Case-Based Plan Recognition in Computer Games

Michael Fagan and Pádraig Cunningham

Department of Computer Science
Trinity College Dublin
Ireland
{Michael.Fagan, Padraig.Cunningham}@cs.tcd.ie

Abstract. In this paper we explore the use of case-based plan recognition to predict a player's actions in a computer game. The game we work with is the classic Space Invaders game and we show that case-based plan recognition can produce good prediction accuracy in real-time, working with a fairly simple game representation. Our evaluation suggests that a personalized plan library will produce better prediction accuracy but, for Space Invaders, good accuracy can be produced using a plan library derived from the game play of another player.

1. Introduction

Graphics in computer games have now reached a standard where most users are more than satisfied with the quality on offer, and it is difficult to differentiate a new game title by the graphics alone. Game designers are turning to Artificial Intelligence (AI) to produce a more interesting and more realistic gaming experience. The main way that AI can help game design is by supporting the development of more realistic non-player characters (NPCs). A first objective in this direction is to support adaptive behaviours where the NPCs do not do the same stupid thing all the time. A fundamental requirement for this is for the NPC to have some model of the player's behaviour that will allow the NPC to anticipate and thus adapt to the player's actions. In AI terms, it would be useful for the NPC to be able to perform plan recognition.

In this paper we present a prototype plan recognition system called COMETS that uses the case-based plan recognition idea [4]. The idea is to have the NPC observe the player's behaviour and identify plans or patterns that recur. Then the NPC can identify future executions of these plans and anticipate what the user will do next. In the case-based plan recognition (CBPR) methodology the observed plans are stored in a case-base (plan library) and the player's behaviour is constantly compared to the case-base to identify the onset of the execution of a recognized plan.

For this initial evaluation we use the classic Space Invaders game. Space Invaders (SI) has been chosen because it is a straightforward example of a game where the player executes plans. While COMETS does produce good predictions for Space Invaders there is no scope within the SI game to react to this information so in future work we need to apply these techniques in other games to exploit the full potential.

In the next section we present a brief overview of research on plan recognition before presenting in sections 3 and 4 the representation of plans used in COMETS.

Section 5 presents an evaluation of the prediction power of COMETS that considers the impact of case-base size and the provenance of the cases on prediction accuracy. The paper concludes in section 6 with a discussion of some directions for future work.

2. Plan Recognition

Plan Recognition (PR) is the process whereby an agent observes the actions of another agent with the objective of inferring the agent's future actions, intentions or goals. Several methods for plan recognition have been explored. The most notable are deductive [1], abductive [2], probabilistic [3] and case-based [4]. PR approaches may also be classified according to whether the PR process was *intended* [1] or *keyhole* [5]. If the observed agent cooperates to convey his or her intentions to the recognising agent, as in natural language dialogue systems [6], then the PR process is said to be *intended*. Whereas if the relationship between the observing and the observed agents is non-interactive then it is termed *keyhole* PR [7][8].

In some PR systems the plan library is handcrafted by the system designers and often must be complete (see for example the work by Kautz [1]). Building complete plan libraries is a tractable knowledge acquisition task only when the domain complexity is low. In real-world scenarios this is not often the case and constructing complete plan libraries may be out of the question. Furthermore, it is often the case that extraneous plans are included which bloat the plan library with irrelevant information that impacts on the efficiency of the recognition mechanisms.

In recent years efforts has been made to automate the process of building the plan library using Machine Learning (ML) techniques [9][10][4]. Constructing the plan library in this fashion allows the plan library to be personalised, tailoring it to reflect the idiosyncrasies of an individual's behaviour. This is one of the real attractions of the CBPR idea where the plan library can be built from actual data rather than by hand-crafting cases/plans. Since CBPR is in the lazy learning spirit of CBR the plan library can initially be seeded with generic cases with personalised cases being added as more plans are observed.

