On the Role of Explanation for Hierarchical Case-Based Planning in Real-Time Strategy Games

Héctor Muñoz -Avila[1] & David W. Aha[2]

[1] Department of Computer Science & Engineering;
Lehigh University; Bethlehem, PA 18015
[2]Navy Center for Applied Research in Artficial Intelligence;
Naval Research Laboratory (Code 5515); Washington, DC 20375
munoz@cse.lehigh.edu   david.aha@nrl.navy.mil

Abstract. Explanations can play a key role in case-based planning systems. We describe an application of hierarchical case-based planning that involves reasoning in the context of real-time strategy games, describe a representation for explanations in this context, and detail four types of explanations.

1 Motivation

The capability of a knowledge-based system to provide a meaningful explanation of its actions is a crucial factor affecting the acceptance of the system (Majchrzak & Gasser, 1991). Several approaches have been proposed to provide meaningful explanations (Doyle *et al.*, 2003). One such approach is case-based explanation, in which cases themselves are used as the explanation for decisions made by the system. This approach is particularly useful for case-based reasoning systems that perform analysis tasks, such as diagnosis (Cunningham *et al.*, 2003). Despite these advances, the role of explanation in case-based planning systems is not well understood.

In this paper we investigate the role of explanation for hierarchical case-based planning systems in the context of real-time strategy (RTS) games. In such systems, episodic knowledge (cases) can be used to model the behavior of a computer player. Cases describe strategies to achieve gaming tasks. The role of case-based planning for such systems is to develop a winning strategy. The capability of a system to generate meaningful explanations in this context is important for the following reasons:

- *Allowing users to interrogate the behavior of a computer player*. For example, a user may want to know why the system selects a particular strategy. In this example, a computer player (i.e., an automated player) may be biased towards producing more cavalry units than archer units. But what is the rationale for the system's bias?
- *Explaining what caused/lead to the current state*. What caused the game's outcome? For example, suppose that the computer player attacks a city but fails to capture it. What circumstances in the game world lead to this failure?
- *Explaining the motivation for knowledge/reasoning refinement*. Is it possible to refine the conditions and strategies encoded in the cases? If so, can the system explain the rationale for a change or at the least the rationale for suggesting that a change is needed even without specifying its details?

- *Preventing the learning of unrealistic/non-doctrinal behaviors*. Games follow rules that must be observed by the computer player. For realistic game simulations (e.g., war gaming), these include rules of physics (violating these would yield unrealistic behavior) or explicitly stated rules (in the context of war gaming, these rules are called *doctrine*). Is it possible to construct an agent that observes a computer game, identifies when the computer opponent (i.e., an automated opponent) is violating such rules, and explains how it is violating these rules?

The next section describes the background for our investigation. Section 3 then describes the representation we use for explanations in hierarchical case-based planning systems. Section 4 describes the kinds of explanations that may occur in the context of RTS games. Finally we discuss related and future work in Sections 5 and 6.

2 Background

In this section we describe the background for our work, which includes an architecture for integrating reasoning systems with gaming engines (TIELT), a specific gaming engine (Stratagus), and our representation for plans (Hierarchical Task Networks).

2.1 TIELT

Our investigation takes place in the context of TIELT (Figure 1), a testbed for investigating and evaluating learning techniques that we are developing for the DARPA Information Processing Technology Office's thrust in cognitive systems (Aha & Molineaux, 2004). TIELT is being designed to:

- Facilitate the empirical investigations of learning techniques in gaming simulators by AI (e.g., machine learning, cognitive systems) researchers.
- Permit developers of commercial and military gaming simulators to assess the utility of learning techniques and learned behaviors for selected tasks in their simulators.
- Support one or more DARPA challenge problems in machine learning.

TIELT is being designed as middleware for integrating game engine simulators and learning-embedded reasoning systems. We are targeting it to support multiple game genres, including RTS, discrete strategy, role-playing, team sports, and action games. Learning techniques will normally have to be embedded in reasoning systems that perceive (processed) states of the simulator, can simulate decision making, and can transmit decisions as effector actions. For example, implementations of cognitive architectures can support these capabilities, among others (Langley & Laird, 2002).

