University of Edinburgh
School of Informatics

Plan Recognition in R.I.S.K.

4th Year Project Report
Artificial Intelligence and Software Engineering

Jibran A.Z. Khan & Dr. Michael Rovatsos

March 30, 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 agent's goal is to attempt to infer the unknown mission card of an agent from observations of their behaviour in the R.I.S.K. environment. EXTEND TO SUMMARISE CONTENTS OF YOUR REPORT.

Acknowledgements

A thank you to my supervisor Michael Rovatsos for agreeing to supervise this project, his herculean effort and level of involvement with his supervisees is inspirational.

A thank you to my parents and family for their unending support in matters big and small.

A thank you to Christopher Geib for taking his time to meet with me and how he managed to decipher my cryptic blabbering to give his invaluable advice on plan recognition and the intricacies of his algorithm.

Yura.net for their work on Domination, in particular Yura who kindly took her time to explain various aspects of the program.

Friends

All the people who helped with me collect many mountains of data.

Contents

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 Motivation

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, in his book Artificial Intelligence for Computer Games, An Introduction, John Funge argues [5] that academic A.I. and game A.I. are notably different in both scope and application.

Where the primary goal of academia is to further human understanding, often by solving particularly complex problems, the scope of developed A.I. techniques from academia tend to be 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 video 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 have kept their customers interested over the years.

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. and indeed there have already been examples of successful games acclaimed for their game A.I, one such example is F.E.A.R.

A First-Person-Shooter psychological horror video game, F.E.A.R utilizes a S.T.R.I.P.S. style planning-architecture which developer Jeff Orkin termed Goal-Oriented Action Planning [10]. The game gained much acclaim and is commonly referenced as an excellent example of game A.I.

With this shift in focus, there are growing incentives for developers to create more sophisticated game A.I. likely, as in the past, with techniques developed in

academia. Plan recognition may provide such an opportunity for development.

Plan recognition is the problem of inferring an agent's plans from observations. Significant research in plan recognition began in the 1980's, the results of which have had numerous applications in several fields. In Nate Blaylocks paper on "Retroactive Recognition of Interleaved Plans for Natural Language Dialogue" [1] he highlights a few of the most prominent 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

Plan recognition continues to receive attention from various computer science communities due to its ability to both provide personalized responses, and an understanding that the consequences of failing to recognise plans can in some situations be dire; though not so much the latter point, applying plan recognition to video games is no different.

Plan recognition in video games has been used in performing dynamic analysis in game environments such as Real Time Strategy [12] and Multi-User Dungeons[4]. More exciting though is the prospect of research into combining plan recognition algorithms with planning-architectures such as G.O.A.P. The desired result being A.I. capable of recognising plans and subsequently building counter plans.

1.2 Artificial Intelligence and Board Games

NOT COMPLETE

In 1950 Alan Turing published a landmark paper "Computing Machinery and Intelligence" establishing the Turing test, marking what many consider to be the birth of Artificial Intelligence as a field of research. Soon after John McCarthy officially coined the term Artificial Intelligence at a conference in Dartmouth College. He defined it as "the science and engineering of making intelligent machines".

Interestingly though thirty five years earlier Leonardo Torres y Quevedo had built "El Ajedrecista" a chess automaton[ref] capable of playing a king and rook endgame against a king from any position. It was considered the worlds first computer game and arguably the beginning of Artificial Intelligence and board games a relationship it appears older than the term Artificial Intelligence itself.

Since then famous events such as Garry Kasparovs defeat to IBM's chess computer DEEP BLUE in 1997 continues to impress the public.

From history it seems clear that A.I. 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 A.I. [Ref]. There has been some academic work done on RISK. Development of an Intelligent Artificial Player for the Game of Risk[ref]. The author Michael Wolf**[ref]** claims R.I.S.K. is generally well known but under recognised by academia.

Work by undergraduate/graduates at Stanford on A.I. **[ref]** agent to play R.I.S.K. They say R.I.S.K. is an intersection between traditional and modern board games.