3. States and Actions

Plans comprise states and actions that enable state transitions. In systems such as STRIPS [12] or PRODIGY [11] a state of the world is represented by a conjunction of first order predicates. An action may only be executed if its preconditions are matched by the state of the world. This idea is simplified in COMETS so that a state is an atomic concept. There are three possible states that make up the *state set*:

{Safe, Unsafe, VeryUnsafe}

The states are an abstraction of what is going on in the game and the player is defined to be in one of these states depending on what conditions are satisfied in the game at that point. These conditions may be likened to the first order predicates that encode

the state of the world in PRODIGY [11] or in Kerkez and Cox's CBPR system [4]. The three states are shown in Fig. 1. On the left the player is 'safe' behind a bunker; in the centre the player is 'unsafe' in the open but not under fire; on the right he is in the open under fire and so 'very unsafe'. This state-based representation of SI that is used in COMETS is very simple; while it does capture the progress of the game in general terms, some detail is lost.

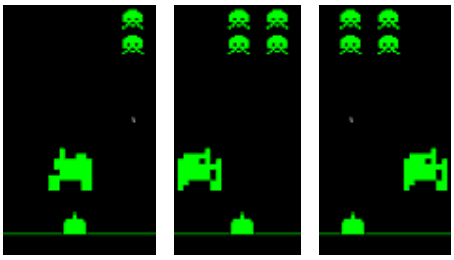

Fig. 1. The three states in the COMETS view of Space Invaders are; *Safe*, *Unsafe* and *VeryUnSafe*.

Like all planning systems, state transitions are brought about in COMETS by *actions*. In STRIPS-like planning systems [13] an action may only be performed if its *preconditions* are satisfied by the player's/planner's state. If the predictions are met then the action can be performed and the action's *effects* are realised. These preconditions and effects define an action. However, since COMETS is *observing* rather than *managing* the planning process there is no need to worry about all this detail. All COMETS has to do is record what happens as a sequence of state-action pairs.

As is the case with the states, there is a predefined set of actions called the *action set*. The action set consists of 5 player actions and one exogenous event. An exogenous event is any event, the system can account for, that occurs in the world that is out of the control of the player. For example, an exogenous event is generated when the player is the target of enemy fire. The action set is as follows and the resulting transitions are shown in Fig. 2:

{fire, hide, emerge, dodge, suicide, exogenous}

So a crucial component of COMETS is the ability to 'watch' the game and abstract what happens into a plan expressed as a sequence of state-action pairs. This abstraction process involves interpreting the raw game data into a more meaningful format. Sensory inputs include the position of the player, the position of the enemies, the presence of a threatening attack etc. The player's current state is calculated using this information. This state is stored in the *state register*. Each time the content of the state register changes an action will have occurred; e.g. a transition from *Safe* to *VeryUnsafe* results from an *emerge* action. In this game representation there is only one action that can produce each state transition with the exception of the transition between *Unsafe* and *VeryUnsafe* that can result from *suicide* and *exogenous* actions. This simplicity makes the recording of the player's actions a light-weight process.

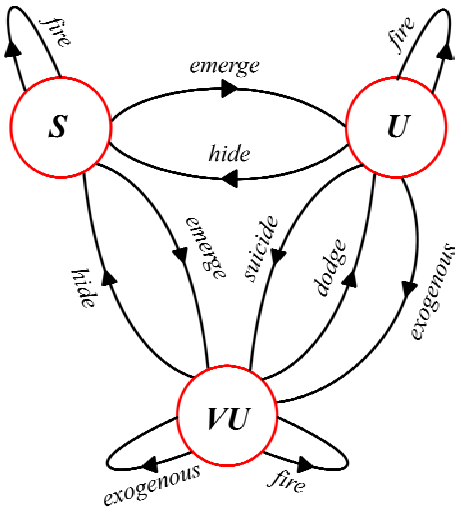

Fig. 2. The state transition diagram for the representation of SI used in COMETS; the arcs represent actions and the nodes represent states.

