University of Edinburgh
School of Informatics

Plan Recognition in RISK

4th Year Project Report
Artificial Intelligence and Software Engineering

Jibran Abdus Zaahir Khan

March 20, 2013

Abstract: This paper presents the design and implementation of a plan recognition agent based on an algorithm published by Christopher Geib and Robert Goldman in 2009 [6] called The Probabilistic Hostile Agent Task Tracker (PHATT). The plan recognition agents goal is to attempt to infer the unknown plan of a hostile agent from observations of their behaviour in the RISK environment.

Acknowledgements

Michael for agreeing to supervise project, parents, family for support, C.Geib for advice and taking his time to meet with me, friends, people who helped with results

(Distinguish between academic and non-academic work.)

Contents

1 Introduction **1**
- 1.1 Motivations . . . 1
- 1.2 Artificial Intelligence and Board Games . . . 2
- 1.3 Aims . . . 3
- 1.4 Objectives . . . 3
- 1.5 Hypothesis . . . 3
- 1.6 Paper Structure . . . 4

2 Background **5**
- 2.1 Previous Work in Plan Recognition . . . 5
  - 2.1.1 An Example of Plan Recognition . . . 6
- 2.2 Introduction to RISK . . . 8
  - 2.2.1 Equipment . . . 8
  - 2.2.2 Rules . . . 8
- 2.3 Why Plan Recognition in Board Games . . . 12
- 2.4 Summary . . . 12

3 Design **13**
- 3.1 Introduction to PHATT . . . 13
  - 3.1.1 Computing an Explanation's Probability . . . 14
- 3.2 Environment Modelling . . . 14
  - 3.2.1 Root Goals . . . 14
  - 3.2.2 Sub-Goals . . . 15
  - 3.2.3 Actions . . . 15
  - 3.2.4 Territory . . . 18
  - 3.2.5 Continent . . . 19
  - 3.2.6 Player . . . 19
- 3.3 Explanations . . . 20
  - 3.3.1 Prediction Agent Pseudo Code . . . 21
  - 3.3.2 Example of Operation . . . 27
  - 3.3.3 Conceptual Issues . . . 27
  - 3.3.4 Summary . . . 28

4 Implementation **29**
- 4.1 Modifications to Open Source Project . . . 30
- 4.2 System Construction . . . 30
- 4.3 System Architecture Concepts . . . 31
  - 4.3.1 Google Guava Libraries . . . 31
  - 4.3.2 Replaying Games . . . 31

4.4 Summary 31

5 Evaluation **33**
5.1 Experiments 33
5.1.1 AI Play 33
5.1.2 Constrained Play 33
5.1.3 Free Play 34
5.2 Experimental Format 34
5.3 Data Collection 34
5.3.1 Experimental De-briefing 35
5.4 Experimental Findings 35
5.5 Outcomes 36
5.6 Criticism 36

6 Conclusions **37**
6.1 Future Work 37
6.2 Final 38

7 Appendix **39**
7.1 Class Diagram 39
7.2 Experimental Results 39

Bibliography **41**

1. Introduction

AI has the potential to become the new driving force behind computer game innovation.

John David Funge, Artificial Intelligence for Computer Games, An Introduction

1.1 Motivations

Artificial Intelligence for games or commonly termed 'game A.I.' has been an area of research ever since the beginning of significant work on A.I. Though techniques for game A.I. have typically come from academia and as John Funge argues [5] is notably different in both scope and application.

The primary goal of academia is to further human understanding often by solving particularly complex problems,as a result the scope of developed A.I. techniques from academia are usually more general in their application.

On the other hand game A.I. is built to provide an enjoyable experience for those playing the game, usually by creating the illusion of intelligence. Game A.I. is frequently built with a singular purpose in mind, which is generally considered to be any kind of control problem in a game.

Companies in the games industry who utilize game A.I. (which today is the vast majority) tirelessly seek an edge over their competition. This edge has typically come from computer graphics, effects such as dynamic rendering, the move from two dimensional to three dimensional to keep their customers interest.

Emerging though is the idea that as users become accustomed to high quality graphics, developers will require something new to give their product an edge over their competitors. That edge will hopefully be better quality game A.I.

There has already been some examples of successful games such as F.E.A.R which Goal-Oriented Action Planning which utilized STRIPS style planning techniques[ref], the game gained much acclaim and is referenced as a good example of game A.I. by the industry and others.

With this shift in focus, there grows greater incentives and more opportunities for the continued development of more sophisticated game A.I. likely with techniques developed in academia. The application of plan recognition algorithms may provide such an opportunity.

Plan recognition is the problem of inferring an agents plans from observations. Research in plan recognition began in the . The results of which have been numerous applications to a variety of fields.

In Nate Blaylock paper on "Retroactive Recognition of Interleaved Plans for Natural Language Dialogue" [2] he highlighted a few of the most prominent applications of plan recognition as being:

Field	**Some Applications**
User Modelling	Operating Systems, Intelligent Help Systems, Intelligent Tutoring
Multi Agent Interaction	Military Tactical Defence, Multi Agent Coordination
Natural Language Processing	Story Understanding, Machine Translation, Dialogue Systems

WRITE THIS SECTION!!!!!

Plan recognition has received a great deal of attention due to its ability to provide personalized responses. Applying plan recognition to games are no different.

Plan recognition can perform automated game analysis which can be used by players to improve their performance.

Plan recognition could be potentially be applied to games, leading to responses that are personalized based on for example a players strategy preferences.

Combined with planner systems one could imagine and A.I. capable of recognising a plan and building a counter plan.

1.2 Artificial Intelligence and Board Games

In 1915 Leonardo Torres y Quevedo's built a chess automaton[ref]. It was considered the worlds first computer program and arguably the beginning of Artificial Intelligence and board games.

Thirty five years later Alan Turing published a landmark paper[ref], marking what many consider to be the 'birth of Artificial Intelligence'. Soon after John McCarthy officially coined the term Artificial Intelligence at a conference in Dartmouth College[ref]. He defined it as "the science and engineering of making intelligent machines"[ref].

E.g There existed Christopher Stracheys Game AI for the board game draughts built in 1951[ref]. As AI continued to develop so did the complexity of the A.I's

seen in computer games, first AI to appear which used enemies was arcade game [ref].

Since then notable achievements such as Garry Kasparovs defeat to IBMs chess computer Deep Blue in 1997 [ref] have continued to impress the public.

From history it seems clear that AI and its application in games are intertwined becoming a growing area of interest in both commercial and academic fields.

Games are often considered a good metric for testing the quality of an AI [Ref]..

Some academic work done on RISK in particular. Development of an Intelligent Artificial Player for the Game of Risk[ref]. The author Michael Wolf[ref] claims RISK is generally well known but under recognised by academia.

