Implementation of Goal Oriented Behavior Planning, Re-planning for SAF

Jungyoon Kim, Daehoe Choi
REALTIMEVISUAL Co.
Seoul, South Korea
jkim@realtimevisual.co.kr, choiday@realtimevisual.co.kr

ABSTRACT

We suggest a method which enables simulation entities to execute goal oriented behavior planning by dynamic behavior linking. Existing behavior processing methods especially in Semi-Automated Forces (SAF) are mainly based on rather strict plans which are given at the initial stage of simulation, thus they are not effective to cope with contingencies especially at the human in the loop simulation cases as human acts unexpectedly. Even worse, those methods usually suffer from explosion of behavior combination in attempt to describe all possible countermeasures which may be required in various simulation situations, and such combinations may be frequently prone to being inconsistent to the situations or may miss a certain inevitable behavior that an entity must execute. Our method is a way of generating a sequence of behaviors by retrieving behavior linkages from the goal in the manner of back-propagation. In this method we tag behaviors with behavior links (pre/post-conditions), thus simulation entities can be more consistent and more like human since an entity generates behavior links dynamically according to a certain contingency. This paper briefly shows the actual application of the method to the combat model for Korean military and discloses the partial test result.

ABOUT THE AUTHORS

Dr. Jungyoon Kim is a research engineer in REALTIEMVISUAL Inc., Seoul, Korea. He has been working on development of military logistics systems and joined the M&S area in 2010. He obtained BS in Aeronautical Engineering from Korea Air Force Academy, received MS. from Texas Tech University and Ph.D from Korea Advanced Institute of Science & Technology (KAIST) in computer science.

Mr. Daeheo Choi is an engineer in REALTIEMVISUAL Inc., Seoul, Korea. He has been working on M&S area since 2005, and has focused especially on Artificial Intelligence and architecture of simulation software. He obtained BS in Computer Science from Hong-Ik University, Seoul.

Implementation of Goal Oriented Behavior Planning, Re-planning for SAF

Jungyoon Kim, Daehoe Choi
REALTIMEVISUAL Co.
Seoul, South Korea
jkim@realtimevisual.co.kr, choiday@realtimevisual.co.kr

INTRODUCTION

War game is the tool to experiment and analyze various tactics or strategies in battlefield by modeling combat environment, so it has been popular because it is advantageous over the real world exercise with lower cost [1, 2, 3]. For the game it is important to build proper simulation in that the entities in the game must properly and correctly reflect the participants in a real battlefield. Recently as the techniques evolve, more autonomous and intelligent entities are preferred for more effective simulation with lower budgets, so such trend is replacing old practices in which humans control entities [4]. In other words, it is more important to implement model reflecting real combat situation more correctly, and thus to give entities autonomy and intelligence [3, 5, 6]. For that purpose, it is required to implement autonomous entity which can understand its mission and decide proper tasks to achieve the mission. Such autonomous entity also can figure out the situations and plan activities to cope with the contingencies [1, 4]. However, the entities in most existing war-games use finite state machine which simply conducts pre-planned activities rather than plans with tasks and activities by actively figuring out its situations and making decision required to complete its mission [5].

To overcome the limitation of finite state machine technique, we suggest an automatic planning technique in which an entity plans by arranging behaviors to cope with given situations. Our technique is a way to compose the mission of computer generated forces (CGF) with behaviors hierarchically. In the hierarchy, multiple tasks are specified for a single mission and multiple behaviors are composed to achieve a task. Each task or behavior includes precondition and post condition. Precondition is to check whether to initiate task or behavior, and post condition is to check the termination condition of the task or the behavior. Thus the CGF initiates task or behavior if the precondition is met, then it terminates if the post condition is met regarding it has achieved its goal. Pre and post conditions are expressed in logic expression for condition comparison. Now on the term ‘task’ is interchangeably used with ‘composite behavior’ meaning it includes multiple behaviors and the term ‘behavior’ means sometimes composite behavior or primitive behavior according to its context.

In reality, humans do neither count nor compare the number of enemy soldiers or tanks for a certain criterion to decide whether to start a certain action or not; but rather they consider just many or few. In this work, conditions are transformed into fuzzy facts, and then those facts are compared to mimic such human’s indefinite decision process. For example, the conditions are described as “if enemy soldiers are few” rather than “if enemy soldiers are fewer than three.” By doing so, it is possible to simplify condition descriptions and make an entity more like human who is indecisive.

Generally, more detailed level of behavior specification (high fidelity) enables more effective simulation, but it requires more effort for such specification and its overhead becomes heavier. Thus right decision for the proper level of model fidelity is important in war-game simulation. This paper introduces a behavior selection mechanism which uses fuzzy rule in order to enhance fidelity while alleviating overhead. The selection of the mechanism is to adjust planned behaviors to cope with a certain unexpected situation. The behaviors as the output from the mechanism are, for examples, “down when bombshells fall” which is not in the initially planned tasks but needed to reflect more natural reaction of entities that want to preserve troops, and “withdraw when our troops are almost annihilated” which is the counteraction against exceptional situation in which each entity has to decide whether to continue its mission. Moreover, even in case of the same task, the mechanism selects proper behavior through fuzzy rule when the behavior should be different such as different type of entity that would execute the behavior or different situation where the entity is surrounded.