4. Plans

Plans in COMETS are an ordered sequence of state-action pairs. Plans are assumed to be linear in nature [11][13], i.e. it is assumed that operations may not execute simultaneously. A single SI game is a complete plan. However, the plan recognition process operates on sub-plans that are four steps long (see Fig. 3), the idea being that if game play matches the first three steps in a sub-plan we can guess that the fourth step in this sub-plan will be the player's next action.

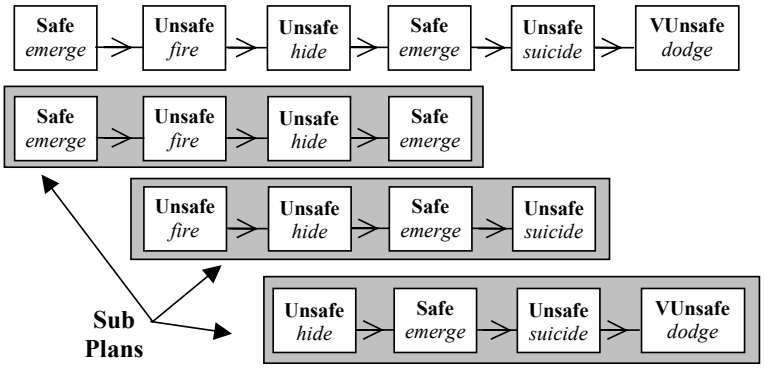

Fig. 3. Sample sub-plans from COMETS.

4.1. Building the Plan Library

One of the benefits of using CBR for plan recognition is that the plan library can be built automatically by observing actual game play. Kerkez and Cox build their plan library dynamically, as the stacking agent is observed executing new plans in the blocks world. In COMETS, the plan library is built after the player has played the game for three sessions. These sessions yield several plans corresponding to the games played in those sessions. A plan of length *n* contains *n-(k –1)* sub-plans of length *k.* If we view the plan segment of length 6 in Fig. 3 as a complete plan it consists of 3 sub-plans of length 4. However, all three of these are not of interest. We are only interested in those that turn up systematically. The length 4 as the best length for sub-plans was arrived at after some examination of recorded game play. 3 is too short, and results in too many false matches. While 5 is too long and would not yield enough matches.

To build the plan library, the recorded passages of game play are scanned for frequently followed sub-plans. Each sub-plan is assigned a *support* value that is a count of the number of occurrences of that sub-plan in the plans. Sub-plans with support values above a threshold (currently 5) are candidates for inclusion in the plan library. The plan library's quota of sub-plan's is filled with the sub-plans with the highest support values.

4.2. Storing and Retrieving Sub-plans

Since the process of plan retrieval happens continuously while the game is being played, it needs to be computationally efficient. As an example, let us consider a scenario where the player executes the plan segment shown in Fig. 3. After the *Safe/emerge* state-action pair is observed the retrieval mechanism must consider as candidates all sub-plans which start with this state-action pair. This set of sub-plans is called the *conflict pool*. When the next state-action (*Unsafe/fire)* is observed all sub-plans that do not match this are deleted from the conflict pool and all sub-plans that start with this are added to the conflict pool. At this stage the conflict pool will contain all sub-plans with first two steps *Safe/emerge - Unsafe/fire* and also sub-plans with first step *Unsafe/fire.*

COMETS continues like this, adding and deleting sub-plans from the conflict pool until a single sub-plan is found that matches three steps of game play. The player is *recognised* to be executing this sub-plan and the 4^{th} action from that sub-plan is predicted to be the player's next action.

To support all this, sub-plans are organised in the plan library using an indexing structure based on an integer encoding of their initial state-action pair. The plan library is in fact a hashtable and sub-plans are stored in bins indexed by a common initial state-action pair. (This is based on the techniques used by Kerkes and Cox [4]).

In summary, COMETS 'recognises' the execution of a sub-plan when it matches three consecutive steps of game play and it is the only candidate to match on three steps. The evaluation in section 5 shows that the policy for managing the plan library has important implications for this uniqueness constraint. We might expect that

expanding the plan library to include poorly supported sub-plans will damage retrieval performance and the evaluation shows this to be the case.