1.3 Aims

What do we want to achieve?

To design a plan recognition agent for the board game RISK based on the PHATT algorithm developer by Geib and Goldman.

To implement a plan recognition agent for the board game RISK that can make successful predictions.

To further understanding of the complexities of performing plan recognition in the board game RISK.

1.4 Objectives

To show plan recognition algorithms can be used successfully to predict an agent's plan in the board game RISK with an accuracy better than randomly picking a mission card.

To provide insight into the complexities of creating plan recognition models for the RISK environment.

To provide insight into the nature of making plans in RISK.

1.5 Hypothesis

What do you want to answer?

Are plan recognition algorithms beneficial in games? Specifically RISK?

If the player is a winner, then the prediction accuracy will be better.

If the probability of a mission card is the highest among all the other mission cards of an agent, then it will be the correct mission card.

Why because they are an artificially derived benefit which can be used by a player to optimize their behaviour, not solely , but in conjunction with the results of a plan recognition system.

Using more data from risk environment in the computation of the likelihood of explanations, the better the prediction accuracy will be.

1.6 Paper Structure

The structure of the paper is as follows:

The following chapter presents a background to the project where plan recognition and the board game RISK are introduced. Reasons for using plan recognition in board games are then discussed.

Chapter 3 describes the methodology of the project, it is split into two sections. The first is design, in this section PHATT is introduced and the design of the plan recognition agent is detailed. The subsequent section is implementation. Code related issues such as modifications to the open source project and any relevant design concepts are discussed.

Finally an an evaluation plan and conclusion is presented. In these, the experimental findings are presented, discussed and any conclusions derived from the experimental findings are presented.

2. Background

2.1 Previous Work in Plan Recognition

Many consider one of the earliest projects on plan recognition to have been in 1978. Having identified plan recognition as a problem in itself, Schmidt et al [9] conducted experiments to determine whether or not people inferred the plans of other agents. From their results they created a rule based system called BELIEVER which attempted to capture the process of plan recognition.

Three years later Cohen, Perrault and Allen identified two different types of plan recognition *keyhole* and *intended* [4].

They defined each as:

- *Keyhole plan recognition* is the recognition of an agent's plan through unobtrusive observation".
- *Intended plan recognition* is the recognition of the plan of a cooperative agent who wishes to be understood.

In 1986 Kautz and Allen published a paper titled "Generalized Plan Recognition" [8] which set the frame work of many plan recognition projects that followed and formed the basis of plan recognition through logic and reasoning. They defined keyhole plan recognition as involving the identification of a set of top-level goals from possible plans, which could be decomposed into related sub-goals and basic actions, thus creating an event hierarchy also known as a *plan library.*

It was Charniak and Goldman [3] who first argued that plan recognition was largely a problem of reasoning under uncertainty and that any system which did not account for uncertainty would be inadequate. They went on to propose a probabilistic rather than logic based approach to plan recognition using a Bayesian model. Their research continues to be popular in many avenues of research including its application in games.

Albrecht, Zukerman and Nicholson [1] performed research on keyhole plan recognition using dynamic Bayesian networks to represent features of an adventure game. The result of their experiments they claimed "showed promise" for some domains.

More recently Synnaeve and Bessiere [10] published a paper of the design and implementation of a plan recognition agent that through observations of building

construction in the game Starcraft, can predict the types of units a player intends to produce based on what buildings they construct.

2.1.1 An Example of Plan Recognition

We can introduce common concepts, assumptions and the process of plan recognition with an example of an agent attempting to infer the plan of another agent in a non-adversarial environment.

Person A is cooking a meal for Person B, in other words A has a single *root goal* (a state they wish to reach), such as cooked beef hamburgers. This root goal can be decomposed into a number *sub-goals* of such as cook meat patties or *actions* such as make meat patties which can be further decomposed into single actions

A wanting to surprise B, will not tell B what his root goal is. B cannot know what A is thinking but B must then somehow infer what A's root goal is likely to be by *observing* A cook in this way A's behaviour can be modelled as a Hidden Markov Model. Person B then models A's plan as follows.

Person B assumes A is rational and that A has a *plan* to achieve their root goal. Whatever A's root goal is, it can likely be decomposed into *sub-goals* such as prepare burger buns or cook beef patties. These sub-goals can then be broken down into basic *actions* to achieve those sub-goals, such as take beef patties out of packet. An important point to note is that these basic actions are not limited to only being part of a sub-goal.

To simplify the example, we introduce various assumptions:

- A believes that they can cook a meal.
- B can only infer the cooking plan of A by observing what A is cooking.
- B knows everything that A can cook.
- Through observing A cook, B can predict with absolute certainty what A will cook.
- A cannot hide any observations.
- A only wishes to cook one meal.
- A has no preferences of what to cook, in other words given a choice the probability of choosing is equally likely.
- One A does not change cooking plan.

Given these assumptions and B's knowledge of A's cooking abilities we can model the set of all of As plans, following the notation set by Geib and Goldman in their

paper on PHATT.

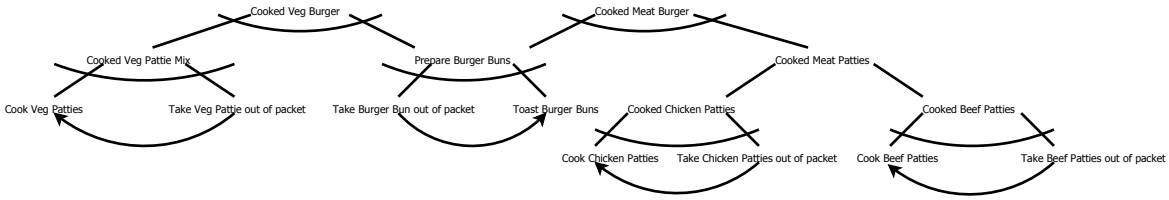

Figure 2.1: Root goals are nodes that have no parent nodes. Sub-goals are nodes that have both a parent and children and actions are nodes that have no children. And-nodes are represented by an undirected arc across the lines connecting the parent node to its children. Or-nodes do not have this arc. Actions or goals that are dependant on a previous action are represented by an arc with an arrow.

A top level or root goal is cooked vegetarian (veg) burgers, it is an and-node and this means that its children represent what needs to be done by A to accomplish that root goal.

Children of or-nodes such as cooked meat patties represent options available to A to accomplish the goal. For example to achieve the goal cook meat patties A could either accomplish cooked chicken patties or cooked beef patties.

Arrows represent dependencies between nodes. For example to be able to accomplish the action cook beef patties, A must first perform the action Take Beef Patties out of packet.

Now A first performs the action *Take Burger Bun out of Packet*, then proceeds to *Toast Burger Buns*. At this point looking at the plan library we have two possible *explanations* for As behaviour, cooked veg burger or cooked meat hamburger.

