Ernest *et al*. *Complex Eng Syst* 2023;3:4
DOI: 10.20517/ces.2022.54

Complex Engineering Systems

Research Article

Open Access

Formal verification of Fuzzy-based XAI for Strategic Combat Game

Nicholas Ernest, Timothy Arnett, Zachariah Phillips

Thales Avionics Inc., Cincinnati, OH 45242, USA.

Correspondence to: Dr. Nicholas Ernest, Thales Avionics Inc., 4358 Glendale-Milford Rd., Cincinnati, OH 45242, USA. E-mail: nick.ernest@us.thalesgroup.com

How to cite this article: Ernest N, Arnett T, Phillips Z. Formal verification of Fuzzy-based XAI for Strategic Combat Game. *Complex Eng Syst* 2023;3:4. http://dx.doi.org/10.20517/ces.2022.54

Received: 11 Dec 2022 **First Decision:** 31 Jan 2023 **Revised:** 13 Mar 2023 **Accepted:** 14 Mar 2023 **Published:** 30 Mar 2023

Academic Editor: Hamid Reza Karimi **Copy Editor:** Yin Han **Production Editor:** Yin Han

Abstract

Explainable AI is a topic at the forefront of the field currently for reasons involving human trust in AI, correctness, auditing, knowledge transfer, and regulation. AI that is developed with reinforcement learning (RL) is especially of interest due to the non-transparency of what was learned from the environment. RL AI systems have been shown to be "brittle" with respect to the conditions it can safely operate in, and therefore ways to show correctness regardless of input values are of key interest. One way to show correctness is to verify the system using Formal Methods, known as Formal Verification. These methods are valuable, but costly and difficult to implement, leading most to instead favor other methodologies for verification that may be less rigorous, but more easily implemented. In this work, we show methods for development of an RL AI system for aspects of the strategic combat game *Starcraft 2* that is performant, explainable, and formally verifiable. The resulting system performs very well on example scenarios while retaining explainability of its actions to a human operator or designer. In addition, it is shown to adhere to formal safety specifications about its behavior.

Keywords: Explainable AI, reinforcement learning, formal verification, starcraft, genetic algorithm, fuzzy logic, genetic fuzzy trees, formal methods

1. INTRODUCTION

Artificial intelligence (AI) applications in reinforcement learning (RL) have garnered significant attention in recent years[1,2] due to their wide ranging applicability to previously intractable problems. In particular, the success of DeepMind's AlphaGo system[3] ignited a revitalization of research and attention in the area, specifically with the introduction of new techniques combining RL with deep neural networks (DNN), known as Deep Reinforcement Learning (DRL). However, while improvements within the overarching field of RL and DRL continue to increase the scalability and performance of these approaches, verification and explainability efforts have not received comparable attention. There have been efforts to take highly performant DRL solutions and increase explainability and trustworthiness *ex post facto*. An example of this was DARPA's XAI program, introduced to study and identify the importance and usage of explainability in AI[4]. Their conclusions were that many DRL solutions were brittle, unverifiable, and opaque to human designers/operators that may want to audit, verify, or extract knowledge from what the agent learned.

Fuzzy inference systems (FIS), function approximators that utilize Fuzzy Logic and inference rules to map inputs to outputs[5], have several properties that lend themselves towards XAI, but have other potential drawbacks compared to DNNs, namely scalability. Fuzzy Logic-based systems have long been used in control system development due to their approximation capabilities[6], ease of implementation from expert knowledge[7], robustness to input noise[8], explainability and transparency to humans[9], and ability to be formally verified[10]. However, scalability issues with respect to the number of inputs limited the potential applications. Fuzzy trees were introduced in 2015[11] in order to mitigate scalability issues while also retaining explainability and approximation capabilities by combining multiple FISs arranged in a network or tree-like structure.

Genetic Algorithms are a class of gradient-free search algorithms that evolve solutions by mutation and recombination over a number of generations by evaluating their *fitness* against one or more metrics in a *fitness function*. GAs have long been used to great effect in many areas, but also in a large body of work related to optimization of FIS parameters[12]. Combining Fuzzy Trees with Genetic Algorithms led to Genetic Fuzzy Trees (GFTs)[11], a powerful combination that uses an explainable and formally verifiable function approximator with a gradient-free optimizer and has been applied to several complex use cases in both supervised[13] and reinforcement learning domains[14]. Thales's GFT software toolkit includes both a Fuzzy Logic engine, Psion, and a state-of-the-art Genetic Algorithm-based optimization tool, EVE[15]. Its strengths include ease of utilization for finding low wall-time solutions and wide applicability due to the nature of gradient-free optimization. Perhaps the most notable prior use-case was the Alpha system[14], a super-human AI for beyond visual range air-to-air engagements within high-fidelity simulations against expert human pilots[14].

Another benefit of GFTs is that they can be verified using Formal Methods. Formal Methods are often defined as "mathematically rigorous techniques for the development, specification, and verification of systems." Many methods and techniques fall under the umbrella of Formal Methods including the boolean satisfiability problem (SAT)[16], satisfiability modulo theories (SMT), model checking, theorem proving, reachability analysis, etc. Formal Verification is the utilization of Formal Methods towards verifying the correctness of a system. In general, verification is about confidence in the correctness of a system, and formal verification extends traditional verification methods (e.g., Monte Carlo evaluation) towards definitive proofs of correctness. The adoption of formal verification has been slow in the world of AI and ML mainly due to the difficulty in proving properties of DNNs as their scale continues to increase.

In this work, an AI agent using the GFT construct is created and then trained using reinforcement learning to play specific scenarios within StarCraft 2. Note that this study does not analyze an entire standard Starcraft 2 match. Instead, the focus will be on specific control applications with a focus on explainability and formal verifiability, though an entire standard game of Starcraft 2 could certainly be studied through utilization of the GFT approach. This study is not aimed towards demonstrating a performance disparity between Fuzzy logic-

based AI approaches and any other methodology, but rather to demonstrate how these systems can be created in a way that maintains explainability and formal verifiability. These capabilities are highly desired, and often required, for mission/safety-critical applications. Starcraft 2 is used because it is a commonly used environment in modern RL research, it allows for creation of publicly-shareable mission/safety-critical use-cases, and allows for extension of this work for comparisons with other highly performant RL methodologies.

The GFT is initialized with a structure, given initial parameter values where applicable, and then trained by interaction in the game across a training set of episodes. The structure of the GFT is such that output actions can be explained by extracting the activated rules and membership functions. Specifications about the system's behavior are then created and verified against the system using Formal Methods[17]. In cases where the specifications are violated, counterexamples are returned showing where the violations occur, and then corrections are performed. The corrected systems are then verified to not violate specifications showing definitive correctness with respect to the developed behavioral specifications.

Four specifications have been developed for this study, which is by no means an exhaustive potential set. This work will demonstrate the learning capability to solve a particularly difficult sort of engagement, demonstrate the potential explainability possibilities, and prove adherence to a set of relevant specifications. The primary objective for this study is to showcase an example of a fuzzy logic based AI system which can be formally verified to adhere to safety specifications within a mission/safety-critical scenario.

The structure of the remainder of the paper is as follows. Section 2 details the methods used to create, train, and verify a GFT for specific scenarios in SC2. Section 3 shows the results including RL training, verification against specifications (and generated counterexamples), and results after modification to ensure specification adherence. Section 4 discusses the results in depth and offers thoughts on extensions and future work. Finally, Section 5 gives a concise conclusion on the work, results, and impact of this study.

2. METHODS

This section describes both the general background info and setup of a GFT solution for training in a RL context followed by the specific implementation for the SC2 use case.

2.1. General GFT workflow

The typical workflow for creating a GFT AI solution involves creating a GFT-based agent with a tree structure that is purposefully designed such that it is explainable and formally verifiable. The performance of the agent is evaluated through a number of different scenarios for training, testing, and validation. The system is then formally verified against specifications about its behavior. Explainability aspects are used both as an auditing tool when evaluating counterexamples from the formal verification process, and also as a knowledge transfer device to allow human observation and understanding of what the agent learned during the RL process.

