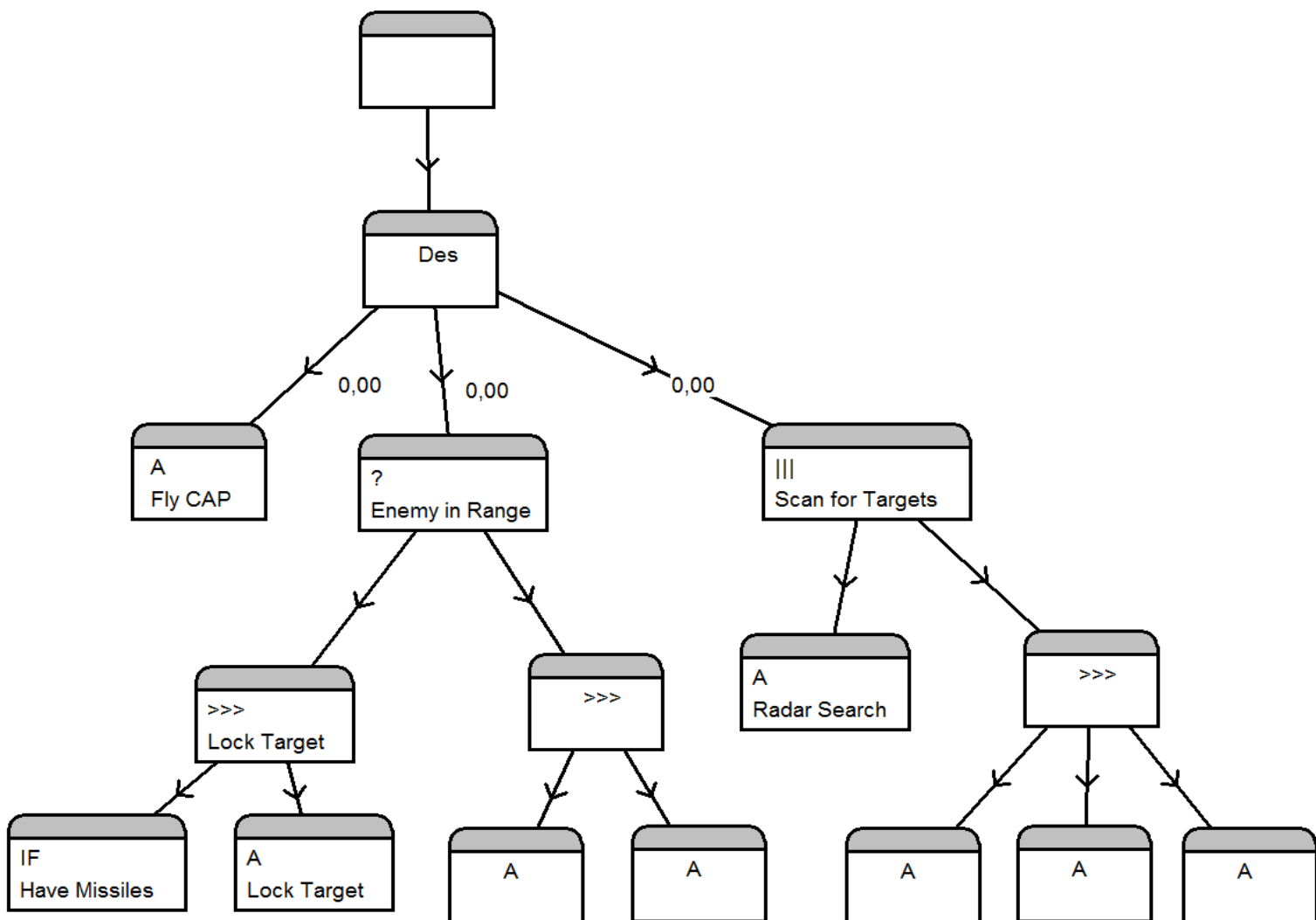


Architecture for goal-driven behavior of virtual opponents in fighter pilot combat training

Customer

National Aerospace Laboratory NLR

NLR-TP-2013-071 - March 2014



National Aerospace Laboratory NLR

Anthony Fokkerweg 2

1059 CM Amsterdam

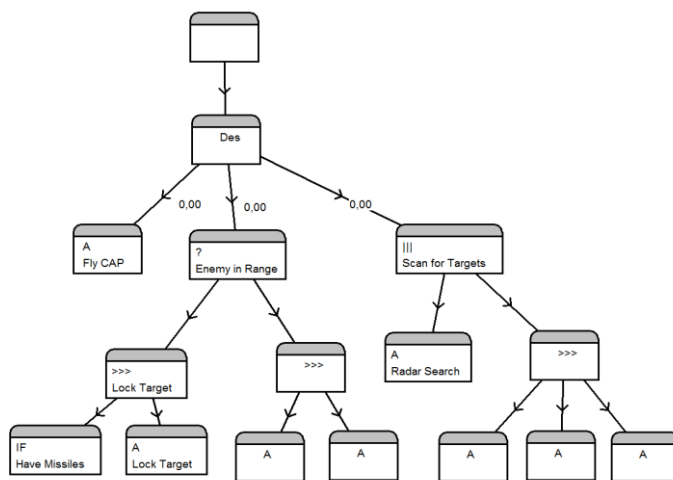
The Netherlands

Tel +31 (0)88 511 3113

www.nlr.nl

EXECUTIVE SUMMARY

Architecture for goal-driven behavior of virtual opponents in fighter pilot combat training



Problem area

A significant part of modern-day training of fighter pilots consists of exercising tactical mission scenarios in live training, or in (networked) simulators. In such training the role of the hostile forces, or 'red air', is often performed by instructors or other pilots. While using humans may yield satisfactory fidelity for the behavior of opposing forces, there are major disadvantages. Expert role players for the opponent role are scarce and expensive resources and the training value for such experts in the 'red air' role is generally low since the focus of the training is on the 'blue' tactics rather than the 'red' tactics.