Integrations with TIELT require defining the five knowledge bases shown near the bottom of Figure 1. Experts on the game engine, and a selected game defined in it, will develop the Game Interface and Game Model Descriptions, while the reasoning system's expert will define the Reasoning Interface Description. The learning and performance Task Descriptions will be selected from the Game Model Description. Finally, the Empirical Methodology Description will define what control messages to

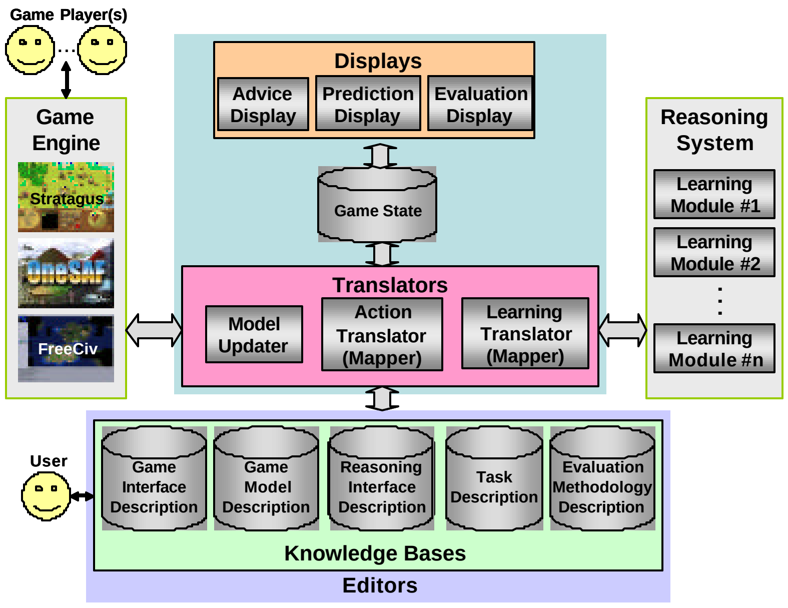

Figure 1: TIELT's integration architecture

send to the game engine and reasoning system to conduct the user's investigation. By accumulating these knowledge bases over time, researchers and simulation developers will be able to more easily include multiple games and learning-embedded reasoning systems in their investigations.

TIELT will provide reasoning systems with access to controllable objects, agents, and processes in the game engine. In addition, it will support such tasks as predicting an opponent's strategic and reactive plans, posting alerts, and updating (a possibly incomplete and/or incorrect) Game Model Description. There are multiple opportunities for studying the roles of explanation in TIELT. We focus on one such role that involves case-based planning and an RTS game engine.

2.2 Stratagus

Stratagus (2004) is a free RTS game engine. RTS games are a genre of computer strategy games where the objective is to accomplish some game-winning condition in real time in a simulated world. The most typical winning condition is to destroy all the enemy's forces. For some games, this involves developing an economy to support the purchase of units and installations, constructing installations to produce and upgrade the units, and using these units to fight the enemy. Currently, nine games have been defined using Stratagus. Figure 2 displays a screen for *Magnant*, a game in which the units are simulated ants of various types including soldiers and flying ants.

Figure 2: Snapshot of the Stratagus/Magnant game world

Stratagus is implemented primarily in C, and is separated into modules with a common *include* directory. To encode the computer opponent, the popular Lua scripting language is used (Lua, 2004). Lua combines simple procedural syntax with powerful data description constructs based on associative arrays and extensible semantics.

2.3 Hierarchical Task Networks

HTNs (Hierarchical Task Networks) are a formalism for representing hierarchical plans. HTNs refine high-level tasks into simpler tasks (Erol *et al.*, 1994). In the context of Stratagus, high-level tasks indicate complex goals such as *CaptureCity(X)*, where X is a city that belongs to an enemy civilization. Low-level tasks include simpler tasks, such as capturing a road, to concrete actions, such as bombarding the city with a specific artillery unit. Tasks representing concrete actions are called *primitive* because they cannot be decomposed into other subtasks. *Compound* tasks are tasks that can be further decomposed into simpler subtasks.