This approach has been applied to a pilot project of CGF. We could find the feasibility of our approach with the pilot project in which we tested a scenario where red force and blue force are confronting. In the execution of the

scenario the entities counteract situations on the way to complete their given missions. We feel that it needs some functionality to make CGF more ideal, which are actively changing tasks according to the various circumstances, autonomous planning according to the change of situations to find proper solutions, or collaborating with other CGF to conduct tasks. Such functionality makes CGF to be more real world entities. However, as limitation of this work, we rather focus on automated planning technique as preliminary phase, and regard rest functionalities as further work.

In this paper, overall description of our automatic planning technique is in the 2nd section, and the experiment result with discussion is in the 3rd section in which the experiment scenario is explained with a simulation log result of a simple war-game. It concludes with the 4th section with brief comments on further works.

AUTOMATIC PLANNING TECHNIQUE OF CGF

To make CGF in war-game to be similar to the real combatant it is required to implement various functionalities of the combatant into the CGF. Among such functionalities, autonomous planning and decision making can be regarded as keys. Autonomous planning is to plan with required behavior to complete given mission and decision making is to properly counteract contingencies. This section introduces the autonomous planning technique which enables the CGF to arrange behavior to complete given mission, in which an entity automatically selects behaviors required for the arrangement. This paper focuses on the autonomous planning technique assuming other required functions are properly operating by other parts of the software.

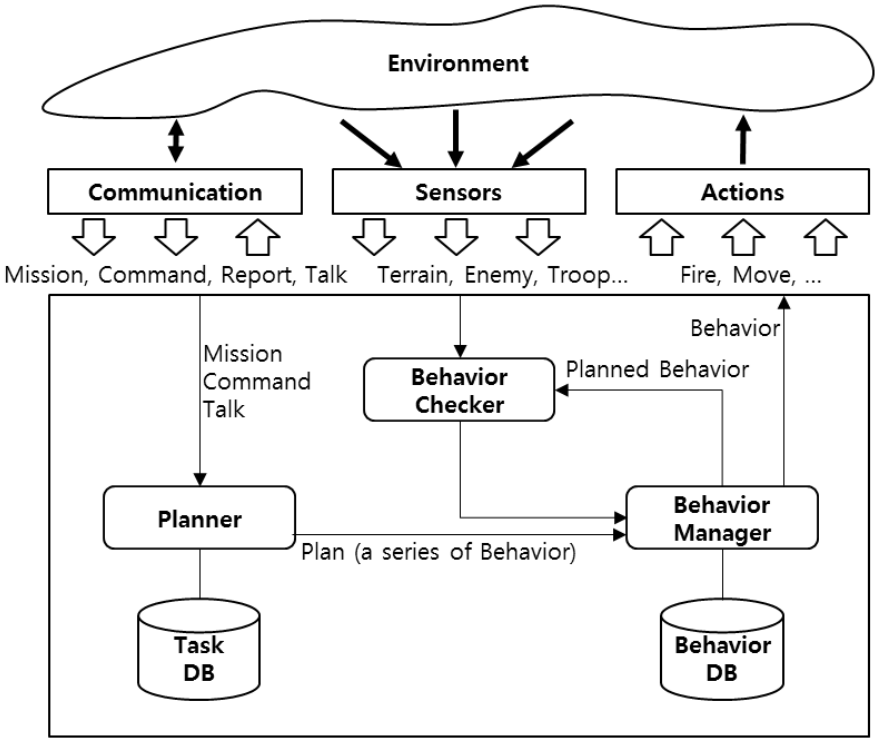

Figure 1. Architecture of planning technique

The overall architecture of the planning technique is shown in Figure 1. A CGF entity acquires information of external environment through sensors which can be a sort of detection functions. It also communicates through orders and reports with other entities which can be leaders, comrades, or subordinates. It should arrange and execute behaviors required to complete its given mission and cooperate with other entities whenever it is required to. To execute its behaviors, following three functions are identified.

- Function 1. Arranging behaviors according to situations.
- Function 2. Monitoring and counteracting situations actively and continuously.

- Function 3. Executing planned behaviors in cooperation with other entities.

Planner is for the Function 1. It generates sequences of behaviors for the mission given to an entity and such behaviors are prepared in Task DB. The Task DB is designed in similar way of the HTN: Hierarchical Task Network [7] which is noticeable in the area of existing automated planning technique. Behavior Checker is for the Function 2. It watches terrain, detects enemies, and monitors the friendly force's situations. It also checks entity's plan. Behavior Manager is for Function 3. It receives planned behaviors which are planned by Planner, evaluates the plan with Behavior Checker, then executes the plan or adjusted plan. However, each function is not implemented by a single part. Functions are performed in collaboration with multiple parts in the architecture.