Intelligent Computer Generated Forces (CGFs) can provide a solution to overcome these problems. The behavior of such intelligent agents is generally governed by a set of rules and mechanisms that constitute a behavioral model.

Report no.

NLR-TP-2013-071

Author(s)

A. Khatami
P.F. Huibers
J.J.M. Roessingh

Report classification

UNCLASSIFIED

Date

March 2014

Knowledge area(s)

Training, Missiesimulatie en
Operator Performance

Descriptor(s)

Goal-driven behavior
Computer Generated Forces
Opponent modeling
Artificial Intelligence

Typically, agents that are designed to exhibit motivation-based behavior are best suited to convincingly act in the opponent role. Constructing a rigorous model to create such behavior on a complex battlefield is, however, very challenging. The aim of this paper is to take on this challenge by proposing a goal-directed architecture for a computational model that exhibits human-like behavior in the domain of air combat.

Description of work

In this work we designed and tailored the implementation of goal-directed agents that would fit within the architecture of Smart Bandits project. This was done with improvement of scalability and dynamic properties in mind.

To minimize the need for human intervention during training, the behavior of CGFs is modeled by a hierarchical goal structure which is partly assembled dynamically during run-time. The mechanisms of determining which sub-goal structure to use are explored and established.

Results and conclusions

The proposed goal-directed architecture was implemented in our demonstration program where it is possible for agents to engage each other in opposing teams. We were able to observe the change in behavior for each team by assigning different settings and tactics to team members.

Goal-directedness provides additional flexibility to e.g. Finite State Machines FSMs. Despite being very similar to FSMs, a goal-driven architecture has better dynamic properties, is more scalable and may be extended with predictive behavior and state memory, providing effective means to create more challenging and valid opponent CGFs.

Applicability

The implementation of goal-driven agents for Smart Bandits is suited for any air combat training that would require the use of intelligent CGF's. This concept can be adapted for use in other types of trainings as well. It can be used for any type of simulated military or civil equipment and non-player characters (e.g. enemy soldiers or civilians).

In the future, simulators will play an increasingly important role in the training of fighter pilots. Believable and challenging

opponent behavior is essential to improve the quality of the training. Simulation of a hostile fighter pilot requires specific knowledge and skills that are not always in possession of a Simulator Operator. The current state of CGFs does not live up to the required level of fidelity and should be improved.

Once the behavior model, as described in this paper, is tailored for general use (i.e. not specifically for Fighter 4-ship), it will help Royal Netherlands Air Force (RNLAf) to improve the effectiveness of their simulator training.



Architecture for goal-driven behavior of virtual opponents in fighter pilot combat training

A. Khatami, P.F. Huibers and J.J.M. Roessingh

Customer

National Aerospace Laboratory NLR




March 2014

This report is based on a poster presentation held at the Behavioral Representation in Modeling and Simulation, San Antonio, TX, USA, March 12-14, 2013.

*The contents of this report may be cited on condition that full credit is given to NLR and the authors.
This publication has been refereed by the Advisory Committee AIR TRANSPORT.*

Customer National Aerospace Laboratory NLR
Contract number -----
Owner NLR
Division NLR Air Transport
Distribution Unlimited
Classification of title Unclassified
Date March 2014

Approved by:

Author A. Khatami <i>b/a</i> 	Reviewer J. van der Pal 	Managing department H.G.M. Bohnen 
Date <i>18-03-2014</i>	Date: March 2014	Date <i>20-03-2014</i>

Summary

The Smart Bandits project, undertaken by National Aerospace Laboratory for the Royal Netherlands Air Force, aims at developing Computer Generated Forces (CGFs) exhibiting realistic tactical behavior so as to increase the value of simulation training for fighter pilots. This paper explores the use of goal-driven behavior in opponent CGFs. Here, the behavior of CGFs is governed by a hierarchical goal structure which is determined dynamically during run-time. Although the definition of goals bears similarities to hierarchical Finite State Machines (FSMs), its dynamic nature makes it a more powerful method since it does not depend on predefined state transitions. Any number of goal-driven agents can be instantiated, and cooperate towards common goals, without the need to re-model their (collective) behavior. This improves the scalability of our implementation. The dynamic properties and scalability of this goal-driven agent architecture make it an effective method to create CGFs that exhibit human-like behavior.

This page is intentionally left blank.