Formally, an HTN (Erol *et al.*, 1994) is a set of tasks and their ordering relations, denoted as $N=(\{t_1,\ldots,t_m\},<)$ $(m\geq 0)$, where $<$ is a binary relation expressing temporal constraints between tasks. One of the most important properties of HTNs is that HTNs are strictly more expressive than STRIPS representations (Erol *et al.*, 1994).

Following the conventions of the SiN algorithm for HTN planning (Munoz-Avila *et al.*, 2001), tasks can be decomposed using methods and cases, which encode strategies for accomplishing compound tasks. Methods capture general problem-solving knowledge about the domain whereas cases capture episodic problem-solving knowledge. For this paper, we discuss cases only so as to focus on explanations for episodic knowledge. A ***case*** is an expression of the form *C*=(*h*,*P*,*ST*,*Pref*), where *h* (the case's *head*) is a compound task, *P* is a set of *preconditions*, *ST* is the set of *C*'s (children) *subtasks*, and *Pref* are *preferences*. A method is *applicable* for decomposing a task if its preconditions are valid in the current world state. Preconditions differ from preferences in that, while preconditions are necessary conditions to determine a method's applicability, preferences indicate desirable conditions for selecting a case, and are used to select among multiple applicable cases for a task (i.e., select a case with the highest percentage of preferences fulfilled).

Head: *CaptureCity(X)*
Preconditions:
    Road(R)
    Connects(R,X)
    TaskForce(TF)
Subtasks:
    CaptureApproachRoad(R,X,TF)
    AssaultCity(X,TF)
Preferences:
    InProximity(TF,X)

Figure 3: Example of a case for accomplishing the task *CaptureCity(X)*

Figure 3 shows an example of a case to achieve the task *CaptureCity(X)*. This case calls for using a task force (TF) to capture an approach road to the city and assaulting the city by the same force. The preconditions of the case are that there is a road (R) that connects to X and that a task force is available. The case's only preference condition states that the task force is in the proximity of the city. It is not necessary to fulfill this condition but it is desirable.

An important characteristic of HTN planning is that applying a case to achieve a task ***does not change the state of the world***. Compound tasks represent high level goals and cases capture strategies to achieve (decompose) them. Changes in the world occur only when operators accomplish primitive tasks, and HTN operators differ from standard STRIPS operators in that they have no preconditions, only effects. The reason is that the actual preconditions are evaluated in the cases. When primitive tasks are reached, the strategy has been selected and it is executed by performing concrete actions (i.e., the operator's effects). Formally an operator is an expression of the form *O=(h,effects)*, where *h* (the operator's *head*) is a primitive task, and *effects* indicate how the world changes. The set of actions obtained after decomposing all tasks form the plan that transforms the world state to achieve the high level tasks.

3 Representing Explanations

We view explanations as a collection of annotated plan elements. In the context of hierarchical case-based planning, an explanation is a collection of cases' preconditions and preferences, tasks (compound and primitive), and actions. Figure 4 illustrates an abstract task hierarchy, an abstract plan, and an abstract explanation. The task t is the top-level task to be accomplished. Two alternative cases can decompose t ($case_1$ and $case_2$). Case $case_1$ is selected and two new subtasks (t_{11} and t_{12}) are generated. The task decomposition process continues until tasks t_i and t_k (and

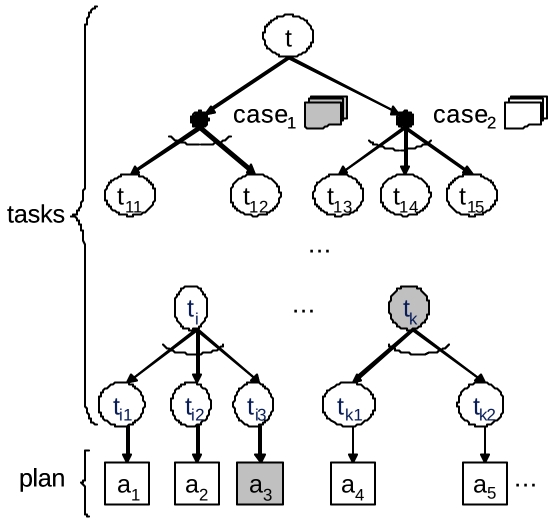

Figure 4: Illustration of an explanation in an HTN decomposition process

perhaps others) are generated. These in turn are decomposed into the primitive tasks t_{i1}, t_{i2}, t_{i3}, t_{k1}, and t_{k2}, which are achieved by actions a_1-a_5. Syntactically, an explanation is a subset of these elements. Figure 4 illustrates an explanation (highlighted) consisting of the preconditions and preferences of case case$_1$, task t_k, and action a_3. In Section 4 we will show concrete examples of explanation.