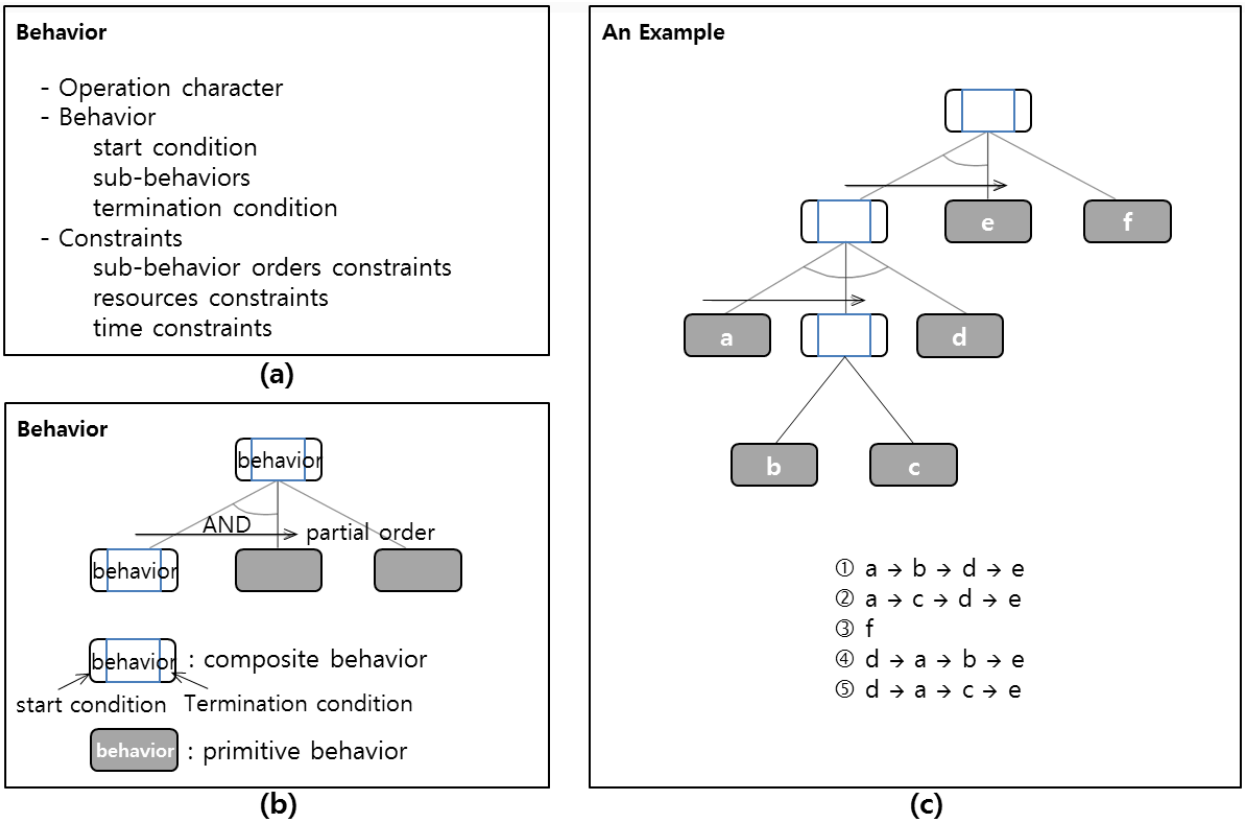

Figure 2. Behavior composition: (a) behavior elements (b) hierarchical behaviors (c) example

Figure 2 shows our way of behaviors arrangement with an example of possible outcomes. In Figure 2 (a) a behavior is composed of three parts. Those are the operation character (entity identifier) which will execute the behaviors, the information of the behaviors, and the constraints related to the behaviors. Constraints include the order of sub-behaviors required to complete the behavior, the resource limitation, and the time restriction in behavior completion. A behavior is tagged with start conditions (preconditions) and termination condition (post-conditions). (Hereunder pre and post-conditions refer to start and termination conditions respectively.) Sub-behaviors are also tagged with the same way, thus the structure of a behavior becomes tree as shown in Figure 2(b). Behaviors are categorized as composite behaviors which include sub-behaviors and as primitive behaviors which are atomic and have no sub-behavior. A few primitive behaviors include preconditions and post-conditions, but generally these do not include conditions so they are not shown in the figure. For sub-behaviors which are needed to be executed in a sequence the order of them can be defined as partial order (the arrows in the Figure (b) and (c)), or for mandatory behaviors 'AND' is tagged (the 'AND' in Figure (b)), or for the optional behavior 'OR' is tagged. An arc between behaviors means 'AND', otherwise (no arc) 'OR.' Figure 2 (c) shows five possible plans from an example tree.

The planning module generates proper plan under the consideration of given conditions at a certain situation. Under the situations in board-games such as chess or go it may not be a big problem in execution of the plan which is

generated by such module. However, under the situation of environment in which CGFs operate, combat conditions are dynamically changing thus simply executing the plan generated by the module may be improper or irrelevant.