Content

Abbreviations	6
1 Introduction	7
2 Related Work in Opponent Modeling	8
2.1 Cognitive modeling	8
2.2 Finite-state machines	9
2.3 Machine learning	9
3 Chosen Approach to Goal-directedness	11
3.1 Goal-driven agent behavior	11
3.2 Goal evaluator	12
3.3 Goal hierarchy	13
4 Implementation	14
4.1 Use-case	14
4.2 Design	14
4.3 Agent interactions	15
4.4 Desirability	16
4.5 Tactics	18
5 Preliminary Results	19
6 Discussion and Conclusions	20
7 References	21

Abbreviations

Acronym	Description
FSM	Finite State Machine
CGF	Computer Generated Force
AI	Artificial Intelligence
NLR	National Aerospace Laboratory
CAP	Combat Air Patrol

1 Introduction

A significant part of modern-day training of fighter pilots consists of exercising tactical mission scenarios in live training, in (networked) simulators. In such training the role of the hostile forces, or Red Air, is often performed by instructors or other pilots. While using humans may yield satisfactory fidelity for the behavior of opposing forces, there are major disadvantages. Expert role players for the opponent role are scarce and expensive resources and the training value for such experts in the Red Air role is generally low since the focus of the training is on the Blue tactics rather than the Red tactics.

Intelligent Computer Generated Forces (CGFs) can provide a solution to overcome these problems. These CGFs are autonomous entities that potentially provide challenging training scenarios on the basis of their own decisions, without interference of human experts (pilots of instructors). The behavior of such intelligent CGFs is generally governed by Artificial Intelligence (AI), e.g. a set of rules and mechanisms that constitute a behavioral model. Different types of behavioral models have been proposed, ranging from simple predefined behaviors to complex cognitive architectures with learning capabilities. The resulting behaviors, however, can be divided into four categories (Roessingh et al, 2012):

1. Non-responsive behavior.
2. Stimulus-Response (S-R) behavior.
3. Delayed Response (DR) behavior.
4. Motivation-based behavior.

Typically, agents that are designed to exhibit motivation-based behavior are best suited to convincingly act in the opponent role. Constructing a rigorous model to create such behavior on a complex battlefield is, however, very challenging. The aim of this paper is to take on this challenge by proposing a goal-directed architecture for a computational model that exhibits human-like behavior in the domain of air combat.

This paper is structured as follows: First we review existing work in Section 2. In Section 3 we describe our approach to implement goal-driven agent behavior. Section 4 explains our implementation. We present our results in Section 5. We discuss and conclude in Section 6.

2 Related Work in Opponent Modeling

The initial step in researching different AI models for controlling CGF behavior was the development of an architecture in which AI models were decoupled from the CGFs they were controlling. This enabled AI models to be developed in any (generic) programming language and be linked to simulated platform(s) in scenarios that are managed by a so-called scenario management tool. Roessingh et al (2012) provide a description of the developed architecture. For the management tool, we used the commercial-of-the-shelf product STAGE (Presagis, 2013). Abdellaoui et al (2009) reviewed different of these scenario management packages with respect to their AI capabilities. The following subsections discuss various approaches applied at the National Aerospace Laboratory (NLR) to model opponent CGF's behavior using AI techniques.

2.1 Cognitive modeling

The interaction between pilot and opponent determines for a large part the challenge of air combat. In order for flight simulators to provide pilots with realistic tactical training, their computer-controlled opponents need to behave intelligently and humanlike.

One approach to create humanlike opponent agents is cognitive modeling. The idea behind this approach is that to have agents behave humanlike, they need to have computer models of human cognitive processes. Development of these models is based on cognitive theories, input from domain experts and artificial intelligence modeling techniques. In the study 'Making Enemies: cognitive modeling for opponent agents', Merk (2013) develops several cognitive models for such opponent agents.

One of these cognitive models is the Situational Awareness (SA) model (Hoogendoorn et al. 2011). It defines the activation of concepts (the pilot's beliefs) on the basis of the observed state of the world. It is based on Endsley's (1995) model consisting of three levels of SA: at the lowest level, the pilot's perception of the world, subsequently comprehension of what is perceived, and at the highest level, projection of these comprehensions into the future, as to anticipate on future situations. The agent that is enriched with such SA takes its perceptions from the simulation environment and uses these to create complex beliefs about the current and future state of the environment.

In the current implementation these derived beliefs are used to influence the tactical decision making processes of the opponent agents by using the activation values of the beliefs as criteria for state transitions in Hierarchal Finite State Machines (HFMSMs) as will be explained hereafter.

2.2 Finite-state machines

A more traditional approach to defining agent behavior is given by (Hierarchical) Finite State Machines or HFSMs where behavior is decomposed into several states. Each state contains the logic to determine transitions to other states (Fig. 2.1). Using HFSMs to define agent behavior gives the programmer a great amount of control over the resulting behavior, but this behavior will be quite rigid and predictable. Furthermore, adding new behavior will make the model increasingly difficult to maintain and less adaptable for new scenarios.

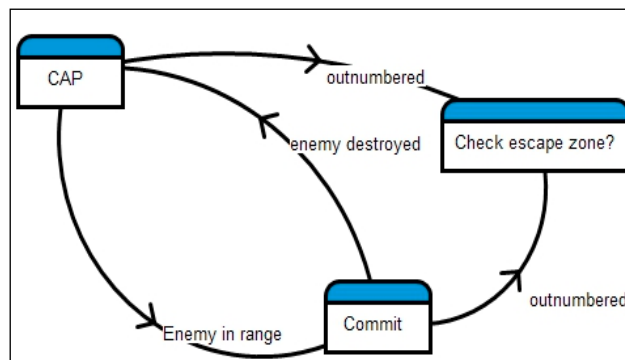


Figure 2.1: The states of an FSM. The states transition into one another when a certain condition holds.