Each of these explanations are equally likely at this point because the actions that have occured so far could be part of either plan.

A then takes the Chicken Patties of the packet, now given our assumptions we can conclude that A's plan is to cook Meat burgers. In this way B has recognised A's plan thus performing the process of plan recognition.

2.2 Introduction to RISK

Developed and released by french film director Albert Lamorisse in 1957 as La Conquête du Monde, RISK is a turn-based board game for two to six players. There exist many variations but the standard version, which this paper is concerned with, is an adversarial environment where players vie to control a fully observable board portraying a static geographical map of the Earth.

2.2.1 Equipment

Figure 2.2: Risk Equipment

The game consists of three pieces of equipment:

- A board portraying a geographical map of the earth.
- Different coloured tokens called *armies*.
- Two sets of regular six-sided dice.
- A deck of forty four cards.

2.2.2 Rules

The board is divided into forty two *territories*. Each territory is a representation of a real country on Earth. These territories are partitioned into six *continents* usually corresponding to their real continent grouping.

For each territory is a corresponding card, in addition to the name of that territory each card either depicts the symbol of an infantry, cavalry or artillery. By successfully occupying another player's territory in a turn, a player is then award a card and no more than one in that turn. Additionally there are two wild cards which can be substituted to represent a symbol of the players choosing. Owning a set of three cards with the same symbols or a set of three distinct symbols gives the player the opportunity to trade the set of cards for additional armies.

Armies are placed in territories. If a player places an army in a territory, this declares that the player *occupies* that territory. Players must have at least one army placed in a territory they own at all times. Players can choose to place as many additional armies as they wish in that territory.

How a player wins largely depends on the *Game mode*. Game modes significantly impact the behaviour of players and is decided by players beforehand. In standard RISK they are:

Game Mode	**Description**
Domination	Players aim to conquer a certain number of territories,
Mission	Each player is given a single unique mission card. A mission card describes a state they must reach e.g. Occupy North America and Africa, for the rest of the game this is their *root goal* which only they know. In order to win they can either complete this root goal or eliminate all other players.
Capital Risk	Each player is given a Capital territory and to win a player must occupy all other Capital territories.

This paper is concerned with the mission game mode **only**, therefore the following sections describe rules only for that game mode.

After an initial setup, each player's turn is split into three distinct phases which always occur in the same fixed order. These phases are:

1. Reinforcement
2. Attack
3. Movement

In each phase a player performs atleast one discrete *action* which helps to further the players goals, thus forming a sequential task environment.

2.2.2.1 Initial Setup

The game begins with an an initial setup, it involves:

- Territories being divided equally between players.
- Players being given a starting number of armies which is inversely proportionally to the number of players.
- Players distributing their starting armies over their territories.
- Each player being given a mission card.

2.2.2.2 Reinforcement Phase

At the start of a player's turn, they receive *reinforcements* in the form of additional armies. The act of placing an army in a territory is called *reinforcement*. The number of additional armies received is based on the number of territories a player occupies and whether they occupy any continents.

Occupied Continent	Number of Bonus Armies
Asia	7
North America	5
Europe	5
Africa	3
Australia	2
South America	2

Table 2.1: Occupied Continent Army Reinforcement Bonus

2.2.2.3 Attack Phase

Once a player has distributed their armies from the reinforcement phase, they can choose to *attack*.

Territories occupied by a player that contain more than one army can attack neighbouring territories occupied by any other player. An attack consists of several *battles*. The outcome of a battle is decided by means of rolling two sets of dice, thus making it a stochastic environment. One set is rolled for the *defender* of the territory and the other for the *attacker*.

The number of dice each player receives for a battle is dependant on the number of armies placed in each of the players respective territories. The defender receives

a die per army up to two armies. The attacker receives a die per army upto three armies not including the single army they are required to have in that territory while occupying it.

The general rules of engagement are, dice rolls of each player are compared on a one to one basis in descending order of values. The player with the lower value at each comparison loses a single army, if the die are equal the attacker loses an army. The number of comparisons per battle are set by the number of dice the defending player has rolled. The attacking player can commence as many battles as they wish during an attack provided they have more than one army in the attacking territory.

An attack of a territory has three outcomes:

- A *failedAttack* by the attacker as they have only one army remaining in which case they must retreat and a *successfulDefence* by the defender who retains the territory.
- A *failedAttack* by the attacker as they choose to retreat before having only one army remaining and a *successfulDefence* by the defender who retains the territory.
- A *successfulAttack* by the attacker who occupies the territory and a *failed-Defence* by the defender who has no armies remaining and so loses the territory. The attacking player, leaving atleast one army behind, must then move armies from the territory they attacked from into the newly occupied territory.

A player can perform any number of attacks from any territory they own during their turn, provided they have more than one army in the territory they choose to attack from.

2.2.2.4 Movement Phase

When either the player chooses to end the attacking phase or can no longer attack because they do not occupy a territory which contains more than one army their movement phase begins.

During their movement phase a player may move armies from one territory to an neighbouring territory they own, provided they leave atleast one army in the territory the armies were moved from. This action can only be done once per turn in this phase, after which the movement phase is finished.

After the movement phase has been completed the players turn ends and another players reinforcement phase begins.

2.3 Why Plan Recognition in Board Games

Algorithmic landmark - such as the solution of the game of checkers

In the realm of board games such as chess, there have for many years dominated a number of algorithms such as the:

Min-max Algorithm with Alpha-Beta Pruning

Many games today have a large number of alternative moves are stochastic and have hidden state attributes. "Applying traditional game tree search algorithms designed for perfect information games that act on the raw state representation is infeasible" [ref Monte Carlo Planning in RTS Games], this has been said because the search space for such games is large and that finding the best move would take an unreasonable amount of time (Find a ref for this)

Instead of a brute force method of finding the best alternative the idea of plan recognition offers another method. Use plan recognition algorithms one can weed out unlikely explanations before committing to a search of the

Main benefit is that responses can be personalized.

This issue prompted research into the development of variants of the original algorithms that could cope but this

Using these algorithms people like Kabanza are starting to be able to tackle certain aspects which are harder to model?

2.4 Summary

3. Design

3.1 Introduction to PHATT

PHATT was published by Christopher Geib and Robert Goldman in 2009. In their paper they presented the central realization of PHATT's approach being "that plans are executed dynamically and the actions that an agent takes at each step depend critically on the actions that the agent has previously taken".

They introduced a model of plan execution based on the notion of *pending sets* of actions. A pending set is what actions an agent can take based on actions it had already performed, calling it "actions that are pending execution by the agent".

Plan execution according to PHATT was modelled as the following. An agent would first choose a root goal, then a set of plans to achieve that root goal. Any actions of those plans that had no pre-requisite actions would form the initial pending set of the agent. The agent would then perform an action from the initial pending set and this would result in some actions being appended to the pending set and others being removed to form a new pending set. The agent would then continue to perform actions until an outcome such as, the agent concluding the root goal had been achieved.