The plan generated by Planner is transferred to the Behavior Manager in Figure 1. Prior to execution Behavior Manger transfers behaviors in the plan to Behavior Checker, then Behavior Checker checks whether required conditions are met to execute the relevant behaviors by comparing the precondition of the behavior to the information acquired from sensors.

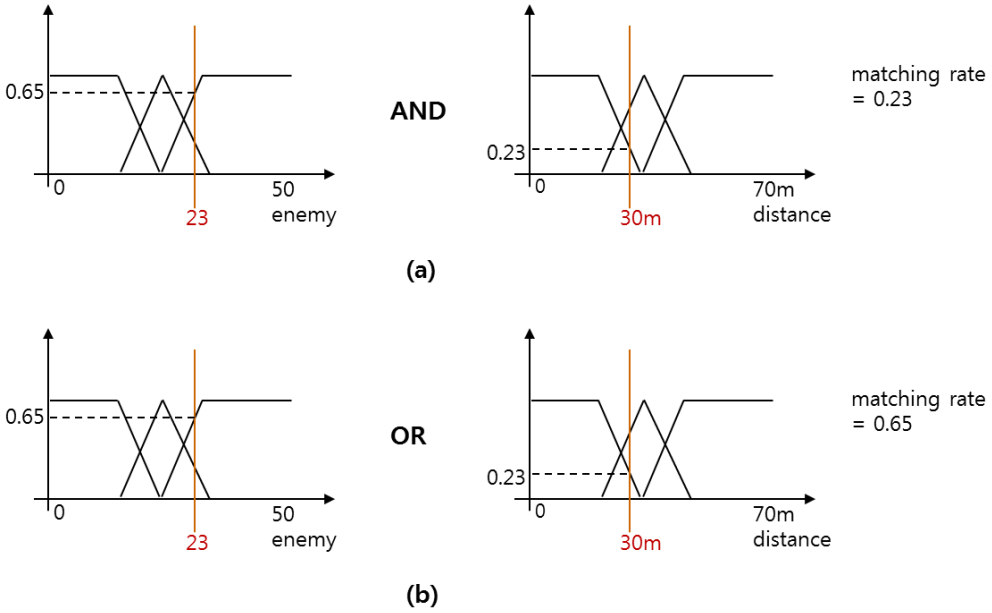

Figure 3. Matching rates of start and termination conditions (a) AND condition (b) OR condition

In Behavior Checker fuzzy matching [8] is adopted to check the precondition. Figure 3 shows such an example. In the example the precondition, "if the number of enemies is many and they are near," is defined as in the figure. When the number of enemies is expressed as "few, normal, many" and the distance as "near, normal, distant," and if the number of enemy soldiers is 23 and the distance is 30m, then the situation is evaluated as following. If the enemy number 23 is in "many" of level 0.65 and the distance 30m is in "near" of level 0.23, the matching rate value of the precondition, "enemies are many AND they are near," can be defined as the level 0.23 (as in Figure 3 (a)). If the condition, "enemies are many OR they are near," then the value becomes 0.65 (as in Figure 3 (b)). The information (charts) defined in Figure 3 is considered as domain knowledge. In this work only example domain knowledge is established. The complete knowledge can be obtained with proper activities of knowledge acquisition and engineering in cooperation with domain experts, but in this work it is out of scope.

Re-planning is the ability of CGF to counteract contingency by sustaining planned behavior and replacing it with an alternative. The process of re-planning is explained in Figure 4 with small example. This happens recursively until it reaches the primitive behavior as shown in Figure 2 or right one as shown in Figure 4, and then traces sustained behaviors back to the original (sustained) plan. In the figure Behavior Checker retrieves potential alternative behaviors if its precondition is not met, and then selects one of the alternatives. Suppose the threshold is 0.5 and the precondition is defined as (a), then Behavior Checker decides to continue current behavior. If the precondition is defined as (b) with the same threshold, then Behavior Checker will decide to sustain current behavior and retrieve behavior pool (Behavior DB) to find alternative which has similar post-condition. The comparison between pre and post-condition is done with logic expressions in normalized form (DNF). This process can be seen as more like the reflection of participant's decision making in real world battlefield.

The fuzzy matching between pre and post-conditions is done on linguistic terms, thus it can be simpler and more readable to humans. In fuzzy matching, a threshold value must be given to evaluate whether to execute the behavior.

For example shown in Figure 3(b), "concentrate fire," if the precondition is "enemy is many or distance is near" and the given threshold value is 0.6, then since the value of matching value is 0.65 the evaluation triggers the behavior. However in the course of the way, beware that it is required to articulate whether the interval of such threshold value is proper and the value reflects human decision in properly similar way. If not, the CGF entity's decision is far different from the decision that an actual human generally decides. It is not easy to find a general way of selection for the interval and the threshold value. Usually it seems to be done by repetitive modification through experiments after preliminary selection is performed depending on subject matter experts' experiences.