1.3 Aims

NOT COMPLETE

- To design and implement a plan recognition agent for the board game R.I.S.K.
- For the plan recognition agent to be able to perform better than a random guess.
- To further understanding in the complexities of performing plan recognition in R.I.S.K.

1.4 Hypothesis

NOT COMPLETE

What do you want to answer?

Are plan recognition algorithms beneficial in games? Specifically RISK?

Do certain explanations perform better? Why?

In what way does overlap of missions effect predictions? Is it negative or positive, or is it both?

Are there any common explanation misclassification occurences and why?

- To determine whether the accuracy of predictions are the same for winners and losers.

- 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.

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

1.5 Paper Structure

The following chapter presents a background to the project where the process of plan recognition and the board game R.I.S.K. 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 major projects on plan recognition to have been in 1978. Having identified plan recognition as a problem in itself, Schmidt et al [11] 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* [3].

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" [9] which set the framework 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 [2] 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 [4] 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 [12] published a paper of the design and implementation of a plan recognition agent in the Real-Time-Strategy game Starcraft, that through observations of building construction can predict the types of

units a player intends to produce.

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 making a meal for Person B. For this meal A can cook only one of two kinds of burger, a meat burger or vegetarian (veg) burger. In other words A has two possible *root-goals* (a state they wish to reach) either *Cooked-Veg-Burger* or *Cooked-Meat-Burger*.

A wanting to surprise B, will not tell B what his root-goal is but B wants to know what to expect for lunch. Since B cannot know what A is thinking, B must somehow infer what A's root-goal is likely to be by *observing* A cook. In this way A's behavioural process is similar to a Hidden Markov Model to B. Person B thus models A's behaviour in the following manner.

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 a number *sub-goals* such as *Cooked-Beef-Patties*. Sub-goals can then often be further decomposed into *actions* to achieve them, such as *Take-Beef-Patties-Out-Of-Packet*. An important point to note is that these actions are not limited to only being a component 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 A's plans, following the notation set by Geib and Goldman in

their paper presenting PHATT.

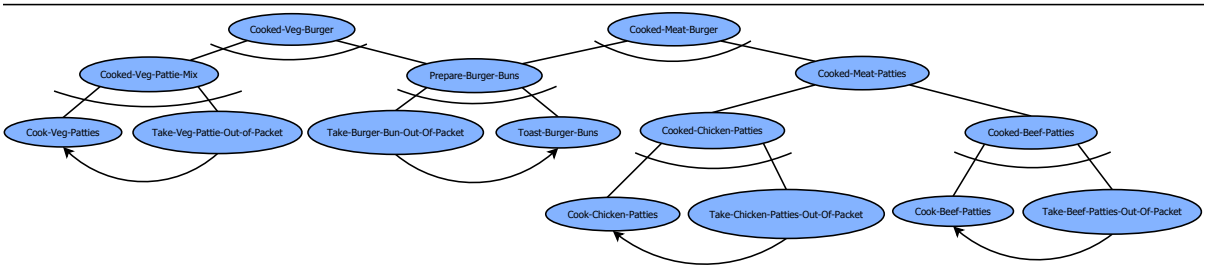

Figure 2.1: Root-goals are nodes that have no parent nodes. Sub-goals are nodes that have both a parent and children and actions are nodes that have no children. And-nodes are represented by an undirected arc across the lines connecting the parent node to its children. Or-nodes do not have this arc.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

To elaborate on the above diagram one of A's top level or root-goals is *Cooked-Veg-Burgers* it is an And-node and so means to accomplish it, requires successfully performing all of its children nodes.

Children of Or-nodes such as *Cooked-Meat-Patties* represent choices available to A to accomplish the root-goal. For example to achieve *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*.

Beginning the example, 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 A's behaviour, *Cooked-Veg-Burger* or *Cooked-Meat-Burger*.