As we saw in the Section 2, HTNs are a natural representation of the AI opponent's strategies and as such are intended to be modeled in TIELT's Game Model Description. Explanations, on the other hand, are meta-inferences on the Game Model that are to be shown to the user. Thus, explanations are intended to be part of TIELT's Reasoning Interface Description, which is the module responsible for interfacing the results of an integrated reasoning system.

4 Explanation Types

In Section 3 we identified the syntactic components of an explanation. In this section we identify and describe the types of explanations that are of interest for a case-based planning system in the context of RTS games. These types of explanations are:

- Strategy selection
- Course of action outcome
- Game model update
- Prediction

Strategy selection refers to explanations for why a particular strategy is selected. In the context of HTN case-based planning, strategies are captured in cases. As a result, strategy selection indicates the conditions for retrieving a particular case. As an example, consider a gaming task in Magnant such as taking a city, the AI program can have several cases that are applicable for accomplishing this task. Table 1 shows two alternative cases. The first case is the same as the one shown in Figure 3. The second case calls for a 3-stage strategy: capture all access roads to the city by a task force (TF1), bombard the defenses until they are weakened using a bombarding force (BF), and assault the city using another task force (TF3).

When observing the behavior of the computer opponent, a user may request an explanation for why a particular strategy was selected to capture a city. In this situation, the explanation consists of alternative cases, their preconditions and preferences, and how they matched the current situation (e.g., which preconditions and preferences are fulfilled in the state of the world where the choice was made).

The case matching all preconditions and with the highest number of preferences fulfilled is the one selected. As an example, the second alternative may be selected if the two task forces (TF1 and TF3) and the bombarding force (BF) are available and are in the proximity of city X (even though both cases have all their preconditions satisfied). Strategy selection explanations are similar to the explanations for case-based reasoning (CBR) systems that perform analysis tasks, particularly diagnosis (Cunningham *et al.*, 2003). Essentially, strategy selection explanations return the selected case itself as the explanation. The main differences between strategy selection and explanation in CBR systems that perform analysis tasks are that (1) the explanation includes alternative cases with information on how they matched the current situation and (2) strategy selection explanations are chained to the explanations of each of its task's subtasks and parent tasks. For example, one of the alternative cases is selected to accomplish the task *CaptureAccessRoads(X,TF1)*. The chained explanations form a hierarchy similar to the one depicted in Figure 4. This kind of explanation relates to retrieval because it explains why a case was selected.

Table 1: Alternative cases for accomplishing the task *CaptureCity(X)*

Case 1	**Case 2**

Head: *CaptureCity(X)*
Preconditions:
Road(R)
Connects(R,X)
TaskForce(TF)
Subtasks:
CaptureApproachRoad(R,X,TF)
AssaultCity(X,TF)
Preferences:
InProximity(TF,X)
	Head: *CaptureCity(X)*
	Preconditions:
	TaskForce(TF1)
	…
	Subtasks:
	CaptureAccessRoads(X,TF1)
	BombardCityUntilWeak(X,BF)
	AssaultCity(X,TF3)
	Preferences:
	InProximity(TF1,X)
	InProximity(BF,X)
	InProximity(TF3,X)

Course of action outcome refers to explanations of the outcome from pursuing a particular gaming strategy. In the context of CBR this refers to explanations of the results of reusing a case. As an example, suppose that Case 2 in Table 1 was selected to capture a city. If the AI player (i.e., the player under the control of our AI system) cannot take the city, an explanation is needed. For example, the reason for this failure might have been the inability to secure access roads as a result of geographical conditions (e.g., natural obstacles such as rivers), which lengthens the execution time required by the task and permit s the enemy's reinforcements to arrive before the task can be completed. This explanation can be used to refine the cases by adding preconditions or preferences for selecting them. Thus, the explanation consists of a list of tasks and actions that could not be accomplished. These kinds of explanations can be used to update TIELT's Game Model Description because they highlight potential problems with the current model.