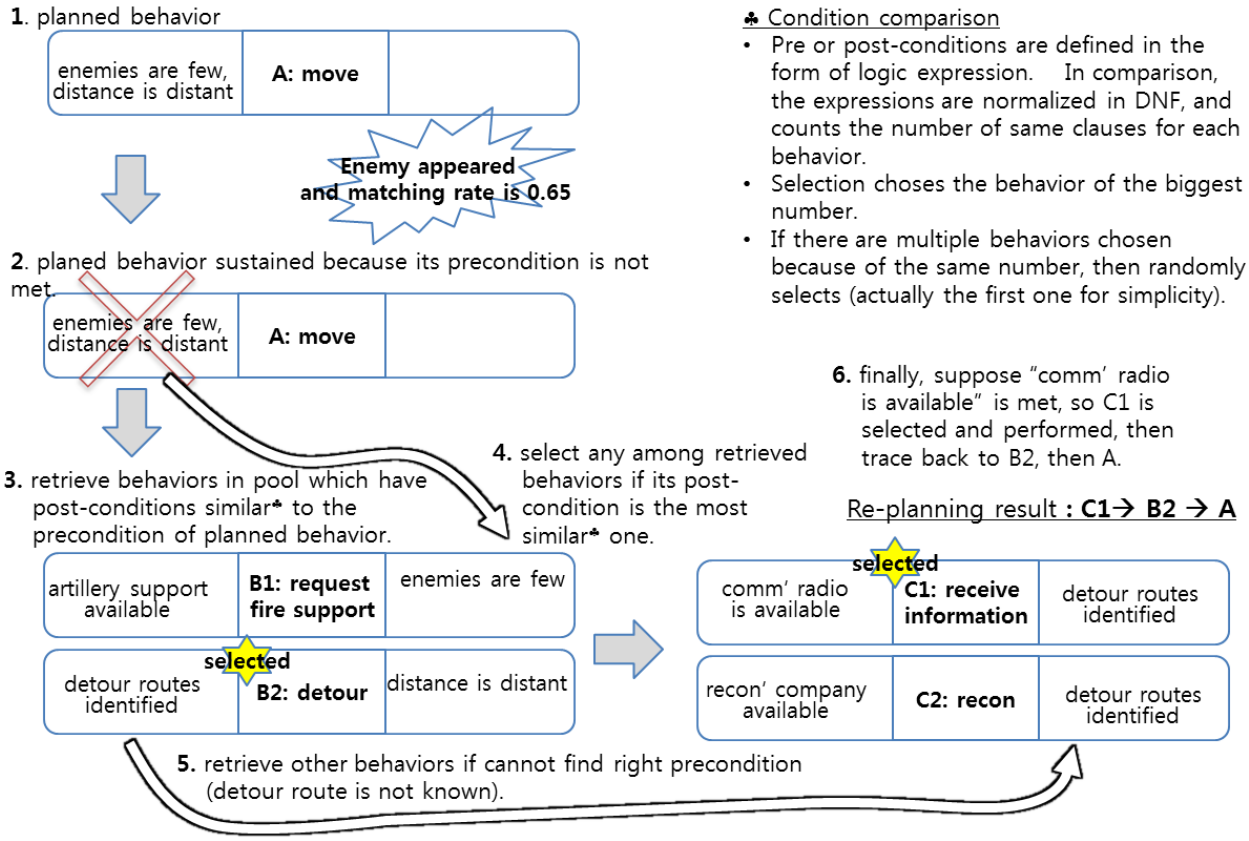

Figure 4. Example of re-planning

Once Behavior Manager finally decides a right behavior, then it transfers the selected behavior to the relevant physical module which will launch the required action of the entity. As an example of "enemies are almost annihilated" for concentrate fire, suppose the matching value is 0.87 (the number of survived enemies at the time may be two) and the threshold value is 0.8, then Behavior Manager terminates the behavior, "concentrate fire," then may launch the next behavior in plan, such as "tactical move to acquire another target."

At this point, some questions arise. How should entity do if the precondition of the behavior is met at the time of the initiation but the precondition is no more met during the execution? Even though, should it continue the behavior to meet the post-condition of it? Such questions may arise from modeling view, but it seems better to plan another new behavior for simulation which reflects more of real world. In this respect, we adopt the way of fuzzy rule to decide whether to continue the current behavior or to launch another behavior. For example, suppose the precondition, "there is no enemy and friendly troops are enough to acquire target" for the behavior "tactical move to acquire target," was met at the time of the behavior initiation. After some time passed if the friendly force was under enemy's bombardment during tactical move so its troops severely diminished with heavy casualties, it will not be reasonable to continue the "tactical move." In this work, through the fuzzy rule Behavior Manager can select another behavior for such an abrupt situation change.