GFTs are most often created using a combination of initial expert knowledge and machine learning using gradient-free optimization. Thales's GFT toolkit is a commercial software package that includes both a Fuzzy Inference System (FIS) engine named Psion and a Genetic Algorithm optimization engine, EVE, which combined represent the best software implementation of Genetic Fuzzy Trees available today[15]. An example of a notional GFT structure is shown in Figure 1.

The GFT is then trained using EVE in whatever context the problem is framed (supervised, reinforcment) until termination criteria are met. These termination criteria can include performance metrics, time, number of generations/epochs, stagnation measures, etc. Once this round of training is complete, verification can occur on the GFT (fixed parameters) along with SME/developer auditing and manual changes. Counterexamples

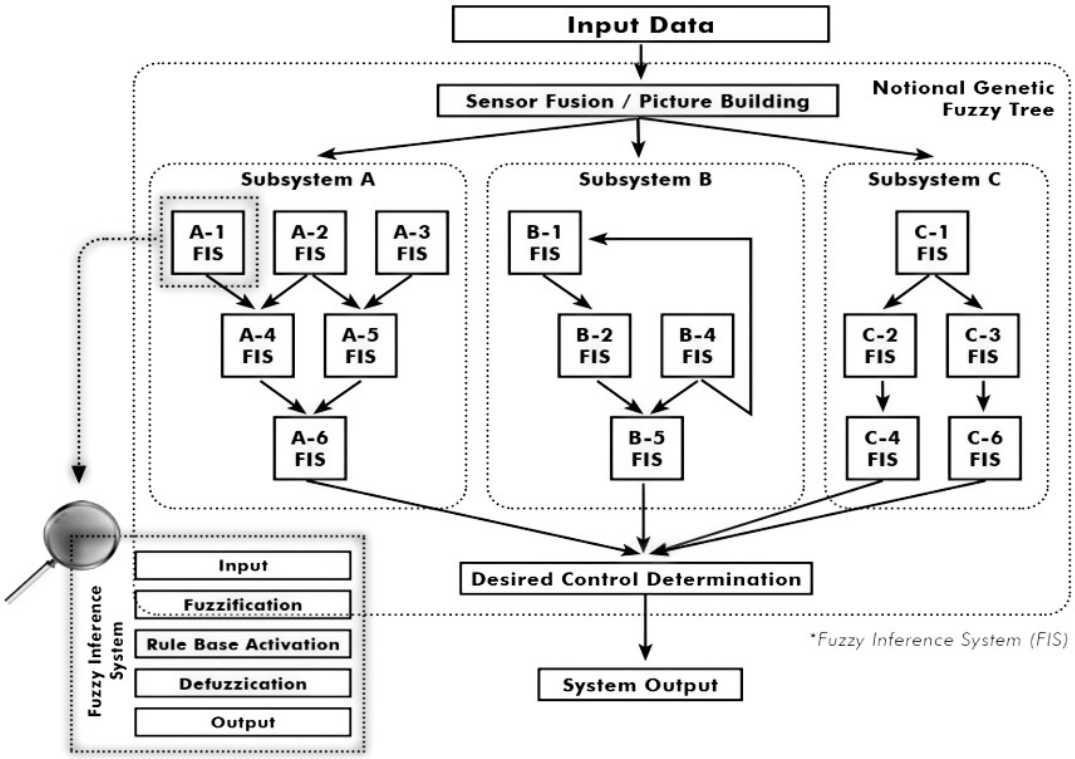

Figure 1. Notional figure of a GFT structure, with a plurality of FIS nodes.

from the verification process can be used to directly change aspects of the GFT, or inform the ML process in the form of constraints, reward modifications, etc. The GFT can then be reinitialized and trained using the ML process again. This process iterates until system requirements are met.

2.1.1. Initial GFT structure

The GFT structure is primarily defined through the input/output flow of each FIS node within the system. Although a GFT structure can be initialized randomly and optimized through EVE, it is often advantageous to initialize using domain knowledge where possible. This is beneficial for two main reasons:

1. It takes advantage of domain knowledge as an initial seeding to use as a starting point. This provides a valuable baseline that can then be expanded/improved upon
2. Depending on methods used, can preserve semantic relationships/importance in the structure itself due to the linguistic nature of FISs and the connections between them in the tree

Note that these are not strictly necessary for explainability or performance, but often lead to shorter project development times and improve transparency/understandability [18]. Assuming a manual structure is provided, the learning agent can be left to optimize all of the Membership Functions (MFs) and Rule-Bases (RBs) of the FISs, either holistically or individually.

2.1.2. GFT RL training

Due to the gradient-free algorithms used for the training process, the methods used for both supervised and reinforcement learning problems are largely the same with the difference being in the formulation of the fitness function. Where in a supervised learning problem the fitness function may be a measure of the fit of the function to the underlying data (LMS, cross entropy, etc.), in RL use cases the fitness is often a measure of how well the agent performed in one or more simulated episodes, or *scenarios*. The resulting fitness score is used

to evolve the population of candidate GFT solutions until termination criteria are met.

After some amount of training, a fixed configuration for the GFT architecture is evaluated, tested, and verified. Additional numerical testing is often performed along with unit tests, formal verification, and human inspection and correction. If counterexamples to formal specifications are found, they are often mitigated through direct designer input, or added to the RL process in such a way to penalize GFT configurations in the GA population that have such violations. This iterative process continues until adherence to all specifications and requirements is achieved and performance metrics are met.

2.1.3. Formal verification

As mentioned in Section 2.1.2, the process for verifying a GFT involves taking a GFT with a particular configuration (fixed structure, parameters, etc.) and evaluating it against formal specifications about its behavior. These specifications are often derived from system requirements, but can also be created from sources such as numeral/simulation-based test results, rules of engagement or other high level behavioral constraints, certification standard criteria, etc. The specifications are then translated from natural language to a formal language using mathematical logic in order to be encoded for formal methods analysis. The formal languages used depend on the necessary expressiveness of the language based on the specification itself. Some of these languages include Propositional Logic, First Order Logic (FOL), and temporal logics such as Linear Temporal Logic, Computational Tree Logic, etc. Many others could be used depending on the use case and formulation of the specifications and models, and Thales GFTs have been implemented with a number of these methods, most commonly FOL and LTL.

Models of the GFT are then translated to formal verification tools such as SMT solvers and model checkers, and then checked against the specifications. If counterexamples are found, modification of the GFT is performed by the designer and/or the ML process. The tools used in this work are the state-of-the-art SMT solver Z3 [19] along with the infinite state model checker jKIND [20].

2.2. Starcraft 2 GFT development

The development process for the SC2 use case is largely the same as the general process described in Section 2.1. An initial GFT is constructed for parts of agent control based on expert knowledge, it is trained using gradient-free RL in the SC2 environment, formal verification is performed and modifications are made based on counterexamples generated, and then iterated until a performant, specification compliant GFT is found.

2.2.1. StarCraft 2 information

As this study is focusing on specific scenarios for demonstrating the explainability and formal verifiability of this capability, we only consider four different unit types within this context:

- **Marine:** A basic human infantry unit which can fire a ranged volley at a certain frequency. Has an advanced technique Starcraft 2 players can utilize in which if they perform rapid movement commands between volleys, they can both re-position slightly as well as fire faster, outputting more damage per second.
- **Medivac:** A human air vehicle which can utilize a limited resource pool to support ground biological units, such as marines, by healing their durability at a set amount per second. Can also be utilized to transport units, though this capability will not be utilized within the context of this study.
- **Siege Tank:** A human ground vehicle, which can move and attack ground units normally. In ”siege mode” the tank becomes stationary, but can attack further and do increased damage per shot, but cannot attack units within a short range of it. Most interestingly for this study, this unit is one of the few that does ”splash damage” that includes friendly units. That is, friendly units near the target will take damage within a certain range. As such, the siege tanks within this study will always be in ”siege mode”.
- **Zergling:** An alien (Zerg) insect unit that is a fast melee attacker. Can not attack air units but can quickly overwhelm ground units. Will serve as the hostile unit type for this study.