2.3 Machine learning

Although cognitive models are very useful to establish human-like behavior in an agent, tailoring these cognitive models towards a certain scenario can be a time-consuming task requiring a lot of domain expertise. Koopmanschap et al (2013) take the earlier described SA model as a basis, and the addition of scenario specific information is for a large part automated. The approach of automatically adding scenario specific information has been evaluated using a case study in the domain of fighter pilots.

In a different research effort (described in Roessingh, Merk, Huibers, Meiland and Rijken, 2012) a technique called ‘off-line learning’ was used. In this approach, the agent is trained in another environment (the so-called ‘off-line environment’) than the ‘on-line environment’ in which it eventually has to function. Training in the off-line environment has the advantage that it can be performed in ‘fast-time’ without all the real-time- and graphics constraints that come with the on-line environment, in this case a manned flight simulator. However, one needs to ensure that the off-line environment is sufficiently similar to the on-line environment in order to ensure transfer of the trained behavior from the off-line to the on-line environment. After the agent has been trained sufficiently in the off-line environment, it will be extracted from the off-line environment and planted in the on-line environment, where it can merely exploit its trained behavior, without further learning. In this case, the opponent model was based on an artificial

neural network. The input nodes of network received information from the simulation environment. This information was passed through one layer of hidden neurons to four output neurons. These neurons corresponded with one of four actions (fly straight/left/right, fire, constrained by predefined dynamic limitations for the aircraft) which the agent could perform at any time. The goal was to let the network learn the best action to perform based on the observations (input) it received from the (off-line) environment. The network was trained using an evolutionary algorithm, where the fitness function was based on the outcome of the tactical air-to-air engagement. Since the set of actions that the agent was able to perform was obviously simplified (for research purposes), the resulting behavior was not as complex as one would expect from a real operator. It was, for instance, not possible for the agent to change its altitude or speed. Adding this kind of actions would greatly increase the realism of the resulting behavior, but at the same time this also increases the search space in which the algorithm has to search for the optimal solution.

For on-line learning, Roessingh, Huibers and Rijken (2012) applied a technique called *dynamic scripting*, a basic form of 'reinforcement learning' (Sutton & Barto, 1998), developed and applied in computer games by Spronck et al (2006). In dynamic scripting an agent is equipped with a large set of rules, called the rule base. These rules have a simple, script-like, form with a precondition and an action (which will be performed when the precondition holds). For each game or engagement by the agent, a subset of the rules in the rule base is chosen which forms the agent's script. Based on the outcome of the game, the weights of the rules in the rule base are adapted, which alters their probability to be selected for the agent's script in the next iteration. Essentially, if a rule has a positive influence on the performance of the agent, its probability to be selected will increase.

The three different examples of machine learning in the domain of air-to-air combat simulation show that machine learning can be a valid approach in creating challenging and adaptive behavior in Computer Generated Entities. A NATO Research Task Group entitled 'Machine Learning Techniques for Computer Generated Entities' (coded: IST-121-RTG-060) will support further research on applications of machine learning in this context.

Although fruitful on a small-scale, the work to-date at NLR in the field of cognitive modeling, finite state machines and machine learning for instilling opponent behavior in agents has been shown to provide insufficient support for scalability, i.e. the definition of a large number of (co-operative) agents, rather than just one or two. Therefore, this work was initiated by the search for techniques that facilitate such scalability. A suitable candidate for such technique is behavior tree modeling, as described in next section.

3 Chosen Approach to Goal-directedness

This section outlines our approach to implementing goal-driven agent behavior within the Smart Bandits project.

3.1 Goal-driven agent behavior

A goal is a state of the world that the agent tries to achieve. Goals can be either atomic or composite. Atomic goals define a single task or behavior, such as locking a target. Composite goals consist of other sub-goals which in turn can be composite (eliminating the target) or atomic (locking the target first, then shooting at it) and thus, defining a nested hierarchy.

Goal-driven agent behavior is a concept similar to Finite State Machines (FSMs). It is becoming popular in the AI developer community under the title of behavioral trees. It is the implementation of a nested hierarchical goal structure.

We denote the highest goal in the structure by the term 'AI brain' (Fig. 3.1). The AI brain's goal is to dynamically determine the path through the goal hierarchy by selecting a goal on the next hierarchical level (i.e. tactical level). The AI brain does this by evaluating the desirability of all other goals at the tactical level; a process that is coined by the term 'goal arbitration'. Furthermore, the AI brain maintains a 'sub-goal' stack to keep track of goals that are temporarily suspended when a new branch of the tree-structure is entered. All sensory data for the CGF (e.g. distances to other entities and number of radar contacts) pass through the AI brain.

All goals are generally able to monitor their status and change their status if they fail. Hence, the status of a goal can be inactive, active, completed or failed.

In our approach, the AI brain is the top-most goal in the hierarchy with the sole purpose of deciding which goal to select next by evaluating the desirability of achieving a particular goal. It also monitors the effect of the actions that are being taken in order to achieve the goal. The AI brain only fails when the agent is deactivated or eliminated during simulation.