The counteraction by simple rule may be enough for a case in which the situation is simple or the counteracting unit is small. However, if the situation is complicated or the counteracting unit is big including significant number of entities or types of entity, simple rule is not enough. In that case the result of a simulation which has to be a more real reflection of the real world becomes limited or useless meaning. Suppose the 3rd Platoon of the 1st Company is under enemy's bombardment when multiple platoons in the company are in tactical move, and each platoon in the company selects counteraction (behavior) through simple rule. In such case each platoon will decide its own behavior irrelevant from others' or the entire platoons do the same decision which is somewhat unreasonable in real world. In this kind of case, to reflect the real world in higher similarity, it is required that cooperation among platoons and systematic counteractions between upper unit and subunits must be done. To acquire such features, Goal Oriented Action Planning (GOAP) is indispensable in which flexible planning can be done keeping given goal until its completion, and also situation report from subunit (platoon leader) to upper unit (company commander) and new task allocation to the subunit must be possible [9,10,11]. However, the suggested method in this paper has been implemented in partial as a preliminary version of it.

EXPERIMENT AND RESULT

We have implemented the proposed technique, GOAP, into a war-game and performed test. All parts of the war-game other than GOAP are reused from existing modules. The war-game can simulate not only combatants but also various vehicles, but in this experiment mainly combatants and a few tanks are playing for simplicity.

Figure 5 shows Scenario Editor editing the scenario for the experiment. In the figure, boxes with cross are combatants and they are gathered in three groups which represent three blue platoons positioned at the lower left corner. Also the red boxes represent the red force and they are grouped in two platoons positioned at upper center. There are number of tanks behind red force which compose a tank platoon at the opposite direction of blue force, and there is also a blue force tank platoon behind blue force but not shown in the figure. The grey lines are the routes for blue force's tactical move and the stake shaped icons on the lines represent way points.

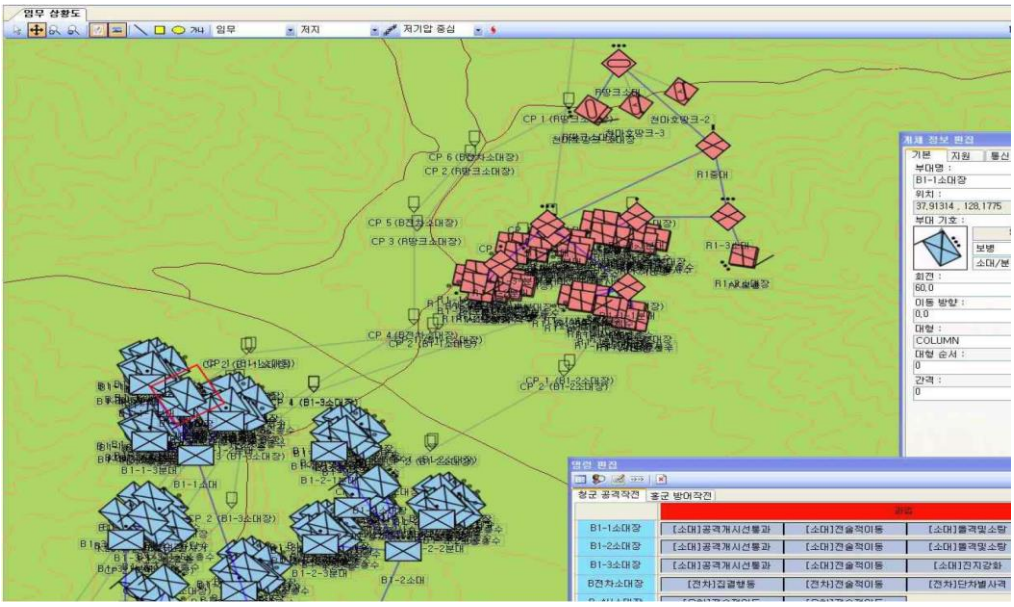

Figure 5. An experimental scenario made by the scenario editor

The situation in the figure is that the given mission to the blue force is to occupy the post on the hill held by red force, and the mission to the red force is to defend any attack. The lower right pane shows rows and columns. Each row defines a unit (platoon) and its phases meaning a unit's behaviors in sequence to complete its mission. Each column represents the phase of a scenario. From the first to the third rows, the plan for the missions of the first, the second, the third Platoon of blue force is depicted as 1) pass the attack commencement line, 2) move tactically, 3) charging and sweeping. In the third row, the assisting plan of the 3rd Platoon is 1) pass the attack commencement

line, 2) move tactically, 3) securing occupied post (no charge and sweep). Each platoon conducts planned behaviors which are stored in the Task DB.

If there is any sub-behavior under a behavior, various sequences of sub-behaviors can be planned according to the conditions of sub-behaviors as shown in Figure 2. However, a behavior such as 'tactical move' is quite simple so it commences required action ('move') directly without any sub-behavior. In case of 'move,' it requires path finding when an entity or a unit encounters unexpected situations such as detecting not known minefield on the planned route or realizing a bridge collapsed that is supposed to stand there. Such cases require sustaining planned route and need to find detour. Such path finding is sort of sub-function needed for move in behavior planning and executing, so it is not required to be in the planning module.

As explained in Section 2, a behavior model includes various alternative behaviors available according to situations. Figure 6 shows such model with three alternatives. If Behavior Checker decides it is not appropriate to continue "move,' Behavior Manager terminates 'move' and commences another sub-behavior among three different options.