Figure 2. Primary scenario to be utilized for training and analysis. Consists of a stronger zergling force within close range of the human force. Pictured (1) siege tank, (2) medivac, (3) marines, (4) zerglings.

A primary training scenario has been developed which will be utilized for reinforcement learning for a difficult engagement made up of these units. A plurality of other engagements have been developed for testing and formal verification.

From a raw performance perspective, the mission shown in Figure 2 is the primary performance objective to complete. This mission has 16 zerglings that make up the hostile forces as well as 6 marines, 1 medivac, and 1 siege tank on the friendly team.

This mission is such that if the in-game controllers for both forces behave natively, the human ground forces lose with on average at least 7 of the zerglings still alive. This mission is feasible for a human to complete with some forces remaining, but is very difficult and requires advanced tactics within the game. Despite expertise in the game, the standard ending with manual full focus on controlling the human forces has notable losses.

2.2.2. Tree structure creation

A GFT has been created to control these 3 specific types of human units, both individually or as an entire force. The general approach for these sorts of control problems is to generate an entire action plan each time step of the environment. As such, the GFT utilized within this study will provide significantly higher action throughput than what would be maintained by a human.

Through subject matter expertise within this type of engagement, we understand that there are a few key control decisions that need to be made each time step:

- **The Medivac** should make healing decisions that efficiently utilize its resources and keeps biological units alive as best as possible.
- **The Marines** must be intelligent in how they spread their fire against the incoming group of targets, focusing fire to eliminate enemy units efficiently.
- **The Marines** optimally will utilize the ”stutter step” strategy, both to increase their effective attack speed as well as to potentially minimize the total incoming damage from the enemy melee units.
- **The Siege Tank** is a powerful unit and should attempt to get as many effective shots off, while minimizing the harm it does to friendly units.

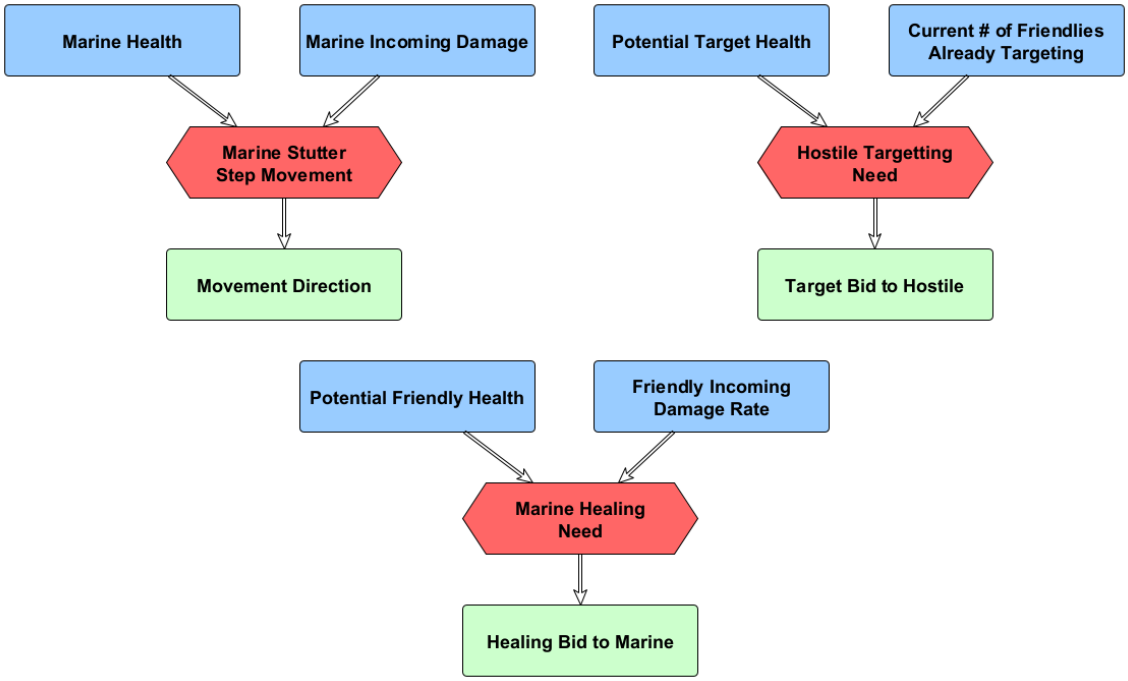

Figure 3. Three standalone FISs for Marine Movement Control, Marine Firing Control, and Medivac Healing Control utilized within the studied model. Normalized inputs in blue, FISs in red, and normalized outputs in green.

A relatively simple FIS structure can accommodate these considerations, with all but the Siege Tank control defined as an output of a single FIS with minimal additional algorithmic work. In Figure 3 these individual FIS nodes are displayed, each being 2-input Mamdani FISs[21].

Specifically, the Marine Stutter Step Movement control will define a relative movement direction to threats based upon both the marines current health, as well as its incoming damage rate from hostiles. The Marine Firing control has a FIS which outputs a bid, from 0% to 100% to select a certain hostile target at every time a shot is available. The potential hostile target with the highest bid is then selected for the marine to attack. This bidding FIS takes in the normalized health of a potential target, as well as the relative quantity of already assigned attacks against this target. The medivac Healing control utilizes a similar approach, considering each friendly unit it can heal and selecting to heal the unit with its highest priority. This priority being determined based upon the normalized health of each marine, and its relative incoming damage rate.

The control for the siege tank's firing logic is more complex and therefore is made up of 3 FIS nodes, shown in Figure 4. The approach utilized for this structure is to make our decisions considering how effective a shot would be against a selected unit, as well as how safe that shot would be to nearby friendly units. Effectiveness is determined by two main items; zerglings are low health units and the siege tank can easily waste some of its damage potential by targeting a very damaged zergling. However, if the very damaged potential target is in the middle of a group of hostiles, selecting that target may still be ideal due to the splash damage. Shot safety is essentially the opposite, now considering splash damage a shot would do to any friendlies, combined with the lowest health of the friendly units that would be affected by said splash damage. A differentiation between the marines' fire control is that the marines should always attack as often as they can in this engagement due to the fact they only damage their hostile target. The fire control for the Siege Tank will select the hostile target to fire at among all hostiles it considers, but only if that maximum fire bid is above 50%, otherwise it would opt to not fire.

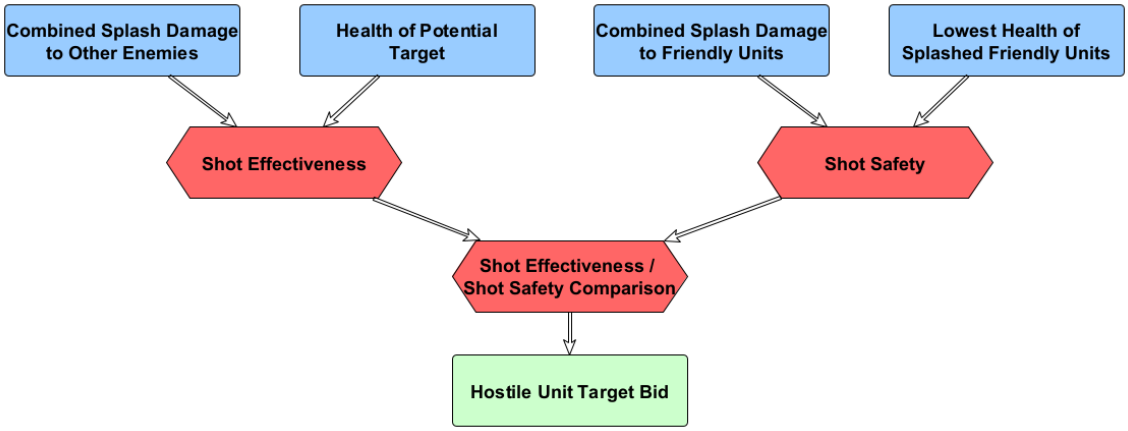

Figure 4. 3-FIS tree for Tank Firing Control utilized within the studied model.