Game model update refers to a justification for refining the gaming strategies. In the context of CBR this refers to explanations for modifying the case's tasks and subtasks or modifying the case adaptation process. As an example, consider a variant of Case 2 calling for replacing the first subtask with the task *CaptureSupplyRoads(X,TF1).* An explanation for this change of strategy captured in the case is that there is no need to capture all roads, but only those that are used by the enemy to supply the city.

Finally, *prediction* refers to explanations about foreseeable outcomes from gaming strategies or conditions. For CBR this may refer to (1) predictions on the feasibility of the preconditions and (2) predictions about the outcome for reusing a case. This kind of explanation can be used to refine the retrieval strategy. As an example of the first kind of prediction consider the condition of having a task force available for Case 1. If, for example, a task force is located in the proximity of the city and it is not currently engaged in fighting an enemy force, the agent may predict that the force can be made available to satisfy the preconditions of Case 1. As an example of the second kind of prediction consider the possible outcome of Case 1. If there are no enemy units close to the city, the agent may predict that reusing Case 1 will be successful. Such predictions should be taken into account when considering the retrieval of this case. While the first kind of prediction can be t aken into account when measuring similarity, the second kind of prediction should be used as a filter for retrieving only those cases whose similarity has been deemed sufficient for retrieval purposes.

Table 2 summarizes the different kinds of explanations, the syntactic components of these explanations, and their associated phases of the CBR cycle.

Table 2: Comparison of the different forms of explanations

Explanation	**Syntax**	**CBR cycle**
Strategy selection	Case's preconditions and preferences	Retrieval
Course of action outcome	Tasks and actions	Reuse, Retrieval
Model update	Cases	Reuse, Retain
Prediction	Case's preconditions and preferences	Retrieval

5 Related Work

Our definition of explanation (i.e., a collection of cases' preconditions, preferences, tasks, and actions) is inspired in part by the notion of justifications in the Redux system (Petrie, 1991). Redux is a justification truth-maintenance system (JTMS) for processing planning contingencies. Redux implements a data structure called the *Goal Graph* to represent relations between plan elements. Justifications are defined as a collection of objects in the data structure. Redux has helped users manage software process models, where justifications are used to explain dependencies between process components (Dellen *et al.*, 1997). From an abstract point of view, our notion of explanation mimics the notion of justifications as a collection of plan elements. The main difference is that we don't need to construct an ad-hoc structure like the goal graph because hierarchical plans play a similar role to the goal graph. Namely, it relates tasks, subtasks, and the methods and operators for achieving them.

Several researchers argue that analyses for explaining case retrieval should present both supporting and disconfirming evidence (e.g., Ashley, 1990; Murdock *et al*., 2003; McSherry, 2003). For strategy selection, this translates to explaining why some subtasks and actions were chosen rather than others, and how to improve the selected strategy's match (e.g., by matching preconditions and preferences more closely).

Doyle et al. (2003) reviewed several CBR systems that support explanation. From the various systems reported there, MoCAS's multi-level explanations (Pews & Wess, 1993) somewhat resemble our chained explanations for strategy selection. In MoCAS the multi-level explanations reflect levels of abstraction of facts, whereas explanation chains in strategy selection relate high-level tasks with simpler tasks.

More recently, Roth-Berghoffer (2004) studied foundational issues of explanations in CBR, where he noted that "the more elaborated a model is, the more explanatory power it has." Strategy selection explanations are derived from the HTN model used for plan generation. As described in Section 2.3, this model is itself elaborated. For this reason, we believe that the model is sufficiently expressive to provide meaningful explanations. Furthermore, strategy selection explanations can be classified as cognitive explanations because they describe why a CBR system obtained its results.