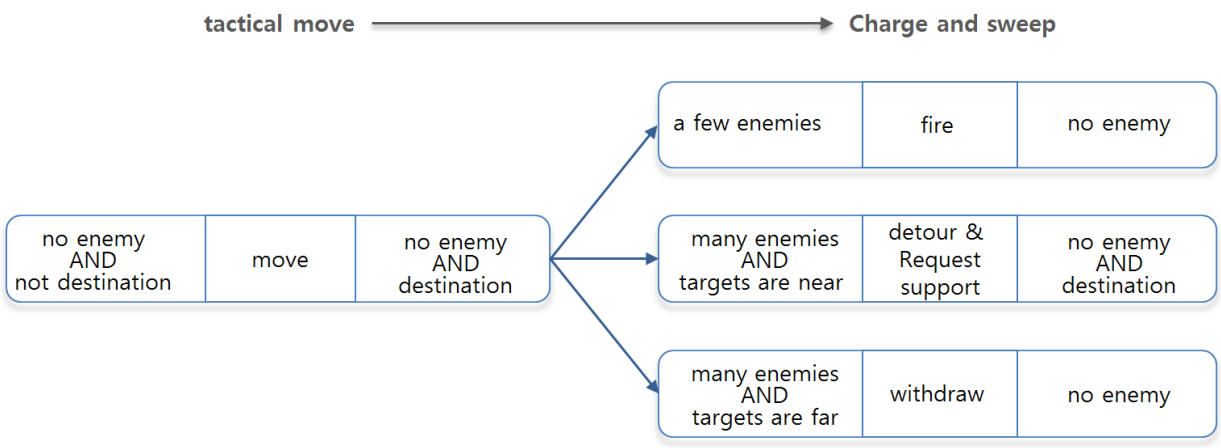

Figure 6. An example of behavior model

Figure 7. Execution example

Experiment is performed to test all the three cases by adjusting thresholds. Figure 7 shows a screen capture of the log of simulation for the second case, "many enemies AND targets are near," in Figure 6. All three cases in Figure 6 are logged in Koreans and those are translated in Table 1. As shown in the table, we could observe the results that entities counteract unexpected situations on the way of executing planned behaviors. Figure 8 shows the 3D display for the war-game simulation. A simulation supervisor can figure out the CGFs' motions with the 3D display and observe the battle situations with more reality. The situation in the figure is that the blue force unit detected enemies so the members of the unit are preparing counteractions. The red colors texts on the hill and in the village imply the red force enemies in hiding. The supervisor also can watch battle situations with various angles and heights of viewpoint.

We could find the feasibility of our approach through this experiment in that it can facilitate scenario editing with various cases of manpower (number of troops), tactics, firepower, and positions of blue force or red force. Simulation analysis has been performed with After Action Review support tool but the detail is omitted here because it is out of issue.

Table 1. Simulation logs for situations in Figure 6

Situations in Figure 6	Blue force	Red force
a few enemies	[00:00:01] Simulation start [00:00:02] [Behavior commencement] move [00:00:24] [Situation] Enemy detected (# of enemies is small) [00:00:24] [Behavior termination] move [00:00:24] [Behavior commencement] fire [00:00:41] [Situation] no enemy [00:00:41] [Behavior termination] fire [00:00:41] [Behavior commencement] move [00:01:12] [Behavior termination] move [00:01:45] Simulation end	[00:00:01] Simulation start [00:00:02] [Behavior commencement] alert [00:00:40] [Situation] casualties occurred (dead) [00:01:45] Simulation end
many enemies AND targets are near	[00:00:01] Simulation start [00:00:02] [Behavior commencement] move [00:00:38] [Situation] Enemy detected (# of enemies is many && distance to targets is near) [00:00:38] [Behavior termination] move [00:00:38] [Behavior commencement] detour [00:00:38] [Report] request support [00:01:52] [Situation] no enemy [00:01:52] [Behavior termination] detour [00:01:52] [Behavior commencement] move [00:02:14] [Behavior termination] move [00:02:31] Simulation end	[00:00:01] Simulation start [00:00:02] [Behavior commencement] alert [00:02:31] Simulation end
many enemies AND targets are far	[00:00:01] Simulation start [00:00:02] [Behavior commencement] move [00:00:34] [Situation] Enemy detected (# of enemies is many && distance to targets is far) [00:00:34] [Behavior termination] move [00:00:34] [Behavior commencement] withdraw [00:01:55] [Behavior termination] withdraw [00:02:11] Simulation end	[00:00:01] Simulation start [00:00:02] [Behavior commencement] alert [00:02:11] Simulation end

CONCLUSION

We have suggested an automatic planning technique for autonomous behavior of CGF in war-game. On one side, the method is somewhat similar to the hierarchical task network method which is widely used in existing automatic planning. However, on the other side, it is advantageous in more real environment for CGF giving an entity ability of flexible behavior selection by tagging behaviors, while existing one defines behavior with effect of the behavior. In other words, the hierarchical task network fits well in the cases that a definitive effect follows a behavior once it is executed, while our automatic planning technique fits better in the cases that the following effect of a behavior is not definitive as most problems in real world battlefields are not definitive. For example, when ‘fire’ behavior is executed, the technique of hierarchical task network describes ‘enemy is killed’ as a result of the ‘fire’ behavior, and it considers that the behavior and its result are definitely occurred without checking if the enemy is actually killed. On the contrary, after ‘fire’ behavior is executed, the automatic planning technique checks the post condition ‘enemy is killed’ and if the condition is not met, then it continues the ‘fire’ behavior. This is more like real situation.