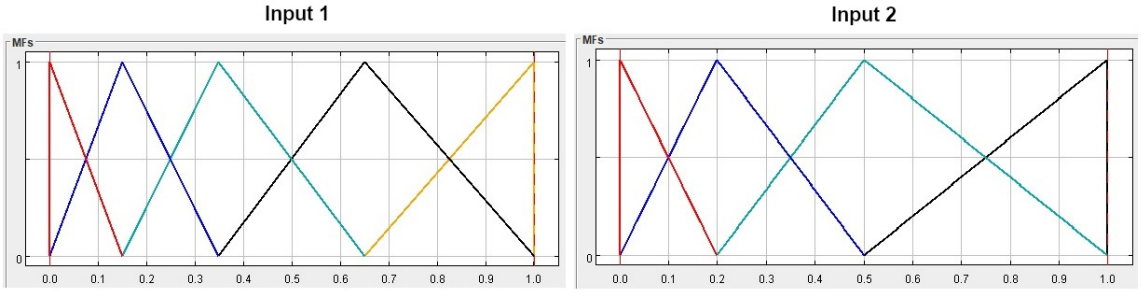

Figure 5. MFs for Marine Firing FIS. Input 1 corresponds to normalized potential target health with 5 MFs for "Very Low, Low, Moderate, High, Very High". Input 2 corresponds to current number of assigned friendly attackers already through this bidding cycle with 4 MFs for "None, Some, Many, All".

2.2.3. Explainability

After creating the tree structure, individual FISs are constructed using expert knowledge where available in most cases. Often this will be in the form of the number of MFs which will be utilized in each FIS; often leaving the specific distribution of said MFs across the input space to the machine learning process.

Explainability can be analyzed through a plethora of manners; the structure of the FISs created for this study were designed to optimize the general explainability and interpretability of the system alongside raw performance. As an example, the input MF sets for the Marine Firing FIS are shown in Figure 5.

A variety of approaches can be utilized to interpret the membership functions and provide explainability, but one of the simplest approaches would be to report or display the most active membership function for each input, corresponding with its rules. This represents the membership function with the largest impact on the output. This can essentially be compared to Shapley Values, a popular explainability metric utilized across multiple RL approaches[22,23]. In general approaches, these values enable a determination of the most impactful variables leading to a particular output. Through the membership functions of a given FIS, this is innately provided. Due to the fact that all system variables have the potential to be human understandable terms within a fuzzy based system such as the GFT, there is further intrinsic value within this approach if the GFT is properly designed. Note that explainability is not inherently granted through the utilization of fuzzy systems; rather fuzzy logic is a construct in which systems that are highly explainable can be developed.

For example, an input combination of 0.04 normalized target health, and 0.01 assigned attackers with its resultant output can be examined, and an explanation structure can be generated from membership function and rule labels: "Bid output is Very High because target health is Very Low and assigned attackers is None". For more extensive fuzzy tree structures, this explanation can be repeated across subsequent cascaded FISs allowing for the creation of a linguistic explanation of the entire decision process. This form of explainability and transparency will be heavily utilized during the formal verification process as manual corrections to the post-training model will be performed if any specifications are found to not be adhered to via formal methods. This requires direct changing of the code of the model at all levels, not just the input or output layers. Holistic understanding of any modifications made throughout this process are critical for any potential deployment of the system post-modification.

2.2.4. Reinforcement learning

The standard RL process for a GFT is to first create a portfolio of training scenarios that each individual in the GA population is evaluated over. This model was created through utilization of an open source Python package for interfacing with SC2 such that constructive runs through these scenarios is possible[24]. Within this study a single mission portfolio is utilized to highlight the formal verification processes, but for most applications a portfolio containing multiple holistic scenarios as well as specific training sub-problems would be included[11,14]

The manner in which the performance will be evaluated must also be defined through the requisite Fitness Function for the GA. The fitness function utilized within this study is found in Equation 1.

$$Fitness = (MarinesAlive * 25.0) + FriendlyHPRemaining - HostileHPRemaining - (Timesteps/100.0) \quad (1)$$

The magnitude range of this fitness function is not critical within EVE, but rather the ability for the evolutionary process to differentiate relative fitnesses between potential solutions, or chromosomes, in a manner that thoroughly rewards good behavior and punishes bad. With this example, the terms specifically are a flat 50 point reward for every marine alive at the end of the scenario. This is then added to the summation of the total friendly health remaining, including that of the siege tank, which has a notably higher health pool than marines. This is subtracted from the hostile force health pool remaining. Finally, there is a slight penalty for the number of timesteps it takes to complete the scenario, as if all other parameters have reached optimality, ideally the solution executes quickly in case additional threats would be inbound to the force. This function is able to be iterated over in future work, but serves as a good basis for the GA to evolve the population of chromosomes.

There is an additional complexity with this particular problem due to the nature of Starcraft 2 and this training setup; non-deterministic fitness evaluations. As the fitness value given to any chromosome within the population will drastically affect both its probability for breeding as well as the relative worth of potential future chromosomes, the ability for a good chromosome to be "unlucky" and a bad one to be "lucky" during their respective evaluations can be damaging to the effectiveness of an evolutionary system. There are mitigating methods, such as evaluating the scenarios, or portions of them, multiple times. Within this study, each chromosome will be evaluated a total of three times, with the worst fitness of those three being the actual fitness that is assigned, to easily mitigate the worst case risk at the expense of computational efficiency of each generation's evaluation.

2.2.5. Specification identification and development

One of the most difficult, and important, aspects of formal verification is specification identification and development. Most often, these specifications come directly from system requirements. However, they can also come from other sources such as certification criteria and standards, external regulations such as Rules of Engagement (RoE), and simulation and testing results. In this work, the specifications come from expert knowledge on desired system behavior with respect to performance and safety of actions taken by the AI. Specifically, that the AI will not take certain actions if unacceptable damage or losses will occur to friendly forces.

Different types of specifications are created based on the FISs that are affected, the needed expressiveness of formal languages in which they will be translated, and the tools available for verifying the systems. For this study, four safety specifications were developed. Safety, in this context, refers to behavior such that *nothing bad ever happens*, where "bad" is defined as an undesirable set of system states. Some would help guarantee not only safe, but also ideal behavior from a performance perspective. However, other safety specifications solely focus on desired safety qualities at the expense of raw fitness performance.

These specifications, in natural language, are as follows:

- **Medivac Healing Spec:** If a marine's health is very high and the marine is not under attack, its corresponding healing bid must be very low;
- **Marine Firing Spec:** If a separate friendly marine has already bid to attack a target whose health is very low, than a marine's fire bid on that target must be very low;
- **Tank Firing Spec:** If a friendly unit would take splash damage from a tank shot and the lowest health friendly in splash range is between very low and low health, only fire at the target if it would do very high damage to the enemy;
- **Tank Conservative Firing Spec:** Never cause splash damage to any friendly unit, regardless of their health.

Through setting the values to the corresponding terms utilized within these specifications to that of our MFs, we can quite simply apply these specifications to a trained Fuzzy Tree through the formal verification tools with the Psion fuzzy logic package. If a counterexample is found where the specification does not hold, typically the designer must either change the spec, or as we will limit this study to, modify the MFs and/or RBs of the associated FISs.

3. RESULTS

3.1. Reinforcement learning

EVE was instantiated to have a relatively small population, 60 chromosomes, each defining a set of MFs and RBs for all seven FISs. The training process was run for 50 generations, each chromosome being evaluated over three scenarios per generation. Thus, this system was trained over 9,000 Starcraft 2 engagements, which was sufficient in this case to reach high performance.

Presented in Figure 6 is a plot showing the best, worst, and mean chromosome fitness within the population of 60 at each generation. Additionally, the running best chromosome's fitness for each generation is displayed. Generally with the fitness function utilized in this system, if the chromosome successfully completed its mission, it would score at least quite close to a positive number. A score of 389.49 was found for the best chromosome thus far. The breakdown of this fitness value is shown in Table 1.

