1998) "Context-Based Representation of Intelligent Behavior in Training Simulations" Transactions of the Society for Computer Simulation International, volume 15, no 4, December 1998

[5] Gonzalez A. J. and Ahlers, R. H. (1994) "A Novel Paradigm for Representing Tactical Knowledge in Intelligent Simulated Opponents" Proceeding of the IAE/AIE Conference, May 31-June 3, 1994, Austin, Texas.

Author Biographies

HANS FERNLUND received his Bachelor degree in Computer engineering from the Dalarna University, Sweden in 1993. In 1995 he obtained his Master of Science in Computer Engineering from the University of Central Florida. Fernlund obtained his Ph.D. from the University of Central Florida in the spring of 2004. He is an assistant professor at the Dalarna University, Sweden.

AVELINO J. GONZALEZ received his Bachelor's and Master's degrees in Electrical Engineering from the University of Miami, in 1973 and 1974 respectively. He obtained his Ph.D. from the University of Pittsburgh in 1979 also in Electrical Engineering. He is a Professor in the School of Electrical Engineering and Computer Science at the University of Central Florida, specializing in Artificial Intelligence and simulation.

SVEN E. EKLUND received his Master of Science in Computer Engineering from Linköping Institute of Technology, Sweden, in 1988. Eklund obtained his Ph.D. from Dalarna University, Sweden in 2004. He is currently an assistant professor at Dalarna University, specializing in efficient hardware architectures for Evolutionary Algorithms.