Fig. 4 shows a screen shot of the prediction mechanism in operation. The upper text region on the right shows the sequence of actions as observed by COMETS. Below that, the predictions are shown. Each prediction displayed is the final state-action pair from a sub-plan whose first three steps matched the game play. In this example COMETS is doing very well having made three predictions, all three of which are correct.

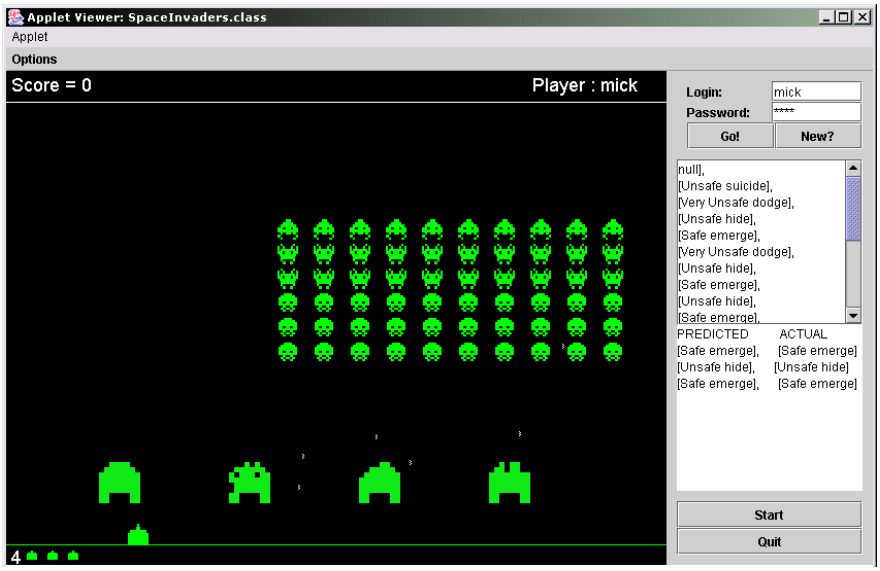

Fig. 4. COMETS observing a Space Invaders Game: The sequence of state-actions is shown on the right with the predictions from COMETS shown below that**.**

5. Evaluation

In this section we look at the accuracy and frequency of predictions coming from COMETS as the size of the plan library varies. We also look at the impact of personalized plan libraries on accuracy. The measure of the system's accuracy is the proportion of successfully predicted actions in the set of predictions.

5.1. Prediction Accuracy v's Plan Size

The CBPR system was trained over three game sessions as described above. A further three game sessions were used to test the accuracy of the resulting plan library. These accuracy tests were repeated for plan libraries varying in size from 10 to 80.

Fig. 5 compares these accuracy figures with the accuracy of randomly generated predictions.

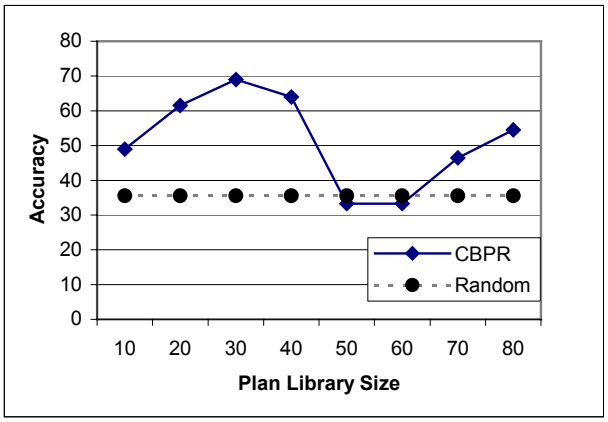

Fig. 5. The accuracy of the CBPR system for different plan library sizes is plotted along side the accuracy of a system performing random prediction.