EVE has a heavy focus on maintaining diversity, and therefore especially with a generally smaller population, it is not surprising to see that during each generation the worst chromosome was not able to successfully complete this difficult mission. The distribution between best, worst, and mean is at least one sign of a healthily diverse population throughout the generations. This can help prevent stagnation and ensure continued improvements.

Table 1. Individual Components that made up best chromosome fitness of 389.49.

Fitness Component	Value
Marines Alive (x25)	100.0
Total Friendly Health Remaining	292.0
Total Hostile Health Remaining	0.0
Timestep Penalty (-Timestep/100.0)	-2.51
Total	389.49

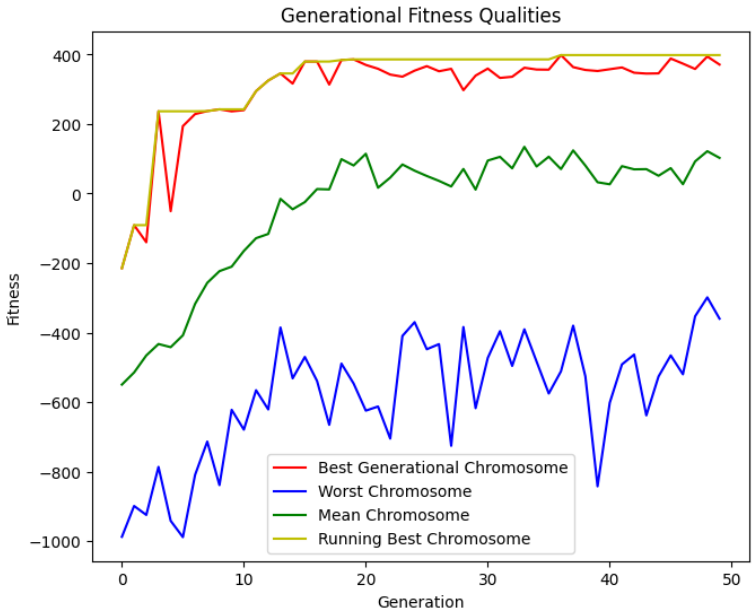

Figure 6. Generational and cumulative fitness values of Genetic Algorithm population.

Figure 7. Frame captures 1 (Left) and 2 (Right) of the mission depicting the evaluation of the best chromosome.

In Figure 7 there are 2 frames of the performance displayed. It is a very active mission, with over 70 actions per second executed by this particular AI model, equating to around 4,200 actions per minute for the roughly 5-9 seconds this mission will last. Obviously this is quite higher than even expert humans would be able to maintain, but again following humanistic restrictions is not a focus of this particular study. There is non-determinism, but the problem typically ends with a few of the marines having been defeated, in this case there were two defeated marines.

Additional training and and increasing the size of the training portfolio would likely lead to greater performance, but for the purposes of this formal verification study, the resultant chromosome is sufficient. It is high

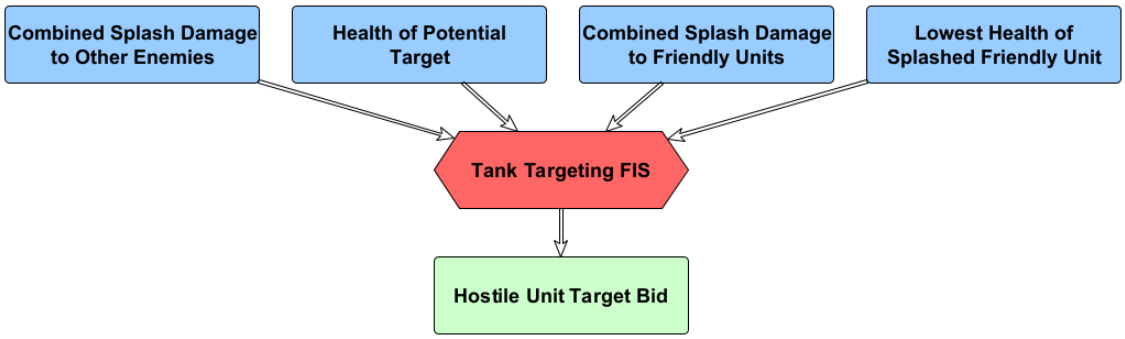

Figure 8. Single FIS variant for siege tank control.

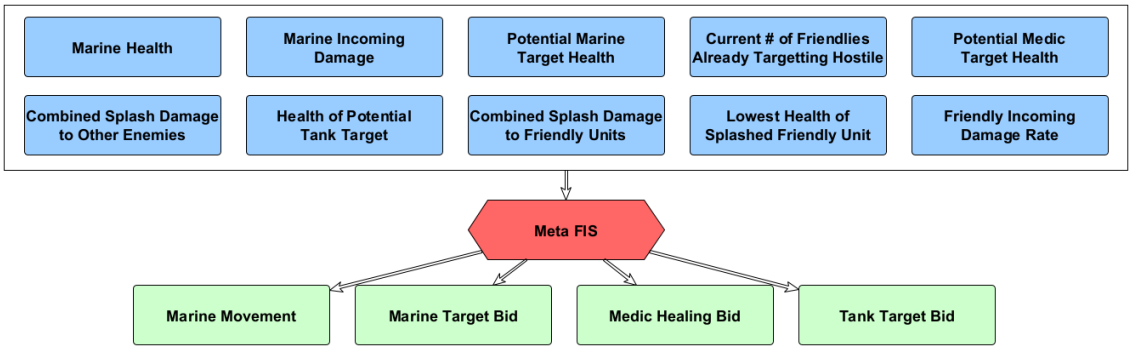

Figure 9. Single FIS variant for all control actions.

performance but will likely have some edge cases that violate the specifications.

3.1.1 Reinforcement learning comparison

Two additional fuzzy systems were developed for the same scenario as a learning performance comparison to the above system. The first comparative system utilized is quite similar to the study system, with the sole difference being that instead of utilizing three separate FISs for control of the siege tank's targeting, this portion of the AI is combined into a single 4-input FIS. The revised diagram is seen in Figure 8.

As five MFs are utilized per input, a full fuzzy rule-base of the original system for siege tank control utilized within this study has 75 rules, 25 rules for each of the three FISs seen in Figure 4. This modified variant has 5^4, or 625, total rules. This change in search space size is significant in terms of total chromosome size, but chromosomes of this size are anticipated to be well within the reasonable scope of the genetic algorithm utilized herein.

Note that the performance of the original system is bounded above by the variant in Figure 8, as any resultant control surface of the original system can be developed within this variant, as can others. The same is true compared to both of these systems for a second variant in which the entirety of the fuzzy AI is encompassed by a single 10-input, 4-output FIS. While the performance ceiling of the system shown in Figure 9 is the same, or higher, as the other variants, the anticipated performance through the same RL process is significantly worse due to an extremely large fuzzy rule-base.

While the original system utilizes a total of 150 fuzzy rules, the single-FIS variant would have $4 * 5^{10}$ or

39,062,500 rules. During the RL comparison process, the compute system utilized however had 32 GB of RAM, an insufficient amount to perform with a chromosome defining a rule-base of this magnitude. However, if this was reduced to 4 MFs per input, making the rule-base contain $4 * 4^{10}$, or 4,194,304 total rules, then 32 GB of RAM was sufficient.

The RL process for these two variants was exactly the same as before, with a total of 60 chromosomes trained over 50 generations. The maximum fitness found by generation for each of the systems is displayed in Figure 10. As anticipated, the 10-input 4-output variant had significantly worse performance than the original system, though improvements were able tobe made throughout the RL process. Though the performance ceiling of the original system is again bounded above by this variant's, the amount of RL that would be necessary to surpass the original's performance is likely prohibitive.

Alternatively, the performance of the variant that solely combined the siege tank's control into a single FIS performed favorably compared to the study's original system. As the genetic process is non-deterministic, and the total number of fuzzy rules is still relatively small, the fact that the initial generation's performance was superior than the original study's is well within expected range. While additional runs of the RL process would create different results, the increased granularity of this variant as compared to the study's original system can be explained through the increase in the search space size. For the majority of this small, 50-generation process, the original system has a higher maximum fitness, though the variant does ultimately end with a higher performing fitness by the end of the run.