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

A then performs the action *Take-Chicken-Patties-Out-Of-Packet*, given our assumptions we can conclude that A's root-goal is *Cooked-Meat-Burger* and not *Cooked-Veg-Burger*. In this way B has recognised A's plan thus performing the process of plan recognition.

2.2 Introduction to R.I.S.K.

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. This paper is only concerned with the standard version of R.I.S.K. which 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 [8]

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 partition of the land mass of the Earth. These territories are further partitioned into six *continents* usually corresponding to their real continent grouping.

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 at least one discrete *action* which helps to further the players goals, thus forming a sequential task environment.

2.2.3 Turn Structure

2.2.3.1 Initial Setup

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

- Territories equally divided in a random order between players.
- Players being given a number of starting armies. This number is calculated by taking an explicit number defined by the R.I.S.K rule book, which is

inversely proportional to the number of players, and subtracting it from the number of territories the player owns. For example if there are three players, each player receives 35 starting armies and as there are 42 territories each player receives 14 territories. Therefore the number of remaining armies 35 subtract 14.

- Players distributing their starting armies over their territories.
- Each player being given a mission card.

2.2.3.2 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

2.2.3.3 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 by 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 though if the die are equal in value 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 at least 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.

2.2.3.4 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 at least 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.

2.2.4 Cards

For each territory is a corresponding card. As well as 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 awarded 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.

2.3 Why Plan Recognition in Board Games

NOT COMPLETE

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], the search space for such games is so large and that finding the best move would take an unreasonable amount of time (Find a ref for this) Michael Wolf paper calculating state space for RISK is infinite.

Instead of a brute force method of finding the best alternative the idea of plan recognition offers another method. Use plan recognition algorithms we 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

The summary of the chapter is ...

3. Design

NEED MORE MATH EXAMPLES

3.1 Introduction to PHATT

PHATT was published by Christopher Geib and Robert Goldman in 2009 [6]. PHATT's approach is that plans are executed dynamically and the set of actions that an agent can take at each step, their *pending set* depends critically on the actions that the agent has previously taken, this formed the basis of their model of plan execution.

The model is as follows. 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, Geib and Goldman proposed an algorithm utilizing a Bayesian approach to perform probabilistic plan recognition.

The algorithm computed $Pr(g|obs)$, the conditional probability of a root-goal g given a set of observations obs or its equivalent form $Pr(exp|obs)$, the conditional probability of a particular explanation exp that the agent had a 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 is 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 ∧ 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:

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

3.2 Environment Modelling and Design

3.2.1 Root-Goals

The mission cards of the R.I.S.K. environment are the root-goals, 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 at least two troops.
- Eliminate a player.

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 therefore in the form of a tuple containing the name of the root-goal and its prior probability.

$$RG = (rootGoalName, 1/n)$$

The collection of these data structures form the basis of the term $Pr(goals)$ in the computation of $Pr(exp \wedge obs)$.

A subset of these where the root-goal involved occupying continents only were chosen to be the focus of this paper. This choice was made due to time constraints and remains a definite avenue for future work.

Root-goals involving occupying continents can be defined as one 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.

Two-Continent are explicit in their sub-goals, but *Two-Continent+1* type root-goals can be decomposed into *children-root-goals*. Children-root-goals of a *parent-root-goal* are a result of a different choice of sub-goals. The number of children-root-goals of a parent-root-goal depends on the environment. For example the root-goal Occupy Europe, South America and one other continent can be expressed as any of the following children-root-goals:

- 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 root-goal in itself, but are still children of the same parent-root-goal. Therefore an assumption is made, where if the plan recognition agent predicts one of the children-root-goals when the players mission card is the parent-root-goal, it is classified as a correct prediction. This modelling choice has significant implications as a design modification to the PHATT algorithm and thus the plan recognition agent.

As PHATT computes $Pr(exp \wedge obs)$ by multiplying three terms together, choosing to operate on a fully enumerated set of root-goals effectively removes the term $Pr(plans|goals)$, the probability of choosing a sub-goal, from the computation of $Pr(exp \wedge obs)$.