The hierarchical nature of this architecture provides us with an intuitive way for implementing motivation-based behavior, which bears similarities to human behavior. Humans select abstract (i.e. nonconcrete) objectives based upon their needs (e.g. buy groceries) and decompose them into a plan of more concrete actions that can be followed. This includes considering various ways to achieve a particular goal (e.g. go on foot or by bike). This process is then repeated until the

actions need no further planning (e.g. take a pack of milk off the shelf). This way, reasoning about goals is dynamic and takes place when goals become relevant.

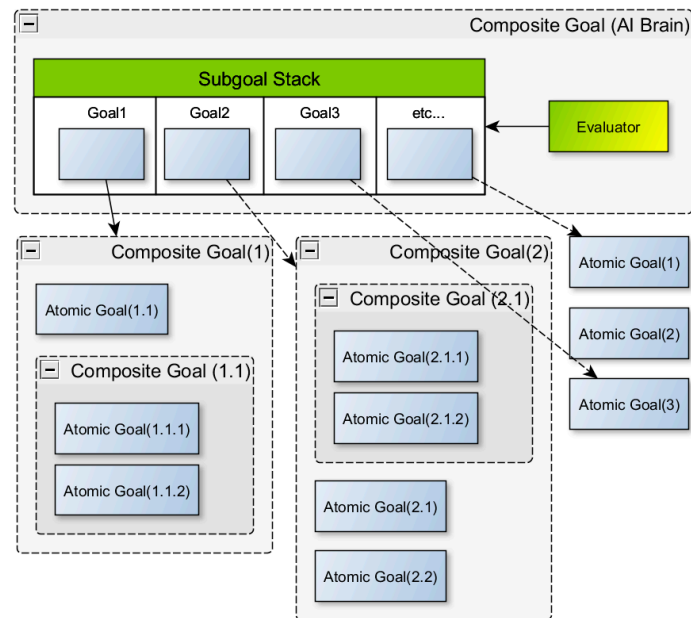


Figure 3.1: Schematic of a goal structure. All sub-goals are defined separately. The evaluator determines the goals in the sub-goal stack which also specifies the order of execution.

The same process is mimicked by the goal-driven agent. During each update of its AI brain, a high-level goal is chosen with the highest desirability value, explained in the next subsection. This goal is then further decomposed until it can be completed through a series of atomic actions in order of their necessity.

3.2 Goal evaluator

In contrast to FSMs, the goal hierarchy is largely determined dynamically during run-time. This is done by evaluating the desirability of achieving a goal and is facilitated by the goal evaluator in the AI brain. The evaluator is defined as a function which takes into account relevant parameters to each goal. In the current implementation, there is a single evaluator function that calculates the desirability of each goal. The evaluator runs in parallel with the processes of a currently active goal.

When the evaluation of a specific goal yields a higher desirability than the current active goal, the latter will be terminated and replaced by the new goal. Proper care must be taken in order to make goal termination justified. For instance, if a goal is near completion then it would be unrealistic to discontinue current action in order to pursue the new goal. The quality of goal arbitration defines the realism of AI implementation.

3.3 Goal hierarchy

Each agent has an instance of the AI brain which is updated in regular time intervals. During each update the hierarchy of goals (active, not yet active or suspended) is constructed and executed.

The next node in the hierarchy is generally a composite goal consisting of sub-goals. In trivial situations, atomic goals can also be chosen. In our construction, goals are mostly executed sequentially. For example, an agent would first intercept its target, then get a lock on it and eventually launch a missile to eliminate it. However, the flexibility of goal-driven agent behavior's frame-work allows for execution of goals in parallel (Fig. 3.2). In the example above, the agent would pursue its target *while* trying to get a lock on it. This is a very useful feature that can be utilized to achieve more realistic behavior.

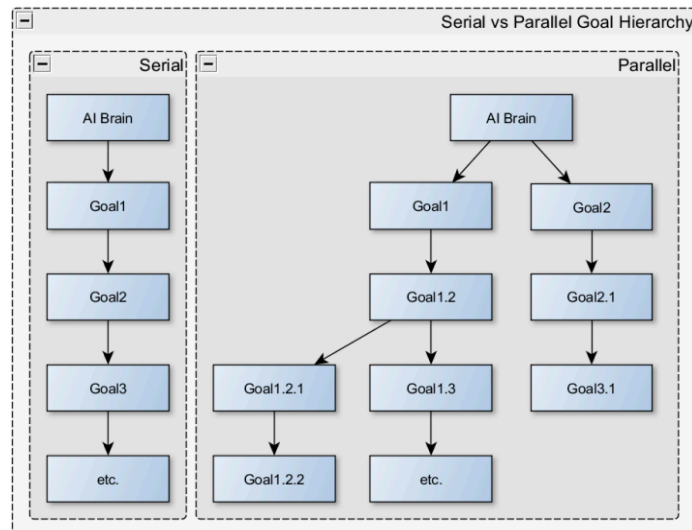


Figure 3.2: Goal hierarchies. In general, goals are executed sequentially. However, some goals may require parallel actions.

4 Implementation

This section describes our implementation design of goal-driven agent behavior. Our design approach as well as implementation closely follows the concepts treated in Buckland (2005). To fit the architecture to our purpose we introduced some minor deviations from the original concepts. Keywords denoting specific implementation terms, such as `Activate`, have a distinct type format.

4.1 Use-case

The Royal Netherlands Air Force provides tactical training for its F-16 fighter pilots. Tactical training aims at learning how to combine aircraft and weapons in order to defeat the opponents. F-16 aircraft usually operate in formations of two or four ships.