We have implemented and tested the suggested technique on a war-game with a simple scenario, and we showed that it enables a simulation entity to counteract unexpected situations with automatic planning. However, as its limitation this technique is needed to be refined more for actual usage and it lacks the ability of cooperation among multiple CGFs. Such limitation and cooperation issues will be further contemplated.

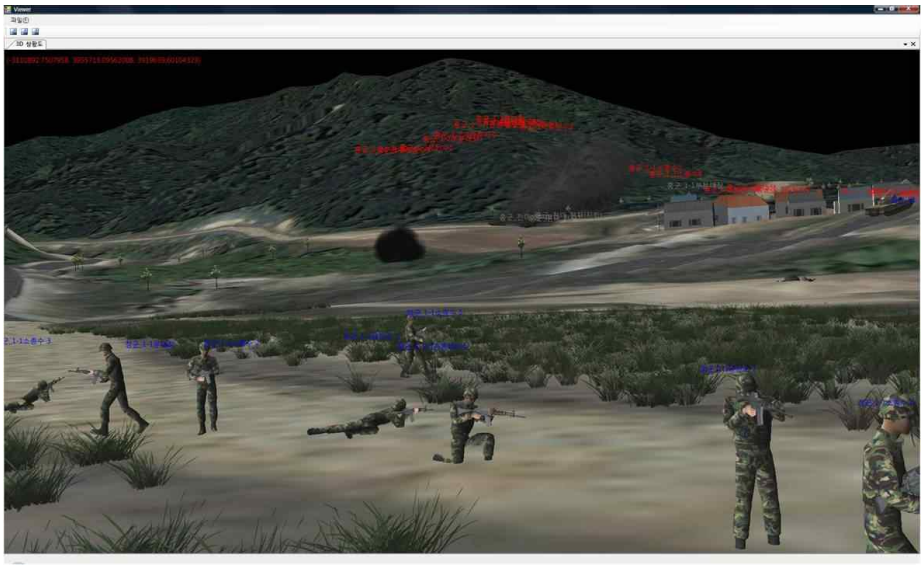

Figure 8. 3D screen of simulation

ACKNOWLEDGEMENTS

This work has been supported by the Agency for Defense Development (ADD), Korea, under the contract No. UC120064PD.

REFERENCES

[1] A. Ozaki, M. Furuichi, K. Takahashi, and H. Matsukawa, "Design and Implementation of Parallel and Distributed Wargame Simulation System," in Proceedings of IEICE/IEEE Joint Special Issue on Autonomous Decentralized Systems and Systems Assurance, vol. E84-D, pp. 1376-1384, 2001.

[2] P. Pearce, A. Robinson, and S. Wright, "The Wargame Infrastructure and Simulation Environment (Wise)," in Proceedings of 7th International Conference of Knowledge-Based Intelligence Information and Engineering Systems, 2003. Oxford, UK.

[3] D. Fu, R. Jensen, and R. Houletter, "Specifying the Behavior of Computer-Generated Forces without programming," in Proceedings of the 2003 Winter Simulation Conference, pp. 969-975, 2003.

[4] Hyun Kyoo Park, YG Kim, YS Park, "Development of The Distributed Real-Time Wargame Simulation Using TMO Model," Proceedings of the 98 Korea Society for Simulation conference. pp. 13-22, 1998.

[5] S. B. Banks, "An Approach to Enhance Human Behavior Modeling for Computer-Generated Actors," in Proceedings of the 4th International SIMTECT Conference, pp. 199-204, 1999. Melbourne, Australia.

[6] D. S. Nau, "Current Trends in Automated Planning," AI Magazine, vol. 28, no. 4, pp. 43-58, 2007.

[7] K. Erol, "Hierarchical Task Network Planning: Formalization, Analysis, and Implementation," Phd thesis, Department of Computer Science, University ofMaryland, College Park,MD, 1996.

[8] B. Kosko, Neural Networks and Fuzzy Systems. ADynamical Systems Approach toMachine Intelligence. Prentice-Hall, 1992.

[9] J. Orkin, "Applying Goal-Oriented Action Planning to Games," in AI Game Programming Wisdom 2, pp. 217-228, 2003. Charles River Media.

[10] E. Long, "Enhanced npc behavior using goal oriented action planning," 2007.

[11] Han-ha Yoo, Kyung-eun Cho, and Ky-hyun Um,"A Cooperation Strategy of Multi-agents in Real-Time Dynamic.