By modelling every root-goal as an explanation of a players behaviour, then assigning that full set of explanations to each player from the start of the game. What is measured over the course of the game is the probability of each explanation of a players behaviour and the highest probability by the end of the game is the plan recognition agents prediction.

3.2.2 Actions

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

- Attacking Territories.
- Defending Territories.
- Reinforcing Territories.
- Moving armies.
- Trading Territory Cards.

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

3.2.2.1 Attacking

Though players can perform several attacks during their turn, they can only attack one territory at a time. Thus players' must choose both which territories to attack and the order in which to attack them, 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 at least 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 player's 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 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	Whether successful or not, attacking a territory is indicative of a players intention to occupy that territory and therefore a consistent action. Though a non-deterministic event, a failedAttack is in part due to a lack of armies which is suggestive that in cases the action has less significance than successfulAttack, as we could assume that players would choose not to attack if their chances of winning were poor.

Table 3.1: Modelling Attack Actions

3.2.2.2 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 the defence action is modelled as consistent or inconsistent with only the explanations it is directly related to.

For example, given the three explanations from the previous section. If a *SuccessfulDefence* were to occur in a territory in North America explanation one and two would be multiplied by a term x which would be greater than 1, whereas explanation three would not be multiplied by anything. In the case of a *FailedDefence* occurring in a territory in North America the same would occur but with x in the range $0.9 < x < 1.0$. This model results in an interesting side-effect.

In PHATT at some point each explanations probability is normalised. The result is that explanations that were not multiplied by x will inversely increase or decrease to explanations that were multiplied x. The more x changes the value of an explanation the greater the difference will be in explanations that were not multiplied by x. To minimize this effect as big changes in explanation probabilities are undesirable for defence actions, the value of x is kept in the range of $0.9 < x < 1.1$.

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

Action	Consistent	Reasoning
SuccessfulDefence	Yes	A successful defence of a territory may be purely a product of chance, but is more likely when they have a 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

3.2.2.3 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.

3.2.2.4 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.

3.2.2.5 Trading Territory Cards

Actions related to trading territory cards were not modelled. This is due to trading sets of cards only giving players bonus armies and the end effect of a player placing any number of armies is already captured by the current model.

3.2.3 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.

$$T = (territoryName, TN)$$

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.

3.2.4 Continent

Each continent contains at least 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.

$$C = (continentName, <T_1 \ ... \ T_n>)$$

3.2.5 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 at least three features. A player name or ID, a list of territories they own PT and a list of explanations PE.

$$P = (playerName, PT, PE)$$

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:

$$E = (explanationName, RG, SGS, ECA, EIA)$$

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.

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.

3.3.1 Prediction Agent Design

3.3.1.1 Building the Set of Explanations

During the initialisation of the R.I.S.K. map, a set of explanations must be built for the environment to be later assigned to players. This first required the explicit specification of a root-goal and of sub-goals for each respective explanation in the environment, or in this context the Domination program. In addition it also required populating of each explanation with a set of consistent and inconsistent actions. This is done by the following operation:

Algorithm 3.3.1: GENERATEEXPLIST(−)

```
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 explanations' 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 an if statement firing.

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

With a complete set of environment explanations ES, each player then allocated a unique instance of every explanation at the start of every game.

3.3.1.2 Computing Action Probabilities

Since the sub-goal probabilities were removed and the root-goal prior probabilities are uniform, finding an appropriate method of computing $Pr(obs|exp)$, the probability of a player choosing an action from their pending set, was critical in the design of the plan recognition agent.

To achieve this of course first presumes that we know what actions a player can perform and for that we require a method of generating a pending set of the available actions a player can perform. This is done by the following operation:

Algorithm 3.3.2: GENERATEPS(PT)

```
playerPS ← ∅
forall the T ∈ PT do
    addAction(ReinforceT) to playerPS
    addAction(FailedDefenceT) to playerPS
    addAction(SuccessfulDefence) to playerPS
    forall the N ∈ TN do
        if playerOwnN 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 then the if condition $playerOwnN$ return true and 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.