From this model of plan execution, Geib and Goldman proposed an algorithm utilizing a Bayesian approach to perform probabilistic plan recognition. It computed *Pr(g|obs)*, the conditional probability of a goal g given a set of observations *obs*, by computing *Pr(exp|obs)*, the conditional probability of a particular explanation *exp* of the likelihood of an agent having that root goal, given a set of observations *obs*.

Using Bayes Rule they defined *Pr(exp|obs)* as:

$$Pr(exp|obs) = Pr(exp \wedge obs) \ / \ Pr(obs)$$

They then (as other practical Bayesian systems do) exploited the equivalent formulae

$$Pr(exp_0|obs) = Pr(exp_0 \wedge obs) \ / \ \sum_i Pr(exp_i \wedge obs)$$

This they described as the conditional probability of the explanation exp_0 being computed by dividing the conditional probability of exp_0 by the sum of the probability mass associated with all possible explanations.

3.1.1 Computing an Explanation's Probability

To compute the term $Pr(exp \wedge obs)$ requires the plan library to be augmented with three probabilistic features:

1. The prior probability of the root goal.
2. The respective probabilities of choosing any sub-goals.
3. The probabilities of picking actions from the agents pending set.

The probability of an explanation is then calculated by multiplying each of the terms together in the following manner:

$$Pr(exp \wedge obs) = P(goals)P(plans|goals)P(obs|exp)$$

3.2 Environment Modelling

3.2.1 Root Goals

The root goals of the RISK environment are the mission cards, these are:

- Occupy Europe, Australia and one other continent.
- Occupy Europe, South America and one other continent.
- Occupy North America and Africa.
- Occupy North America and Australia.
- Occupy Asia and South America.
- Occupy Asia and Africa.
- Occupy 24 territories.
- Occupy 18 territories and occupy each with atleast two troops.
- Eliminate a player.

A subset of these where the mission involved occupying continents were chosen to be the focus of this paper. These mission cards are of two types:

1. *Two-Continent*: Players must occupy two explicitly named continents.

2. *Two-Continent+1*: Players must occupy two explicitly named continents and another of their choice.

As mission cards are handed out at random, the prior probability of a root goal is $1/n$ where n is the number of mission cards. The data structure of a root-goal is in the form of a tuple containing the name of the root goal and its prior probability.

$$RG = \{ \text{ rootGoalName, } 1/n\} \ .$$

3.2.2 Sub-Goals

The root goals of the RISK environment can be decomposed to form a set of sub-goals.

- For *Two-Continent* root goals, the sub-goals of that root goal are occupying each continent.
- For *Two-Continent+1* root goals which requires players to allow an option to occupy a continent of choice, provided they also occupy two explicitly named continents.

These two types of root goals therefore make the occupation of any single continent a sub-goal of atleast one root goal. The prior probability of a sub-goal is modelled as $1/c$ where c is the number of alternate sub-goals available in an explanation. The data structure of a sub-goal is a tuple containing the name of the sub-goal and the probability associated with choosing that sub-goal.

$$SG = \{ \text{ subGoalName, } 1/c\} \ .$$

3.2.3 Actions

Excluding card related actions, in RISK the only actions players perform are:

- Attacking a Territory.
- Defending a Territory.
- Occupying a Territory.
- Losing a Territory.
- Reinforcing a Territory.
- Moving armies into a Territory.
- Moving armies out of a Territory.

Each of these action must be modelled in a manner that contributes towards explanations of a player's behaviour.

3.2.3.1 Attacking

Though players can perform several attacks during their turn, they can only attack one territory at a time. Thus players' must choose what territory to attack and, if they wish to attack additional territories, in which order to attack provided they have neighbouring territories and sufficient armies. Modelling these choices provides an avenue to infer a players plan.

The pending set of a player's attack actions for any turn is to either successfully or unsuccessfully attack territories they do not own, that are neighbour to atleast one territory that they own which also contains atleast two armies.

If a player p successfully occupies a territory T_o, then the attack actions of neighbouring territories of T_o are added to p's attack pending set. If they lose a territory T_l due to another players successful attack, then any attack actions of neighbouring territories of T_l are removed from their attack pending set for their next turn.

The attack action is thus modelled as a singular event that is either *consistent* or *inconsistent* with explanations of a players behaviour.

For example, given three explanations:

1. Occupy North America and Australia.
2. Occupy North America and Africa.
3. Occupy Asia and South America.

Successfully occupying a territory in North America would be consistent with explanations 1 and 2, but inconsistent with 3, or in probabilistic terms the likelihood of 1 and 2 would rise whereas 3 would fall.

Having defined the outcome of attacks as either successful or unsuccessful, attacks can be decomposed into following two actions:

Action	Consistent	Reasoning
SuccessfulAttack	Yes	A good indication of a players plan is the territories they attack and successful attacks are in themselves the best outcome.
FailedAttack	Yes	Though less significant than a successful attack, attacking a territory is indictive of a players intention to occupy that territory and therefore a consistent action with an explanation.

Table 3.1: Modelling Attack Actions

3.2.3.2 Defending

Defending as opposed to attacking can be seen as a 'passive' action as an attack is required before a defence can occur. In that way defence can be modelled as only inconsistent with the explanations it is directly related to and, as an act in itself, not related to those it is not.

Practically this is difficult to achieve given the nature of competing explanations, but by making the probability term of representing a defence action large then this effect can be atleast minimised.

For example, given the three explanations from the previous section. If a successful defence were to occur in a territory in North America explanation one and two could be multipled by 1.01. If a failed defence were to occur then explanations one and two would be multiplied by 0.99. This would model a change in the likelihood of both explanations and should have a comparatively minimal effect on the likelihood of explanation 3.

Defence in the same manner as attack can be either successful or unsuccessful, therefore defence can also be decomposed into the following:

Action	Consistent	Reasoning
SuccessfulDefence	Yes	A successful defence of a territory may be purely based on chance, but is more likely when a player has invested armies in the defence of that territory and so observing the action provides a sign that they have an plan involving that territory.
FailedDefence	No	An inconsistent action because a player would not normally allow a territory to be lost if it is a part of their plan.

Table 3.2: Modelling Defence Actions

3.2.3.3 Reinforce

The reinforce actions that a player p can perform at any turn t is based solely on the territories that p owns during turn t. The pending set of any players reinforce actions is therefore modelled as follows.

For each territory T that p owns at a certain turn t. In p's pending set is an action to reinforce T. If a territory T_l is lost by p (it is attacked then occupied by another player), its corresponding reinforce action is removed from the players pending set for the players turn at $t + 1$. Conversely if another territory T_o is occupied by p then a reinforce action for T_o is added to the players pending set at turn t.