While this is a very small GFT system, this comparison demonstrates trade-off comparisons between increased number of FISs (and thus, less overall rules) and the potential effect on RL performance. Though the single-FIS for siege tank control variant performed slightly better, it would be more difficult to formally verify, and overall has a more complex challenge for explainability as compared to the original system. For the remainder of the study, the original GFT architecture will be utilized.

3.2. Formal verification

3.2.1 Safety specification 1

Again, our first specification, the Medivac Healing Spec, states "If a marine's health is very high and the marine is not under attack, its corresponding healing bid must be very low". Although simple, adherence to this specification is desired in all cases. We can represent this in a numerical form by utilizing the points of our MFs with maximum value for each linguistic term, as shown in Equation 2. This is then translated to Linear Temporal Logic (LTL) as shown in Equation 3.

$$\begin{aligned} Spec1 = &\text{If } (MarineHealth > 0.75) \text{ and } (MarineDamageRate = 0.0) \\ &\text{then } (HealBid < 0.25) \end{aligned} \tag{2}$$

$$\begin{aligned} \varphi_1 = &\square((MarineHealth > 0.75) \wedge (MarineDamageRate = 0.0) \\ &\rightarrow (HealBid < 0.25)) \end{aligned} \tag{3}$$

Utilizing Psion and JKIND, we can evaluate our singular FIS node over this specification and either receive a mathematical guarantee that our system will adhere to this system in all cases, or receive a counterexample trace which proves that our system does not. Table 2 shows that a counterexample was found. Note that interestingly the presented counterexample corresponds to exactly the values 1.0 and 0.0 for our two inputs

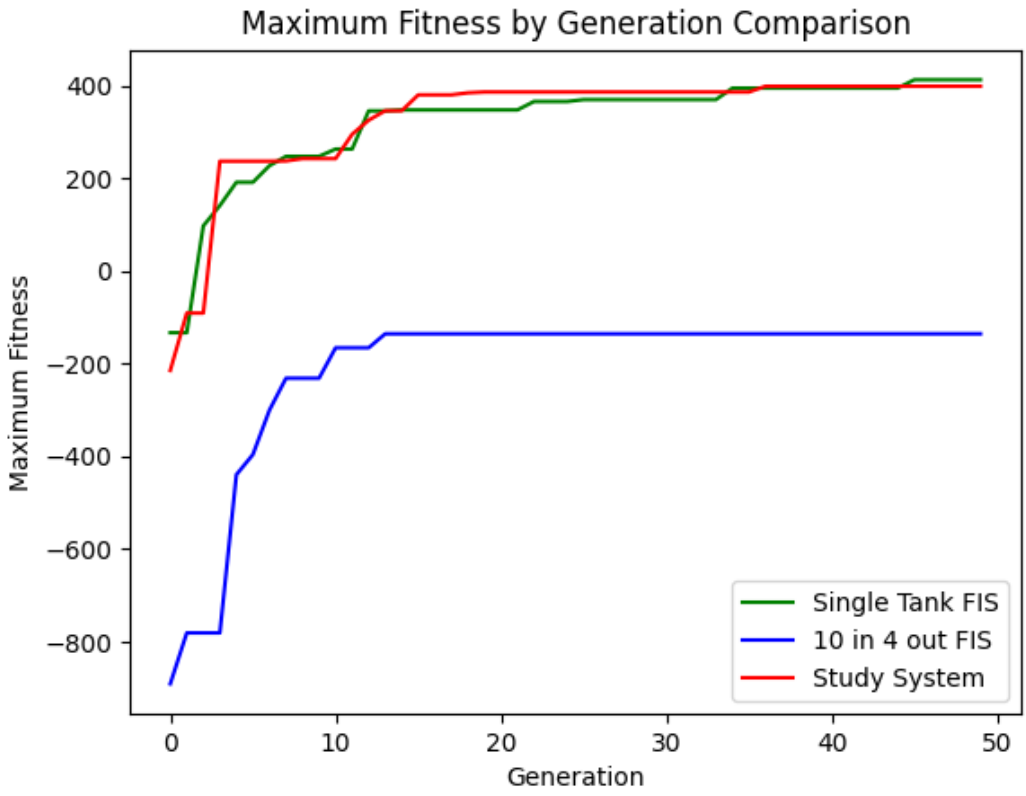

Figure 10. RL performance comparison of the variant with a single FIS for siege tank control, the variant with a single FIS providing all control actions, and the original system that is utilized within this study.

Table 2. Counterexample for Spec 1 failure prior to modification

Variable	Value
MarineHealth	1.0
MarineDamageRate	0.0
HealBid	0.3

respectively. A singular rule in our RB will correspond to exactly when the inputs are at those values, and those values are the extremes of the normalized space for both inputs. Hence, upon investigation of our RB we would see that we have a rule in our chromosome of "If Marine Health is Very High and Marine Damage Rate is None, then Healing Bid is Low". By changing the output MF to "Very Low" for this rule, we can easily solve this specification failure.

Though not a requirement, it is also often very beneficial to develop a scenario within the simulation environment that can help represent a potential case of a found counterexample. Observing the agent's performance through these specifically designed scenarios can both provide insight to the control issue, as well as provide feedback to the solution of the problem once found. Note that removing the failure case from a specific scenario does not guarantee that the specification will always hold, but measures can be taken to ensure this within regions of interest and verified again using Formal Methods. In this case, it is not necessary due to the simplicity of the counterexample found, but was completed and is seen below in Figures 11 and 12.

A scenario was developed in which there are two marines, a medivac, and two zerglings. The zerglings are at full health, but the marines are wounded. In particular, the marine at the top of the formation is moderately

Figure 11. The three frames of the φ_1-non-adherent medivac healing spec scenario are (Left) both marines with the top marine alive and the bottom being healed, (Center) the top marine defeated and the bottom marine still being healed, and (Right) the scenario complete with the bottom marine surviving and all zerglings defeated.

Figure 12. The three frames of the φ_1-adherent medivac healing spec scenario are (Left) both marines alive and the wounded top marine being healed, (Center) the top marine under attack and surviving with medivac support, and (Right) the scenario complete with both marines alive and all zerglings defeated.

hurt, and the marine at the bottom only slightly so. In the case of Figure 11 the unmodified chromosome was used and for a slight period of time, before the zerglings reach the friendly forces, it heals the bottom marine first. This leads to the top marine being defeated, reducing the damage output of the friendly squad by half. The bottom marine does manage to survive, but just barely.

If instead the bottom marine is ignored, at least at first, then the medivac heals the top marine, managing to keep both alive and secure a higher scoring victory. This is shown in Figure 12, though again simply because this specific scenario shows adherence to the specification, formal verification is needed to confirm that it always holds.

3.2.2. Safety specification 2

Safety specification 2 states "If a separate friendly marine has already bid to attack a target whose health is very low, than a marine's fire bid on that target must be very low." This is represented in the same manner as the first specification in Equation 4 and in LTL in Equation 5, utilizing the MF distribution shown above in Section 2.3.

$$\begin{aligned} Spec2 = & If(TargetHealth < 0.15) and (MarinesAssigned > 0.0) \\ & then(FireBid < 0.25) \end{aligned} \tag{4}$$

Figure 13. The three frames from the φ_2-adherent marine firing control scenario are (Left) two marines with six wounded zerglings approaching, (Center) halfway through scenario with three remaining zerglings, and (Right) the scenario complete with both marines alive due to intelligent targeting.

$$\begin{aligned}\varphi_2 =&\square((TargetHealth < 0.15) \wedge (MarinesAssigned > 0.0) \\ &\rightarrow (FireBid < 0.25))\end{aligned} \tag{5}$$

Despite not being explicitly directed to, the resultant chromosome received from training was proven through JKIND to already adhere to this specification, with the resultant output confirming correctness across all input values. However, a relevant scenario was still created and is shown in Figure 13. Within this mission there are only two marines on the friendly forces and six zerglings. However these marines are at full health and the zerglings are each wounded to the point where two shots from the marines would defeat them. Thus, intelligent target assignment is required to win, but victory is possible.