Fig. 6 shows the change in the prediction rate of the CBPR system as the size of the plan library increases. This is the proportion of user actions for which the system is able to produce predictions. Using a plan library of just 10 sub-plans, the CBPR system will produce predictions in response to 12% of the user's actions. It is important to note that these predictions may be wrong and Fig. 5 shows that a little less than 50% of these are in fact correct. Looking at both graphs, we see that a prediction accuracy of 69% is achievable with a plan library of size 30 with a prediction rate of 7.5%.

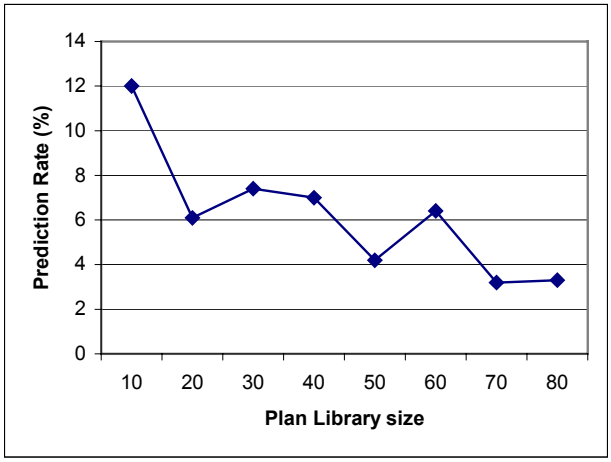

Fig. 6. As the size of the Plan Library grows the frequency of predictions falls.

The prediction rate drops as the size of the plan library increases. This is because, with more sub-plans in the library, the chance of matching on three steps of more than one plan increases; i.e. there may be more than one matching plan in the conflict pool. The current implementation has no means of resolving this conflict so no prediction is made.

The other point to notice in the graphs is the drop-off in accuracy as library size goes from 30 to 60 (Fig. 5). This might be viewed as a form of overfitting where the simpler model represented by the library of 30 cases generalizes better to the player's behaviour. The accuracy rises again above 60 cases but at that stage the prediction rate has fallen to 3%.

This evaluation shows that CBPR can produce predictions with reasonable accuracy in real time. In order to determine the actual 'best' size for the plan library we would need to have some idea of the relative value/cost of good and bad predictions. Clearly, this has no meaning in the context of the SI game as the game does not allow for any adaptive behaviour on the behalf of the NPCs.

5.2. Accuracy for non-Personalized Case-Bases

The other question we consider in the evaluation is that of the origin of the plan library (see Table 1). In the evaluation in section 5.1, the training data and test data were produced by the same player, so the library is personalized to that player. It is important to know if the accuracy depends on this, or if reasonable accuracy can be achieved using a plan library produced from someone else's game play (or using a generic plan library). To assess this, three other plan libraries of size 20 were produced from three other players, two intermediate players and a novice. The prediction accuracy for the target player using the plan library from the novice player fell to 39%. But the intermediate players' libraries kept the accuracy up at 56%. This suggests that a personalized library is best but a generic library can be expected to produce fairly good results. This good result with non-personal cases may depend heavily on the constrained nature of the SI game where good players are inclined to be executing the same plans because of the limited potential for variation in game play.

Table 1. This table shows the accuracy of the predictions for a Target Player using a plan library built from his own play and plan libraries built from the play of others.

Plan Library	Accuracy
Target Player	61.6%
Intermediate A	56.4%
Intermediate B	56.7%
Novice	39%

6. Conclusions and Future Work

Our experience with COMETS suggests that CBPR can be used to predict a player's behaviour in a computer game. Not only can CBPR be the basis for adaptive behaviour in computer games, but it also lends itself well to *personalized* adaptive behaviour.

The performance of the existing system could be improved with a more sophisticated policy for selecting sub-plans for inclusion in the plan library. At present the only criterion is to select the sub-plans with the greatest support in the training data, provided the support is above the threshold of 5. Our evaluation shows that this allows for plans that contradict one another. There is no benefit in having two sub-plans that match on their first three steps so presumably the one with the weaker support should be deleted. There is scope for considerable research on library maintenance questions such as this.