3.2.3.4 Movement

The pending set of movement actions of player p is the set of territories a player owns that are neighbour to a territory where p has more than one army. Moving armies into a territory is modelled as a consistent action with the explanation that the territory that had armies moved into is directly related to.

3.2.4 Territory

Each territory is modelled as an entity. When a player occupies a territory the player gains access to a set of actions which together form that territories action set.

The action set of any territory is:

Action	Description
SuccessfulAttack	Attacking and occupying the territory.
FailedAttack	Attacking and failing to occupy the territory.
SuccessfulDefence	Retaining the territory after an attack.
FailedDefence	Losing the territory due to an attack.
Movement	Moving armies in to the territory.
Reinforce	Placing armies in the territory.

Table 3.3: Territory Action Set

The data structure of a territory therefore is a triple containing the name of the territory the set of territory actions TA and a set of references to the territories neighbours TN.

$$T = \{ \text{ territoryName, } TA, TN \ \}$$

A key concept in the design of the agent is that the pending set of a player is decided *a priori* as it is based on the territories a player owns. By combining the action sets of each territory a player owns into a single set, all the actions a player can perform can be captured.

3.2.5 Continent

Each continent contains atleast two territories and so can be modelled as a tuple of the name of the continent and the set of territories that are contained in that continent.

$$C = \{\text{continentName, } <T_1 \ ... \ T_n>\}$$

3.2.6 Player

The term player has been and can be used inter changeably with the term agent. A defined data structure for a player is essential to be able to separate the numerous explanations of one player from another. The data structure must contain alteast three features. A player name or ID, a list of territories they own PT and a list of explanations PE.

$$P = \{\text{playerName, } PT, PE\}$$

3.3 Explanations

Explanations must be designed to contain all the necessary data required to compute its probability, it therefore contains these features:

- The explanation name.
- A root goal RG.
- A set of sub-goals SGS.
- A set of consistent actions ECA.
- A set of inconsistent actions EIA.

The resulting data structure is:

$$E = \{ \text{explanationName, RG, SGS, ECA, EIA} \}$$

Inconsistent actions are stored in explanations due to modelling decisions as if it was the case that that if an action is not consistent then it is inconsistent then defence actions would be modelled incorrectly. Defence actions are not considered inconsistent to explanations that they are not directly related to, but would be if any actions not in the set of consistent actions are classified as inconsistent actions to an explanation.

Explanations of the form Two-Continent+1 have four alternate forms. For example the root goal Occupy Europe, South America and one other continent can be any of the following:

- Occupy Europe, South America and Asia.
- Occupy Europe, South America and Africa.
- Occupy Europe, South America and North America.
- Occupy Europe, South America and Australia.

Each is a valid explanation in itself, but still falls under its parent mission card. Therefore the assumption is made that if the plan recognition agent predicts one of the four children when it is known the players mission card is the parent, it is classified as a correct prediction.

Given the list of root goals and sub-goals, the complete list of explanations is:

- Occupy Europe, Australia and Africa.
- Occupy Europe, Australia and North America.
- Occupy Europe, Australia and South America.

- Occupy Europe, Australia and Asia.
- Occupy Europe, South America and Asia.
- Occupy Europe, South America and Africa.
- Occupy Europe, South America and North America.
- Occupy Europe, South America and Australia.
- Occupy North America and Africa.
- Occupy North America and Australia.
- Occupy Asia and South America.
- Occupy Asia and Africa.

Each of these are considered as possible explanations of a players behaviour. A design modification to PHATT follows this modelling choice. By unpacking (fully enumerating) each explanation, then assigning the full set of explanations to each player from the start of the game, the element of choice has been effectively removed.

What is then measured is the probability of each separate explanation of a players behaviour over the course of the game.

3.3.1 Prediction Agent Pseudo Code

3.3.1.1 Building the Set of Explanations

During the initialisation of the RISK map, a set of explanations must be built for the environment to be later assigned to players. This require the explicit specification of a root goal and sub-goals for each explanation in the environment. In addition it also requires populating each explanation with a set of consistent and inconsistent actions, this can be done by the following operation:

Algorithm 3.3.1: GENERATEEXPLANATIONLIST(−)

```
forall the C ∈ CS do
    forall the E ∈ ES do
        if C isSubGoalOf E then
            forall the T ∈ C do
                addAction(ReinforceT) to ECA
                addAction(MovementT) to ECA
                addAction(SuccessfulDefence) to ECA
                addAction(FailedAttackT) to ECA

                addAction(FailedDefenceT) to EIA
            end
        end
    end
end
```

The above pseudo code loops through the data structure of each continent C in the set of continents CS. Each continent is checked against each explanation E from the set of explanation ES, for whether its name is contained in the set of sub-goals of each E by the *isSubGoalOf* operation. If true then the result is in an If statement firing.

The If statement contains a loop which iterates over the set territories in the current loops continent and using the *addAction* operation, adds each territories action set to the explanations to either the explanations consistent action ECA set or inconsistent action set EIA.

3.3.1.2 Generating Pending Sets

With the set of environment explanations ES, each player is allocated a unique instance of each explanation. Since the computation of the probability of an explanation requires the probability of choosing an action, a function is required which generates a complete set of the available actions a player can perform.

Algorithm 3.3.2: GENERATEPENDINGSET(PT)

```
playerPS = ∅
forall the T ∈ PT do
    addAction(ReinforceT) to playerPS
    addAction(FailedDefenceT) to playerPS
    addAction(SuccessfulDefence) to playerPS
    forall the N ∈ T do
        if playerOwnsN then
            addAction(MovementT) to playerPS
        else
            addAction(FailedAttackT) to playerPS
            addAction(SucessfulAttackT) to playerPS
        end
    end
    return playerPS
end
```

After initializing an empty pending set $playerPS$. The above pseudo code first loops through each territory T in the list of all the territories that a player owns PT. For each territory a $ReinforceT$ action and a $LoseT$ action is added to $playerPS$.

Before proceeding onto the next territory in PT another loop is performed through the set of neighbouring territories TN of that territory with an if-then-else statement. If the player owns that territory a $MovementT$ action is added to $playerPS$, if not then a $OccupyT$ action is added to $playerPS$. After the operation the $playerPS$ is returned and can be cached if necessary.

3.3.1.3 Computing Action Probabilities

Following an idea from a paper by Goldman, Geib and Miller [7] which proposes weighting actions, a method for assigning probabilities to pending set actions generated by the *generatePendingSet* operation was used. It required an operation that would allow greater control over weighting between consistent and inconsistent actions.