While the training process did generate a model that was adherent to this specification, the confirmation of specification adherence across all possible scenarios is still worthwhile.

3.2.3. Safety specifications 3 and 4

The last two specifications investigated within this study are related, both dealing with the fire control of the siege tank. Specification 3 again being "If a friendly unit would take splash damage from a tank shot and the lowest health friendly in splash range is between very low and low health, only fire at the target if it would do very high damage to the enemy" and able to represented as in Equation 6 and in LTL in Equation 7.

$$\begin{aligned}Spec3 =&If(FriendlySplash > 0.0)and(0.40 > LowestFriendly > 0.15)and(ShotEff. < 0.75) \\ &then(Bid < 0.50)\end{aligned} \tag{6}$$

$$\begin{aligned}\varphi_3 =&\square((FriendlySplash > 0.0) \wedge (LowestFriendly > 0.15) \wedge (LowestFriendly < 0.40) \\ &\wedge (ShotEff < 0.75) \rightarrow (Bid < 0.50))\end{aligned} \tag{7}$$

This is due to the fact that the siege tank will only fire at a target if the fire bid is above 50%, so if the shot effectiveness is below "very high" then we do not wish to fire. This specification did fail with the post-training chromosome, and produced a counterexample as can be seen in Table 3. The counterexample displayed here demonstrates that our AI model was perhaps close to adhering to the specification; the shot priority value was not far above the maximum threshold of 0.50. For this specification there are three FISs to consider, but given the context of the specification a general reduction in the output MFs of the Shot Safety FIS was selected to modify this FIS until it was adherent. However, by changing the MFs and RBs in any of the three FISs

Table 3. Counterexample for Spec 3 failure prior to modification

Variable	Value
CombinedEnemySplash	0.75
CombinedFriendlySplash	0.35
EnemyHealth	0.583
SplashedFriendlyLowestHealth	0.25
ShotEffectiveness	0.5
ShotPriority	0.55
ShotSafety	0.5

there are likely solutions that would be similarly adherent. This specification may be joined by others in the future though, and attempting to modify this FIS tree while maintaining reasonable semantic logic is ideal; thus modifying the safety evaluation FIS was selected.

A scenario was developed to demonstrate this issue which contains one siege tank, two marines, and six zerglings. Readers familiar with Starcraft 2 may be aware that siege tanks within the game do not natively have the ability to hold their fire. Modifications to, or deeper usage of the API can allow a unit to be "paused". In the example scenario created for the tank specifications however, a simple off-screen stationary structure served as a possible target for the tank to attack, rather than the zergling if it chose to not fire at the hostile troops.

Figure 14 shows the post-training FIS in this new scenario. Four specific frames are present in this image, with the upper row being frames 1 and 2, and the bottom being 3 and 4. In the first frame, the tank does take an effective shot at the zerglings before they enter splash range with the top marine. The tank is not able to take another shot again until Frame 2, which is not against the spec as the top marine has very low health at this time and is surrounded by three zerglings. That shot defeats two zerglings, as well as the top marine, but adhered to the specification. The remaining zergling then approaches a full health bottom marine in Frame 3. Unfortunately, in Frame 4, the tank is ready to fire again, and kills a now wounded bottom marine just before he could defeat the zergling and still be alive. However, the tank did survive so the friendly forces secured a victory, just not an ideal or adherent one.

In Figure 15 the engagement plays out almost entirely the same in frames 1 through 3. However, come frame 4, the tank is opting to not attack, allowing the bottom marine to secure the final kill, completing the mission with only 1 marine loss.

Specification 4 is essentially a much more conservative option than specification 3 and states "Never cause splash damage to any friendly unit, regardless of their health". This is represented in natural language in Equation 8 and in LTL in Equation 9.

$$\begin{aligned} Spec4 = & If(FriendlySplashDamage > 0.0) \\ & then(FireBid < 0.50) \end{aligned} \tag{8}$$

$$\begin{aligned} \varphi_4 = & (FriendlySplashDamage > 0.0) \\ & \rightarrow (FireBid < 0.50) \end{aligned} \tag{9}$$

The failure case presented to this system from our trained chromosome is the exact same from JKIND, which is as we expect given the fact that this specification is more conservative. Many solutions exist to create a set of FISs that adhere to this specification, including modifying both the Shot Safety FIS to have much lower

Figure 14. The four frames from the φ_3-non-adherent siege tank firing control scenario are (Upper Left) scenario start showing siege tank taking a safe shot, (Upper Right) siege tank fires at zerglings within range of healthy marine, (Lower Left) marine death due to friendly siege tank fire, and (Lower Right) siege tank firing and defeating all zerglings and remaining friendly marine.

Figure 15. The four frames from the φ_3-adherent siege tank firing control scenario are (Upper Left) scenario start showing siege tank taking a safe shot, (Upper Right) siege tank fires at zerglings within range of near-death marine, (Lower Left) low-health marine death due to friendly siege tank fire, and (Lower Right) siege tank holding fire and allowing marine to defeat final zergling.

Figure 16. The four frames from the mission failure, φ_4-adherent siege tank firing control scenario are (Upper Left) scenario start showing siege tank taking a safe shot, (Upper Right) siege tank does not fire on zergling group in proximity to top marine, (Lower Left) top marine is defeated by zergling group and siege tank continues holding fire, and (Lower Right) mission failure because all marines defeated and siege tank cannot target nearby units.

Output MFs referenced in the RB, as well as in the Shot Effectiveness/Safety Comparison FIS. A version of the modified training chromosome was manually tuned to the point where adherence to this specification was proven by JKIND. Figure 16 below shows the performance of this new system, which essentially opts out of taking a meaningful shot at Frame 2 when the hostile army is heavily grouped around the top marine. While adherent to the spec, this mission ends in failure.

4. DISCUSSION

As can be seen in the Results section, the performance of the AI system after the reinforcement learning process was such that it was likely superhuman[25]. Note that this is heavily dependent on the hypothetical human player. As the actions per minute of the AI system was not constrained, it could select and execute much faster than any human likely could. Note that this is also somewhat different from other SC2 AI systems like AlphaStar, as those are constrained to have near-human constraints on information and action execution rates. The purpose of the AI in this work though is to perform a subset of actions in particular scenarios and is not meant to be a general SC2 player. Instead, at least for specifications 1 through 3, the focus was on creating a high performance AI for unit controls in difficult engagements that were also explainable and formally proven to adhere to safety specifications.

With respect to the safety specifications, it was shown in both simulation and through verification with the model checker JKIND that counterexamples were found for most of the specifications. That is, there were conditions where the system could output actions that would violate the specifications. The counterexamples, due to the transparent and explainable nature of GFTs, were then used to assist in identifying parameters that needed correction. This was achieved by expert knowledge, specifically familiarity with Starcraft 2, but could also be aided via further reinforcement learning runs to correct undesired behaviors. After correction, the

system was verified against the specifications which were shown to always hold. Although simulation results also show that the system performs as intended in the particular runs executed, the benefit of formal verification is that it will hold over any potential run with the given AI system. This is the differentiating factor for formal verification compared to more common methods that rely on numerical evaluation. Formal verification, in this case model checking and SMT solving, can give definitive evidence that the system is correct, while numerical evaluation can only return found counterexamples, but not conclusively identify that there are none.

The case of specification 4 presents a unique point; adherence to certain safety specifications may not necessarily improve overall system performance. For example, there are often strict standards for use-cases that require certification which may overall slow down or weaken control systems, but are still necessary in order to deploy and trust the system. Consider laws such as speed limits placed onto a theoretical optimal driving controller; if the fitness function is minimizing travel time and fuel consumption, an unconstrained RL agent would likely optimize to illegal behaviors in some cases. However, for AI systems in mission/safety-critical applications the benefit of formal verification cannot be understated. In this example of specification 4, all friendly fire was strictly disallowed, even if it would lead to the death of the entire force.