It is also clear that a lot of useful game information is not captured by the simple plan representation in use in COMETS. We have done some work on a more sophisticated representation based on interval temporal logic [2] and the benefit of this more complex representation needs to be evaluated [13].

Before we can take this research much further, we need to move to a game environment that will allow for adaptive behaviour from the NPCs. Questions about library maintenance policies and library size depend on how the predictions are used and on the utility of predictions and the cost of errors. Are frequent less accurate predictions useful? Or are rarer but more accurate predictions better? There are also interesting questions about the use of generic versus personalised plan libraries that can only be answered in a game environment that allows adaptive behaviour.

We must also recognise that the good coverage we get from a case-base of 30 sup-plans may depend on the constrained nature of the Space Invaders game. A more complex game environment may require a much larger case-base to get reasonable coverage. In a more complex environment a generic case-base may also be much less effective and it may be more important to personalise the case-base to the player to get good performance.

References

1. Kautz., H., A Formal Theory of Plan Recognition and its Implementation, *Reasoning About Plans*, Allen, J., Pelavin, R. and Tenenberg, J. ed., Morgan Kaufmann, San Mateo, C.A., 1991, pp. 69-125.
2. Ferguson, G. and Allen, J.F., Events and Actions in the Interval Temporal Logic, *Journal of Logic and Computation,* Special Issue on Actions and Processes, Vol. 4, No. 5, October, 1994, pp. 531-579.
3. Charniak, E. and Goldman, R., A Bayesian Model of Plan Recognition, *Artificial Intelligence Journal*, Vol. 64, pp. 53-79, 1993.
4. Kerkez, B. and Cox, M., Incremental Case-Based Plan Recognition Using State Indices, *Cased-based Reasoning Research and Development: Proceedings of 4th International Conference on Case-Based Reasoning (ICCBR 2001),* Aha, D.W., Watson, I., Yang, Q. eds., pp. 291-305, Springer-Verlag, 2001,

5. Cohen, R., Song, F., Spencer, B. and van Beek, P., Exploiting Temporal and Novel Information from the User in Plan Recognition, *User Modelling and User-Adapted Interaction,* Vol. 1, No. 2, 1981, pp. 125-148.
6. Allen, J. F., and Perrault, C. R., Analyzing Intention in Dialogues, *Artificial Intelligence*, Vol. 15, No. 3, 1980, pp. 143-178.
7. Albrecht, D. W., Zukerman, I., Nicholson, A. and Bud, A., Towards a Bayesian Model for Keyhole Plan Recognition in Large Domains, *Proceedings of the 6th International Conference on User Modelling,* 1997, pp. 365-376.
8. Albrecht, D. W., Zukerman, I., and Nicholson, A., Bayesian Models for Keyhole Plan Recognition in an Adventure Game, *User Modelling and User-Adapted Interaction,* Vol. 8, 1998, pp. 5-47.
9. Lesh, N., Rich, C. and Sidner, C., Using Plan Recognition in Human-Computer Collaboration", *Proceedings of the 7th International Conference on User Modelling,* 1999, pp. 23-32.
10. Bauer, M., Acquisition of User Preferences for Plan Recognition*, Proceedings of the 5th International Conference on User Modelling*, 1998, pp.936-941.
11. Veloso, M., Carbonell, J., Perez, A., Borrajo, D., Fink, E., Blythe, J., "Integrating Planning and Learning: The PRODIGY Architecture, *In Journal of Experimental and Theoretical Artificial Intelligence*, Vol. 7, No. 1, 1995.
12. Fikes, R. and Nilsson, N., STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving, *Readings in Planning*, Allen, James, Hendler, James, Tate, Austin, ed., Morgan Kaufmann, San Mateo, C.A., pp. 88-97, 1990 also in *Artificial Intelligence*, Vol. 2, 1971, pp. 198-208.
13. Fagan, M., *Anticipating the Player's Intentions: A Case-based Plan Recognition Framework for Space Invaders*, M.Sc. thesis, University of Dublin, Trinity College, Dublin, Ireland, 2002.