We use four networked F-16 simulators (the so called NLR fighter 4-ship) as the implementation platform for the tactical training environment. The intelligent CGFs are coupled to this training environment using middleware called Mediator (Roessingh et al, 2012) and the scenario management tool called STAGE (Presagis, 2013).

The opponent CGFs represent multiple hostile fighter aircraft, armed with multiple radar-guided medium-range missiles and equipped with air-to-air radar and associated avionics. The goal of the tactical training scenario for the F-16 pilots is to enter the airspace of the enemy in order to neutralize the opponent aircraft and eliminate ground threats. The opponents should be able to operate in formation to defend their airspace and be flexible in employing tactics when conditions change (e.g. when one of the aircraft in the formation is shot down).

4.2 Design

The nested hierarchy of goals as described in the previous section is best implemented using the composite design pattern (Gamma et al, 1994). As depicted in the UML class diagram (Fowler, 2004) of Fig. 4.1, each instance of a goal implements three member functions `Activate`, `Process` and `Terminate`. When a goal is instantiated, a call to `Activate` will initialize all data that are needed for the planning phase.

During each update step of the goal a call to `Process` will be made. `Process` contains the actions to complete the goal, monitors the goal's status, and will invoke possible sub-goals. It will return one of four possible states of the goal:

- **Inactive:** goal is waiting to be activated.
- **Active:** goal is active and is trying to satisfy its purpose.
- **Completed:** goal has succeeded and can be removed from the stack.
- **Failed:** goal has failed and will be either reactivated or removed from the stack.

When a goal is about to be removed from the stack its `Terminate` function, which could contain any exit-code (e.g. to keep a tally of missiles left), is called.

The AI brain takes care of the instantiation and removal of goals. In each evaluation step, the desirability of all goals is determined and the goal with the highest desirability value will be instantiated and put on the goal stack.

Concerning the computational resources, the required update rate of the AI brain is an important parameter. In the current implementation, air-to-air engagements take place at a distance ‘beyond-visual-range’, i.e. more than 10 Nautical Miles. This means that observations of the opponent are predominantly via radar and radar warning receiver. Since radar systems have a scan rate in the order of seconds, an update frequency of the AI-brain as low as 2 Hz is sufficient to make the agent react adequately and responsively.

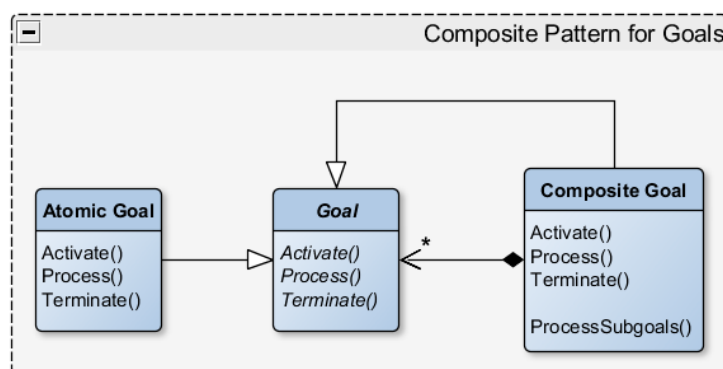


Figure 4.1: Both atomic and composite goals inherit from an abstract class `Goal`. Composite goals can contain one or more sub-goals of either type.

4.3 Agent interactions

One of the advantages of the goal-driven architecture is its scalability. This means that it is suitably efficient and practical when applied to a large number of participating agents. In terms of code implementation, the goal-driven architecture applied to multiple agents requires that group or team behavior is included in the goal definitions. If that is achieved, any number of agents can

be created and controlled by goal hierarchies. For this group behavior to emerge, agents need to be aware of each other to a certain degree. This awareness may include any physical (observable) states such as other agents' positions, speeds and sensor data (e.g. from radar).

In the current implementation, agents communicate via shared data and by exchanging messages. The former method is mostly used to retrieve data about other agents, whereas the latter is mostly used to create a command hierarchy between the agents. For example, if one of the agents is assigned as the leader then it will be able to issue a command to its wingmen to execute a certain maneuver.

4.4 Desirability

For each high-level goal there is an equation which is evaluated during every update step of the AI brain. The outcome is the desirability of achieving a particular goal and depends on the agent's situation. In our demo program we have defined five high-level goals:

- Fly combat air patrol (fly CAP)
- Follow leader
- Intercept target
- Eliminate target
- Evade

The first two goals are trivial and will only be chosen if there is nothing else to do. We will discuss evaluation of interception and elimination goals as an example.

When there are hostile forces present the agent can decide either to engage or evade, depending on its tactics (see next subsection). If the target is out of range then the need for interception will be highest and desirability of attacking will become larger only when the target is in weapons range. The equations which determine the corresponding values are,

$$D_{intercept} = k_1 \left(\frac{A}{D(R+1)} \right) \quad (1)$$

$$D_{eliminate} = k_2 \left(\frac{D_{intercept}}{D(R+1)} \right) \quad (2)$$

where A is the number of agent's air-to-air missiles, D is the distance to target and R is the number of hostile/unknown radar contacts. In order to force values of desirability to have the same order of magnitude, each equation is scaled by a weight factor k . This factor is also important for setting the transition point between competing goals. Note that (Eq.2) is inversely quadratic and will have larger values at shorter distances as is depicted in Fig. 4.2. For reasons of exposition the above desirability equations have been kept simple. In more realistic scenarios, other factors, such as aspect angle, may also affect the desirability equations¹.