Algorithm 3.3.3: ComputeBaseWeight($w, playerTotalActNum, playerConsActNum$)

$sumWeight = w$ * $consActNum$
$leftOver$ = 1.0 - $sumWeight$
$base = leftOver$ / $totalNumAct$
return base

Given a predefined weight w, the total number of actions in a players pending set $playerTotalActNum$ and the consistent actions with an explanation $playerConsActNum$. This operation computes a *base* weight for all actions by subtracting the total weight of the desired actions from 1, then distributing equally the remainder amongst all the actions in the players action set. This operation could easily be replaced with another such similar method.

Given the nature of the environment where the number of actions players can perform are relatively small, this method is suitable provided the weight difference is kept small. For environments where the number of consistent or inconsistent actions a player can perform are large, this may result in the value of *sumWeight* becoming greater than one making *leftOver* negative, breaking the next calculation. Therefore a scalable method of controlling the weights between actions would be necessary in the design of such an agent for a larger action environment.

The operation *ComputeBaseWeight* is part of a larger function which prepares the values which *ComputeBaseWeight* requires.

Algorithm 3.3.4: COMPUTEOBSPROB($playerPS, totalExpConAct, obs$)

$expConsAct$ = *filterPS*(totalExpConsAct, actType)
$playerTotalAct$ = *filterPS*(playerPS, actType)
$playerTotalActNum$ = size of playerTotalAct

$playerConsAct = \emptyset$

forall the $A \in playerAct$ **do**
  if $A \in expConsAct$ **then**
    addAction(A) to $playerConsAct$
  end
end
playerConsActNum = size of playerConsAct

base = *computeBaseWeight*(w, playerTotalActNum, playerConsActNum)

return base

Given the players pending set $playerPS$ generated by the function $generatePendingSet$, and the set of all consistent actions for an explanation $totalExpConAct$. Both the set of actions (must be of the type obs) a player can currently perform $playerTotalAct$ and the consistent actions of the explanation can be be filtered using the $filterPS$ operation which given an action type ad a set, removes all other action types from the given set except for actions of the type specified.

The set of $playerTotalAct$ is then further filtered down into another set $playerConsAct$ which contains only the consistent actions with the explanation. The sizes of $playerConsAct$ and $playerTotalAct$ are saved in two respective variables $playerConsActNum$ and $playerTotalActNum$.

Dividing these two variables effectively gives the proportion of available actions that a player can perform that are inconsistent and this fact is exploited by the next operation which is passed a predetermined weight and each variable to the $computeBaseWeight$ operation which returns the $base$ weight of an action. This base weight can easily be used to weight either inconsistent actions or consistent actions.

This formulae makes three assumptions:

1. That any territories including ones that only contain one army can attack. Though this is not technically true, due to the nature of RISK where players are not restricted in any way on where they may place their armies during

the reinforcement phase a player can practically attack any territory they do not own but have a territory they own neighbour to it at the start of their turn.

2. That the likelihood of consistent attack actions are uniformly distributed.
3. That the likelihood of inconsistent attack actions are uniformly distributed.

3.3.1.4 Computing Explanation Probabilities

Algorithm 3.3.5: COMPUTEEXPLANATIONPROBABILITY(*explanation*, *obs*)

---

if ***not*** *alreadyIni* **then**
  expProb = 1.0
  expProb = *R* * *expProb*
end
playerPS = *generatePendingSet()*

weight = arbitrary number
base = *computeObsProb*(playerPS, totalExpConsAct, obs)

conActProb = *base* + *weight*
inConActProb = *base*
Investment **if** *obs is consistent* **then**
  expProb = *conActProb* * *expProb*
else if *obs is inconsistent* **then**
  expProb = *inConActProb* * *expProb*
end
expProb = *normalized*(expProb)
return *expProb*

After initialising the float *expProb* the explanation is multiplied by the root goal prior R, this is done only when the explanation is initialised and therefore the *alreadyIni* flag is necessary. To complete the computation of an explanation, the operation *computeObsProb* is applied to *obs*. Depending on whether *obs* is consistent or not with the explanation, the appropriate action probability is multiplied with *expProb* .

Normalizing the explanation probability at the end of each operation is vital as this prevents the probability of any explanation from dropping to zero very quickly.

3.3.2 Example of Operation

These concepts from this chapter can be tied together with an arbitrary example.

As mission cards are handed out to players' at random, the prior probability of each root goal is $1/n$ where n is the number of mission cards.

As we know the probability of being a mission card is uniform and though there are six mission cards, two of those have four derivatives resulting in 12 explanations of a players behaviour. Therefore the probability of a single explanation is $1/12$.

Introducing some other assumptions for this example:

The probability of a player choosing to attack any territory is uniform. The probability of a player choosing to occupy a continent as a sub-goal is uniform. The RISK map has every territory connected a with every other territory.

Follow example in slides.

3.3.3 Conceptual Issues

Main change is that PHATT works in an action space whereas I have had to adapt the implementation to represent the state of the 'risk world' for the algorithm to operate.

Another difference is that I do not store or manipulated derivative trees as the original paper includes in the pseudo code it presents, this is due to hard coding derivative trees into explanations.

Another difference is I have removed the need for sub goal probabilities because of working with the full derivative tree.

Normalising after each observation computation rather than only when a probability needs to be computed.

Using the constraints in the environment initially the model was based on the plan library proposed in the PHATT paper. This was not good therefore utilized the constraints of countries to narrow the number of possible explanations and actions.

Initially the movement model was based on the attack model where the smaller the number of actions the bigger the weighting should be, this is an incorrect model as the number of territories a player owns increases as a player gains more consistent movement and reinforce actions as they continue to be successful in their goal.

3.3.3.1 Probability Assignment

Calibrating Mechanisms of algorithm Uniform distribution Simple weighting based on reasoning about implications of an action to an explanation Weighting based on number of consistent and inconsistent actions available to a player when an action is performed.

Manual Simple Weighting Positively weighted probabilities towards consistent actions and negatively towards inconsistent actions

Finding Equal Distribution does not work very well for simple explanations

so made up:

Weighted Distribution based on consistent and inconsistent actions

Though question is How to assign weights to consistent and inconsistent actions?

Total number of actions are counted, and are separated into inconsistent actions and consistent actions and a weight is split among inconsistent actions and consistent actions. Problem is that it seems this system is too sensitive.

Because of sensitivity had to make up a more evenly weighted automatic system.

3.3.4 Summary

The summary for this section is

4. Implementation

Built purely in the Java programming language, the plan recognition agent follows an *event-driven* system architecture, with the following design.

Diagram

(Game -> Processing Class -> Plan Recognition Agent)

Game represents an existing open source project of the board game RISK. Developed by Yura.Net, a team based in London headed by Yura Mamyrin the project is written in the Java programming language[ref - to website].