Regarding explainability, the structure of the GFT created is such that actions are easily explained and interpretable by humans. This is of particular interest when auditing the system for explanations on behavior that was learned throughout the reinforcement learning process. In this case, it was also used for a human designer to correct errors in the AI system. This is a powerful capability when creating complex AI control systems for safety-critical applications. Although SC2 AI play will likely never be considered "safety-critical", it serves as a good proxy problem for other decision-making applications in notionally similar environments.

4.1. Future work and extensions

Potentially the most obvious extension of this work is to make a more general AI for SC2 that plays more portions of the game including things such as: base building, resource management, etc. This would demonstrate the applicability and efficacy of GFT towards a well-known benchmark problem in RL. That could then be used to show how explainability in complex AI systems may be desired, especially as a learning tool for humans.

In the future, these scenarios could also be used for comparisons against other RL approaches. Due to the lack of explainability and formal verifiability of common RL techniques compared to GFTs, the resulting comparison would be purely performance based.

Another important extension of this work would be to extend the formal verification methodology to include portions of the Starcraft 2 dynamics. Doing so would allow for reasoning about higher level behaviors of the system. In this work, only properties about the GFT itself are verified, while inclusion of dynamic models for portions of Starcraft 2 would allow for interesting temporal specifications to be evaluated. E.g. if dynamics are included for unit damage, damage inflicted by attacks, cool downs, unit movements, etc., specifications could be created to reason about, say, whether it's possible for any friendly units to be lost, or finding guaranteed minimums. The value of doing this is exceptional as it can provide definitive evidence of baseline performance, but of course is much more difficult than verification of the AI system without dynamics.

5. CONCLUSION

In conclusion, a GFT model was developed and trained for a particular scenario and then verified against four behavioral specifications. Post-training, the model did not adhere to three of these specifications, and counter-examples were found from the formal methods tools utilized. For these cases, the GFT was evaluated over scenarios within the Starcraft 2 environment which demonstrated the specific failure modes detailed in the counterexamples. Modifications to the system were then made for each specification until the system was

proven to be adherent to these safety specifications. The resulting GFT is then guaranteed to be adherent to specifications over all input values while being explainable and performant. While this study does not intend to demonstrate performance of an entire Starcraft 2 game controller, it demonstrates the capability of a Fuzzy Logic-based AI system to be trained and proven to adhere to safety specifications in a specific subset of the control actions within this game that represent mission/safety-critical elements.

DECLARATIONS

Authors' contributions

Made contributions to conception and design of the work, developed most associated code for Starcraft integration and reinforcement learning, performed data analysis and figures of interest, manuscript writing and related tasks: Ernest N

Contributed to parts of the conception and design, implemented the formal verification techniques towards Fuzzy Systems, performed analysis and translation of formal specifications, manuscript writing and related tasks: Arnett T

Created the interfaces needed to easily integrate the Thales's toolkit with Python environments such as the SC2 interfaces used in this work, manuscript writing and related tasks: Phillips Z

Availability of data and materials

While the resultant GFT AI model and the Psion and EVE software cannot be shared openly, the Starcraft 2 map files utilized in the scenarios shown in this study could be. They are not currently hosted, but can be made available upon request.

Financial support and sponsorship

None.

Conflicts of interest

All authors are employees of Thales Avionics Inc., from which two software packages that are commercially available were utilized in this research.

Ethical approval and consent to participate

Not applicable.

Consent for publication

Not applicable.

REFERENCES

1. Zhao Y, Wang H, Xu N, Zong G, Zhao X. Reinforcement learning-based decentralized fault tolerant control for constrained interconnected nonlinear systems. *Chaos, Solitons & Fractals* 2023;167:113034. DOI
2. Tang F, Niu B, Zong G, Zhao X, Xu N. Periodic event-triggered adaptive tracking control design for nonlinear discrete-time systems via reinforcement learning. *Neural Netw* 2022;154:43-55. DOI
3. Silver D, Huang A, Maddison CJ, et al. Mastering the game of Go with deep neural networks and tree search. *Nature* 2016;529:484-9. DOI
4. Gunning D, Aha D. DARPA's explainable artificial intelligence (XAI) program. *AIMag* 2019;40:44-58. DOI
5. Ross TJ. Fuzzy logic with engineering applications. John Wiley & Sons; 2009. DOI
6. Castro JL. Fuzzy logic controllers are universal approximators. *IEEE Trans Syst, Man, Cybern* 1995;25:629–35. DOI
7. Buckley J, Siler W, Tucker D. A fuzzy expert system. *Fuzzy Sets and Systems* 1986;20:1–16. DOI

8. Coleman CP, Godbole D. A comparison of robustness: fuzzy logic, PID, and sliding mode control. In: Proceedings of 1994 IEEE 3rd International Fuzzy Systems Conference. IEEE; 1994. pp. 1654–59. DOI
9. Moral A, Castiello C, Magdalena L, Mencar C. Explainable Fuzzy Systems. Springer; 2021. DOI
10. Arnett TJ. Verification of genetic fuzzy systems [MS Thesis]. University of Cincinnati; 2016. DOI
11. Ernest ND. Genetic fuzzy trees for intelligent control of unmanned combat aerial vehicles. University of Cincinnati; 2015. DOI
12. Herrera F. Genetic fuzzy systems: taxonomy, current research trends and prospects. *Evol Intel* 2008;1:27-46. DOI
13. Fleck DE, Ernest N, Adler CM, et al. Prediction of lithium response in first-episode mania using the LITHium Intelligent Agent (LITHIA): pilot data and proof-of-concept. *Bipolar Disord* 2017;19:259-72. DOI
14. Ernest N, Carroll D, Schumacher C, et al. Genetic fuzzy based artificial intelligence for unmanned combat aerial vehicle control in simulated air combat missions. *J Def Manag* 2016;;6:2167-0374. DOI
15. Thales. Thales GFT AI Toolkit. Thales; 2022. Available from: https://www.thalesgroup.com/en/markets/aerospace/big-data-aerospace/genetic-fuzzy-tree-ai-toolkit-critical-decisions. [Last accessed on 20 Mar 2023]
16. Marques-Silva J. Practical applications of boolean satisfiability. In: 2008 9th International Workshop on Discrete Event Systems. IEEE; 2008. pp. 74–80. DOI
17. Hinchey MG, Bowen JP. Applications of formal methods. vol. 1. Prentice Hall New York; 1995. DOI
18. Ernest N, Kunkel B, Arnett T. An investigation into the impact of system transparency on work flows of fuzzy tree based AIs. In: North American Fuzzy Information Processing Society Annual Conference. Springer; 2020. pp. 349–59. DOI
19. Moura Ld, Bjørner N. Z3: An efficient SMT solver. In: International conference on Tools and Algorithms for the Construction and Analysis of Systems. Springer; 2008. pp. 337–40. DOI
20. Gacek A, Backes J, Whalen M, Wagner L, Ghassabani E. The JKind model checker. In: International Conference on Computer Aided Verification. Springer; 2018. pp. 20–27. [DOI
21. Mamdani EH, Assilian S. An experiment in linguistic synthesis with a fuzzy logic controller. *Int J Man-Mach Studies* 1975;7:1–13. DOI
22. Scapin D, Cisotto G, Gindullina E, Badia L. Shapley Value as an Aid to Biomedical Machine Learning: a Heart Disease Dataset Analysis. 2022 22nd IEEE International Symposium on Cluster, Cloud and Internet Computing (CCGrid) 2022:933–39. DOI
23. Heuillet A, Couthouis F, Díaz-Rodríguez N. Collective EXplainable AI: Explaining Cooperative Strategies and Agent Contribution in Multiagent Reinforcement Learning With Shapley Values. *IEEE Comput Intell Mag* 2022;17:59–71. DOI
24. Burnysc2. Burnysc2 python-SC2 Python Package. Github; 2022. Available from: https://github.com/BurnySc2/python-sc2. [Last accessed on 20 Mar 2023]
25. Wikipedia. Progress in artificial intelligence. Wikimedia Foundation; 2022. Available from: https://en.wikipedia.org/wiki/Progress_in_artificial_intelligence#Current_performance. [Last accessed on 20 Mar 2023]