6 Future Work

We will continue our work on using hierarchical representations to model behaviors of AI opponents and the automatic generation of explanations in Stratagus games. In addition to Hierarchical Task Networks (HTNs), we will investigate the use of Task-Method-Knowledge Language (TMKL) (Murdock, 2001) for representing models and explanations. While HTNs and TMKL both represent tasks and methods, TMKL provides a more expressive language; it explicitly represents loops. For example, TMKL can easily represent the statement *repeat action X until condition Y holds*. While such statements can be represented in HTNs, doing so would be cumbersome. However, in contrast to TMKL, the semantics of HTNs are well-defined and understood. We plan to take an intermediate approach, starting with HTNs and adding characteristics of TMKL as needed to model AI opponents.

Acknowledgements

Thanks to our reviewers for their helpful comments. This work was funded by DARPA's Information Processing Technology Office and NRL.

References

Aha, D.W., & Molineaux, M. (2004). Integrating learning in interactive gaming simulators. To appear in *Challenges in Game AI: Papers of the AAAI'04 Workshop* (Technical Report WS-04-04). San José, CA: AAAI Press.

Ashley, K.D. (1990). *Modeling legal argument: Reasoning with case and hypotheticals.* Cambridge, MA: MIT Press.

Cunningham P., Doyle D., & Loughrey, J. (2003). An evaluation of the usefulness of case-based explanation. *Proceedings of the Fifth International Conference on Case-Based Reasoning* (pp. 122-130). Trondheim, Norway: Springer.

Dellen, B., Maurer, F., Münch, J., & Verlage, M. (1997). Enriching software process support by knowledge-based techniques. *International Journal of Software Engineering and Knowledge Engineering*, *7*(2), 185-215.

Doyle, D., Tsymbal, A., & Cunningham, P. (2003). *A review of explanation and explanation in case-based reasoning* (Technical Report TCD-CS-2003-41). Dublin, Ireland: Trinity College Dublin, Department of Computer Science.

Erol, K., Nau, D., & Hendler, J. (1994). HTN planning: Complexity and expressivity. *Proceedings of the Twelfth National Conference on Artificial Intelligence* (pp. 123-1128). Seattle, WA: AAAI Press.

Langley, P., & Laird, J. E. (2002). *Cognitive architectures: Research issues and challenges* (Technical Report). Palo Alto, CA: Institute for the Study of Learning and Expertise.

Lua (2004). Lua scripting language. [http://linux.maruhn.com/sec/lua.html] (last checked: 14 May 2004).

Majchrzak A., & Gasser, L. (1991). On using artificial intelligence to integrate the design of organizational and process change in US manufacturing. *AI and Society*, *5*(**4**), 321-338.

McSherry, D. (2003). Explanation in case-based reasoning: An evidential approach. *Proceedings of the Eighth United Kingdom Workshop on Case-Based Reasoning* (pp. 47-55). Cambridge, UK: Springer.

Muñoz-Avila, H., Aha, D.W., Nau, D., Weber, R., Breslow, L.A., &.Yaman, F. (2001). SiN: Integrating case-based reasoning with task decomposition. *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence* (pp. 999-1004). Seattle, WA: Morgan Kaufmann.

Murdock, J.W. (2001) *Self improvement through self understanding*. Doctoral dissertation, College of Computing, Georgia Institute of Technology, Atlanta, Georgia.

Murdock, J.W., Aha, D.W., & Breslow, L.A. (2003). Assessing elaborated hypotheses: An interpretive case-based reasoning approach. *Proceedings of the Fifth International Conference on Case-Based Reasoning* (pp. 332-346). Trondheim, Norway: Springer.

Petrie, C. (1991). *Planning and replanning with reason maintenance*. Doctoral dissertation, Department of Computer Science, University of Texas, Austin.

Pews, G., & Wess, S. (1993). Combining model-based approaches and case-based reasoning for similarity assessment and case adaptation in diagnostic applications. In M.M. Richter, S. Wess., K.-D. Althoff, & F. Maurer (Eds.) *Proceedings of the First European Workshop on Case-Based Reasoning* (Technical Report SR-93-12). Kaiserslautern, Germany: University of Kaiserslautern, Fachbereich Informatik.

Roth-Berghofer, T.R. (2004). Explanations and Case-Based Reasoning: Foundational Issues. *To appear in the Seventh European Conference on Case-Based Reasoning*. M adrid, Spain: Springer.

Stratagus (2004). A real time strategy engine. [http://stratagus.sourceforge.net/] (last checked: 14.May. 2004).