Event objects contain important data from the environment which the plan recognition agent requires to compute the likelihood of explanations. These events are fired from the *Game* at the following key points:

- Player Initialisation
- Player Removal
- Territory Occupation
- Attacks
- Army Movements

MORE DETAIL OF SPECIFIC IMPLEMENTATION REQUIRED Once fired these events are handled by a *processing* class which (routes?) each event object to the plan recognition agent where it is handled in a manner depending on the event.

In this way the implementation of the plan recognition agent is not as a dependant sub-component of the game but rather a component completely separate entity that operates in *parallel* to the game.

Initially the architecture was designed as a distributed system where the system would store its own subset of data structures from the RISK environment. Treated the game as a black box where only the changes in game state would be observable.

Agent data was generated such as countries they owned. Agent explanations computed by the plan recognition agent based on events from the environment.

The reason for this choice was the simplicity of its implementation at the time and to reduce the dependence on the implementation of the project.

4.1 Modifications to Open Source Project

Modifications to the open source project were minimal, modifications included:

Initialising the plan recognition agent and processing class when the game begins.

The plan recognition agent needs to simply be aware of when the game beings, the changes that occur in the game state and when the game is finished. These *events* are registered by the processing class and sent to the plan recognition agent.

Code that injected events into processing which passed it to the plan recognition agent in various places namely:

- Player Initialisation
- Player Removal
- Territory Occupation
- Attacks
- Army Movements

Also included code that generated csv files when the game was registered as being over.

4.2 System Construction

(Diagram of data structure of local copy)

Adding data to this hard coded data structure proved to be overly time consuming and therefore was changed to an automated system of the generation of a data structure allowing the plan recognition agent to work on any RISK map that could be successfully played in the open source game.

Through the modelling process it is apparent that there is no need for the plan recognition agent to hold its own copy of the game state which is update via registration of events in the original code.

It could utilize the data structures from the game itself rather than storing a copy within the plan recognition agent.

This gives advantages such as is less overhead and eliminating the possibility of the plan recognition agents local copy to lose sync with the actual game state. The disadvantage is that this makes the agent more dependant on the methods of the developers implementations.

Implementation of the plan recognition agent was done completely separately from the game. Piggybacked off the games data structures and just inserted code into the project which registered events occurring in the main game.

4.3 System Architecture Concepts

4.3.1 Google Guava Libraries

Extensive use of the freely available Google Guava libraries to allow the prediction agent to operate concurrently with the game as well as more efficiently in its operations.

4.3.2 Replaying Games

Used an inbuilt tool by the developer of saying "Debug Logs" which could be used to reproduce a game exactly, this allows the algorithm to be tested with different parameters on the exact same game thus ruling out any environmental difference when analysing data.

4.4 Summary

5. Evaluation

Two questions to answer,

1) Does it work?

It performs better than randomly guessing but still at a low probability

2) If not why?

Nature of game, simplicity of model.

Three types of games to test the algorithm in:

5.1 Experiments

5.1.1 AI Play

The open source project has built in AI to play for the mission game mode. This provided an opportunity for the collection of a large number of data samples to test the performance of the plan recognition agent.

5.1.2 Constrained Play

Where players are asked only to perform actions that are consistent with their root goal.

This is defined as given a situation a player only:

Chooses to attack a territory that is directly consistent with their plan or a territory that is on the shortest route to a territory that is directly consistent with their plan.

Reinforces a territory that is directly consistent with their plan or a territory that is on the shortest route to a territory that is directly consistent with their plan.

Moves armies to a territory that is directly consistent with their plan or a territory that is on the shortest route to a territory that is directly consistent with their plan.

Why did I choose to do this?

To establish that the algorithm works given that players behave only a consistent manner with the mission they have been given.

5.1.3 Free Play

Where players are asked to play as normal.

5.2 Experimental Format

Explained the rules then

There are two ways to win in mission RISK, one is to eliminate all the other players, the other is to complete your mission.

Explained to players the rules of the game

5.3 Data Collection

How did I get the data?

After the game is finished I automatically generate a log file that allows me to replay a game exactly and a csv file recording significant features such as:

a players actual mission player explanation probabilities over time which player won and lost

The data collection for AI play could be automated using a program called Autohotkey which allowed the creation of macros which automated the data collection process.

Then used python scripts to analyse the csv files.

Used a tool created by developers to replay games in-order to be able to re-test the algorithm with changes in modelling/calibration.

Using an system built by the original developers was able to easily reproduce a game if necessary for the purposes of evaluating the algorithm under different conditions.

Questions to answer:

What did I tell players?

Explained to players the rules of the game and that there were two primary methods of winning in mission risk either by eliminating their opponents or by completing their mission card player were instructed to "Play to Win"

5.3.1 Experimental De-briefing

At the end of the experiment I asked players to give a short high-level summary of their plan, this included when their plan

asked players to give a summary of what they had done in the game, recorded this down to use for comparison purposes at a later point.

5.4 Experimental Findings

Sampling of probability needs to be more frequent rather than just at end of term as many actions occur between turns and the game may end in the middle of a lot of actions during one turn. Players do a lot of significant things in one turn.

Players who spend a game fighting over a single continent raise the probability of all explanations associated with that continent and the result is a prediction of several explanations being equally likely. This also applies to fighting over two continents that are part of a three continent explanation.

Misclassification by Association - Explanations appear likely because players do things related to them even though they haven't done anything in one of the continents e.g Europe SA, Asia appears likely even though there has been occupation of territories in Europe because of lots of activity in SA and Asia. This issue is slightly negated by how I have modelled the attack actions probability contribution to explanations.

Main questions to answer.

Whether is converges?

It does converge.

How fast it converges?

Converges quite quickly after the attacks begin - use data from average data over time.

Players don't ignore their mission card like I first thought, as long as the mission seems easier to complete than eliminating all other players then players will try achieve their mission.

5.5 Outcomes

In evaluation DO NOT JUST RELY ON MEASUREMENTS! DISCUSS OUTCOMES! MAKE INFERENCES FROM DATA!

THE WHOLE PURPOSE OF ACADEMIA IS TO LEARN, SHOW THAT YOU HAVE SOMETHING TO CONTRIBUTE TO THAT AIM TO LEARN!

Like what did you find hard? What did you do that you found easy? What did you do that you found hard?

Would you recommend doing something, what would you recommend not doing?

TALK ABOUT THE OUTCOMES OF YOUR WORK!

NATURE OF GAME PROBLEM

Inherent issues with PHATT is unable to deal with deception[ref] and often players must perform actions unrelated to a plan to be able to complete their plan e.g survive, have to conquer other non-relevant continents, this confuses the algorithm.

The problem of prediction by association, this is where an explanation which involves a continent that a player has not done anything at all to becoming likely because the explanation contains two other continents that the algorithm has observed many things happen to.

How to decide weighting of

How to deal with this problem?