Fine-tuning of weight factors k may be complex and expensive in terms of knowledge elicitation, especially with a growing number of goals. These values may depend on, among other things, weapon specifications (e.g. effective missile and radar ranges). The k values can be determined empirically, and adjusted via a user interface during test runs prior to actual use in training. In future implementations, the weight factors may also be determined using Machine Learning techniques (e.g. in the sense of Koopmanschap et al, 2013).

In the current implementation, the desirability equations such as (Eq.1) and (Eq.2) have to be programmed explicitly for each goal, which may be impractical for more complex air-air configurations for two reasons. First, these desirability equations reflect the expertise of the domain. Second, as these equations are liable to variations in the domain, their modification needs to be flexible and separated from the main implementation. At this time we are working on a scripting tool that takes expert knowledge on air-air engagements as its inputs and generates the desirability equations at its outputs.

¹ Obviously, real pilots are restricted to the extent with which they can numerically estimate parameters such as angles, speeds, distances and durations. In the current research this has been addressed by expressing such observed parameters as 'beliefs' (see Hoogendoorn, Lambalgen & Treur, 2011).

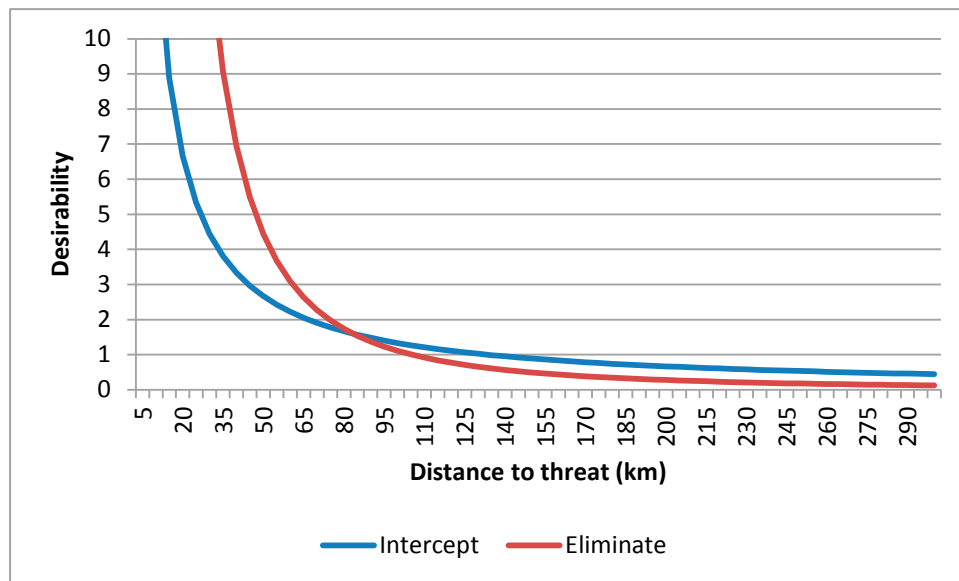


Figure 4.2: The desirability of eliminating a target becomes larger when the distance is less than missiles range.

4.5 Tactics

Overall behavior of agents can be adjusted in various ways. One obvious way is changing behavior by selecting a different goal. However, when a set of goals is defined for a certain kind of agent, all instances of that agent will behave identical. This may cause a noticeable synthetic appearance.

This synthetic appearance can be alleviated by defining different sets of goals that will be considered by the evaluation function. For instance, an agent would select a different set of goals if it were to fly a reconnaissance mission than it would do during a sweep mission (when its goal is to aggressively eliminate all hostile contacts). Although this is a very interesting option, our use-case doesn't require such variation at this time. Hence, we will leave this consideration for a future work.

Another way of introducing more diversity in (group) behavior is to vary weight factors k in desirability equations. As discussed above, the weight factors must be carefully tuned such that appropriate goals will become active in time. We defined three tactics which in fact define three sets of weight factors k for all equations. The tactics are Neutral, Defensive and Aggressive.

The weight factors are chosen in a way that it will change the agent's tactics. For example, a defensive agent would rather provide coverage to its wingman than initiating attack, while an aggressive agent, in contrast, would readily initiate an attack.



5 Preliminary Results

The proposed goal-directed architecture was successfully implemented in our demonstration program.

It was found that:

- Agents are able to engage each other in opposing teams.
- Changes in behavior for each team were observed when assigning different settings and tactics to team members.
- The size of teams can be set to any desired number without computational overhead, as long as team behavior is included in the goal definitions.

It should be noted that these results do not yet express the fitness of CGF's behaviors for training purposes. Currently, the architecture is in the process of being integrated in the NLR fighter 4-ship simulator to allow for such tests.

6 Discussion and Conclusions

In this section, we outline various possibilities to extend our research as part of the ongoing Smart Bandits project.

The evaluation function determines the tactical behavior of an opponent agent. In turn, a similar function can also be used by the agent itself to evaluate the goals of his human opponent. This provides the simulated agent with the capability to reason about the desirability of the goals of his opponent, e.g. determine the most desirable goal of his opponent. In fact, this constitutes a simple Theory-of-Mind model, a cognitive model that has been elaborated in more detail in Hoogendoorn and Merk (2013). Theory-of-Mind is a very important concept in tactical fighter operations and training that we will set to pursue in the implementation of goal-directed agents.