Using the plan recognition software given a programs a plan to detect how well a program performs a plan.

Using genetic algorithms to optimize this by looking at the numbers returned by the plan recognition algorithm and choosing the best configuration that survived and won.

5.6 Criticism

Essentially attack has been modelled following PHATT but the rest of the actions are simple given a weighting of 0.98 for inconsistent and 1.0 for consistent. Thus attack has been modelled as the most significant action in the environment and this we can intuitively tell is not the case.

6. Conclusions

6.1 Future Work

The application of more sophisticated computation models based on the idea of generating a pending set from the state of the world then performing calculations with that pending set.

Only attack actions have been modelled in a matter

Probability model changed to proportion of consistent actions * 0.1 0.9 - proportion for consistent probability 1 - proportion for inconsistent probability

MODEL AUGMENTATION

CRITICISM - ASSUMPTIONS ARE NOT ALWAYS TRUE BECAUSE OF NON-DETERMINISTIC NATURE OF BATTLES! AND OF MULTIPLE ATTACKS PER TURN FOR PLAYERS

MODEL PROBLEM - MODEL HAS BEEN BUILT NOT TAKING THE EFFECTS OF OTHER PLAYERS INTO ACCOUNT EFFECTIVELY

MODEL PROBLEM - PLAYERS MAY BE IN A SITUATION WHERE THEY CANT PERFORM A CONSISTENT ACTION

MODEL PROBLEM - DOES NOT TAKE EFFECT OF TRADING CARDS INTO ACCOUNT

MODEL PROBLEM - DID NOT MODEL MOVING ARMIES INTO A TERRITORY

MODEL PROBLEM - DID NOT MODEL BASED ON IMPORTANCE OF TERRITORIES E.G DID NOT TAKE INTO ACCOUNT CONTINENT ENTRY TERRITORIES

MODEL PROBLEM - ATTACK, TREATS ANY NON OWNED CONNECTED TERRITORY AS ATTACKABLE, GOES ON ASSUMPTION THAT PLAYER HAS AT LEAST TWO ARMIES IN EVERY TERRITORY

MODEL PROBLEM - Talk about other players using a players own territory as a blocking mechanism to prevent another player attacking.

PROBLEM - DUE TO NORMALIZING OTHER EXPLANATIONS BECOME MORE LIKELY AS A BI-PRODUCT EVEN WHEN TAKING INTO ACCOUNT DEFENCE ACTIONS WHERE NON RELEVANT EXPLANATIONS SHOULD

BE UNAFFECTED, TRIED TO AVOID THIS BY MAKING PROBABILITIES MULTIPLICATIONS MINIMAL.

PROBLEM - MODEL NEGLECTS THE MULTI-ATTACK NATURE OF THE GAME, SUCCESSFUL DEFENCE MODEL REALLY ONLY APPLIES FOR SINGLE ATTACKS PER TURN

MODEL IS TOO LOP SIDED TOWARDS ATTACK, SHOULD TAKE INTO ACCOUNT WHAT TERRITORIES PLAYERS OWN AT THE END OF A TURN.

Augmentation possibility - look at where player has already placed armies and figure out a probability based on what they placing tells you.

CRITICISM, DO NO COMPUTATION OF NEW EXPLANATIONS AS AN INSTANCE OF THEM ARE ASSIGNED AT THE START. MEANING ACCORDING TO PHATT A LIST OF PENDING SETS SHOULD BE SAVED AND A LIST OF ACTIONS AN AGENT HAS TAKEN, BUT I HAVE NOT DONE THAT.

A platform has been setup for further improvements of the model.

IMPROVEMENTS TO MODEL

CAPTURING THE CONNECTIONS BETWEEN CONTINENTS

MAKING A TIERED WEIGHTING SYSTEM FOR THE CURRENT METHOD OF COMPUTING PROBABILITIES

E.G

0.1 WEIGHT FOR MORE THAN HALF THE NUMBER OF CONSISTENT ATTACK ACTIONS IN SET OF ALL ATTACK ACTIONS

0.2 WEIGHT FOR HALF THE NUMBER OF CONSISTENT ATTACK ACTIONS IN SET OF ALL ATTACK ACTIONS

0.3 WEIGHT FOR 1/4 THE NUMBER OF CONSISTENT ATTACK ACTIONS IN SET OF ALL ATTACK ACTIONS

0.4 WEIGHT FOR 1/10 THE NUMBER OF CONSISTENT ATTACK ACTIONS IN SET OF ALL ATTACK ACTIONS

6.2 Final

the benefit of such systems would be to such an extent that it would be considering an unfair advantage if used in events such as official tournaments.

7. Appendix

7.1 Class Diagram

Write up class descriptions with a diagram here

7.2 Experimental Results

Table of correct predictions and incorrect predictions in each game type.

Table of correctness (an percentage of correct predictions by end of game for all data)

Graph of the line of the correct explanation for each agent in a single graph.

GRAPHS to make Probabilities of each explanation over rounds, a graph for average correctness for each game type. How to make this graph meaningful?

Bibliography

[1] David W. Albrecht, Ingrid Zukerman, and Ann E. Nicholson. Bayesian models for keyhole plan recognition in an adventure game. pages 5–47, 1998.

[2] Nate Blaylock. Retroactive recognition of interleaved plans for natural language dialogue, December 11 2001.

[3] Eugene Charniak and Robert P. Goldman. A bayesian model of plan recognition. *Artif. Intell*, 64(1):53–79, 1993.

[4] Philip R. Cohen, C. Raymond Perrault, and James F. Allen. Beyond question answering. In Wendy G. Lehnert and Martin H. Ringle, editors, *Strategies for Natural Language Processing*, pages 245–274. Erlbaum, Hillsdale, 1982.

[5] John David Funge. *Artificial Intelligence for Computer Games: An Introduction*. A K Peters, 2004.

[6] Christopher W. Geib and Robert P. Goldman. A probabilistic plan recognition algorithm based on plan tree grammars. *Artif. Intell.*, 173(11):1101–1132, July 2009.

[7] Robert P. Goldman, Christopher W. Geib, and Christopher A. Miller. A new model of plan recognition. *Artificial Intelligence*, 64:53–79, 1999.

[8] Henry A. Kautz and James F. Allen. Generalized plan recognition. In Tom Kehler, editor, *AAAI*, pages 32–37. Morgan Kaufmann, 1986.

[9] Charles F. Schmidt, N. S. Sridharan, and John L. Goodson. The plan recognition problem: An intersection of psychology and artificial intelligence. *Artif. Intell*, 11(1-2):45–83, 1978.

[10] Gabriel Synnaeve and Pierre Bessière. A bayesian model for plan recognition in RTS games applied to starcraft. In Vadim Bulitko and Mark O. Riedl, editors, *AIIDE*. The AAAI Press, 2011.