So far in our implementation, the high-level goals are terminated and removed from the stack when their desirability is exceeded by another goal. If the goals were to be suspended instead of terminated then the agent could be equipped with a memory queue. When the top-most goal has completed, any remaining suspended goal could be reactivated such that the agent would resume its previous interrupted activity. This functionality involves moderate precautions to determine if a suspended goal is still valid before it can be reactivated (e.g. destroy a ground target only if it has not been already destroyed by other CGFs).

The next step in our project is to use the new AI in conjunction with the NLR fighter 4-ship simulator. This enables fighter pilots to evaluate and validate the CGF behavior. Ultimately, the CGFs would have to pass an equivalent of the Turing's test in order to be a perfect substitute for human fighter pilots.

In summary, we proposed an architecture to implement goal-directedness in intelligent agents. Goal-directedness provides additional flexibility to e.g. FSMs. Despite being very similar to FSMs, a goal-driven architecture should theoretically have better dynamic properties, is more scalable and may be extended with predictive behavior and state memory, providing effective means to create more challenging and valid opponent CGFs.

7 References

- Abdellaoui, N., Taylor, A. and Parkinson, G., (2009). Comparative Analysis of Computer Generated Forces' Artificial Intelligence. NATO Modelling and Simulation Group (NMSG) Symposium (MSG-069): Use of M&S in Support to Operations, Irregular Warfare, Defence against Terrorism, and Coalition Tactical Force Integration, Brussels, Belgium, October 2009.
- Buckland, M., (2005). Goal-driven agent behavior. Programming Game AI by Example. (pp. 379-414). Sudbury: Wordware Publishing.
- Endsley, M.R., (1995). Toward a theory of Situation Awareness in dynamic systems. Human Factors 37(1), 32-64.
- Fowler, M., (2004). UML Distilled, A brief guide to the standard object modeling language. Third edition, Addison Wesley/ Pearsons Education, U.S.
- Gamma, E., Helm, R., Johnson, R. & Vissides, J. (1994). Design Patterns, Elements Of Reusable Object-Oriented Software. Pearsons Education, U.S.
- Hoogendoorn, M. & Lambalgen, R. van & Treur, J., (2011) An Integrated Agent Model Addressing Situation Awareness and Functional State in Decision Making. In: Kinny, D., Hsu, D. (eds.), Proceedings of the 14th International Conference on Principles and Practice of Multi-Agent Systems, PRIMA'11. Lecture Notes in Artificial Intelligence, vol. 7047, pp. 385–397. Springer-Verlag, Berlin Heidelberg, 2011.
- Hoogendoorn, M., Merk, R.J. (in press). Utilizing theory of mind for action selection applied in the domain of fighter pilot training. International Journal of Applied Intelligence.
- Koopmanschap, R. Hoogendoorn, M., Roessingh, J.J.M. (2013). Learning Parameters for a Cognitive Model on Situation Awareness. Paper submitted to the 26th International Conference on Industrial, Engineering & other Applications of Applied Intelligent Systems (IEA/AIE). To be held June 2013, Amsterdam, the Netherlands.
- Merk, R.J. (in press). Making Enemies: cognitive modeling for opponent agents. Ph.D. Thesis, VU University / National Aerospace Laboratory NLR, Amsterdam.
- Presagis (2013). Information on STAGE downloaded on 08 January 2013 from http://www.presagis.com/products_services/products/modeling-simulation/simulation/stage/
- Roessingh, J.J.M., Huibers, P., Rijken, R. (2012). Application of adaptive agents in tactical simulation. Paper presented for the Royal Aeronautical Society Conference on Flight Simulation Research 'new frontiers'. 28-29 November 2012, RAeS, London, UK.
- Roessingh, J.J.M., Merk, R.J., Huibers, P., Meiland, R., Rijken, R. (2012). Smart Bandits in air-to-air combat training: combining different behavioural models in a common architecture.

- Proceedings of the 21st conference on Behavior Representation in Modeling and Simulation, BRIMS 2012. 12-BRIMS-023. 147-154. BRIMS Society: Amelia Island, FL.
- Spronck, P.H.M, Ponsen, M.J.V., Sprinkhuizen-Kuyper, I.G., & Postma, E.O. (2006). Adaptive game AI with dynamic scripting. *Machine Learning*, 63(3), 217-248.
- Sutton, R.S., Barto, A.G. (1998). *Reinforcement Learning: An Introduction*, MIT Press, Cambridge, MA, US.

WHAT IS NLR?

The NLR is a Dutch organisation that identifies, develops and applies high-tech knowledge in the aerospace sector. The NLR's activities are socially relevant, market-orientated, and conducted not-for-profit. In this, the NLR serves to bolster the government's innovative capabilities, while also promoting the innovative and competitive capacities of its partner companies.

The NLR, renowned for its leading expertise, professional approach and independent consultancy, is staffed by client-orientated personnel who are not only highly skilled and educated, but also continuously strive to develop and improve their competencies. The NLR moreover possesses an impressive array of high quality research facilities.



NLR – Dedicated to innovation in aerospace

www.nlr.nl