With the $generatePS$ operation and an idea from a paper by Goldman, Geib and Miller [7] which proposes weighting action probabilities towards consistent

actions, a technique of doing so in the R.I.S.K. environment was required. Based on reasoning about the implications of an action to an explanation, consistent actions were positively weighted and inconsistent negatively weighted, but by how much is the crucial question?

The first approach to answer this was given a players pending set, the total number of actions are counted, these actions are then separated into consistent and inconsistent actions then a manually defined total weight is split among consistent and inconsistent respectively. For example out of a total action probability of 1.0, 0.5 would be distributed among consistent actions and 0.5 inconsistent actions. The idea behind this approach was that as the number of consistent actions decreased the likelihood of the them choosing the action given an explanation would in theory increase.

Unfortunately this approach proved to be problematic in cases where the number of consistent and inconsistent probabilities were significantly different and even detrimental in cases where the number of inconsistent actions were greater.

For example, if there are:

- 2 Consistent actions and 4 Inconsistent actions, $Pr(cons) = 0.5 \ / \ 2 = 0.25$, $Pr(incons) = 0.5 \ / \ 4 = 0.125$
- 4 Consistent actions and 2 Inconsistent actions. $Pr(cons) = 0.5 \ / \ 4 = 0.125$, $Pr(incons) = 0.5 \ / \ 2 = 0.25$

These significant differences in probabilities between actions resulted in large changes in the probability of an explanation, therefore it required an operation that would allow greater control over the weighting between consistent and inconsistent actions. The following pseudo-code details the operation:

Algorithm 3.3.3: COMPUTEBASEWEIGHT($w, pTotalActNum, pConsActNum$)

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

Given a predefined weight w, the total number of actions in a players pending set $pTotalActNum$ and the total number of consistent actions with an explanation $pConsActNum$. This operation computes a *base* weight for all actions in a pending set 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 computes a *base* value which the following *computeExpProb* operation requires.

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

$expConAct \leftarrow filterPS(totalExpConsAct, actType)$
$pTotalAct \leftarrow filterPS(playerPS, actType)$
$pTotalActNum \leftarrow pTotalAct.size$

$pConsAct \leftarrow \emptyset$

forall the $A \in playerAct$ **do**
  if $A \in expConsAct$ **then**
    $addAction$(A) to $playerConsAct$
  end
end
$pConsActNum \leftarrow pConsAct.size$

$base \leftarrow computeBaseWeight(w, pTotalActNum, pConsActNum)$

return *base*

Given the players pending set $playerPS$ generated by the function $generatePS$, 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 $pTotalAct$ and the consistent actions of the explanation can be be filtered using the $filterPS$ operation which given an action type and a set, removes all other action types from the given set.

The set of $pTotalAct$ is then further filtered down into another set $pConsAct$ which contains only the consistent actions with the explanation. The sizes of $pConsAct$ and $pTotalAct$ are saved in two respective variables $pConsActNum$ and $pTotalActNum$.

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.

3.3.1.3 Computing Explanation Probabilities

Algorithm 3.3.5: COMPUTEEXPPROB($explanation, obs$)

if ***not*** *alreadyIni* **then**
  $expProb \leftarrow 1.0$
  $expProb \leftarrow R * expProb$
  $alreadyIni \leftarrow true$
end
$playerPS \leftarrow generatePS()$
$totalExpConsAct \leftarrow ECA$
$weight \leftarrow arbitraryNumber$

$base \leftarrow computeObsProb(playerPS, totalExpConsAct, obs)$
$conActProb \leftarrow base + weight$
$inConActProb \leftarrow base$

if *obs is consistent* **then**
  $expProb \leftarrow conActProb * expProb$
else if *obs is inconsistent* **then**
  $expProb \leftarrow inConActProb * expProb$
end
$expProb \leftarrow normalized(expProb)$
return *expProb*

After initialising the float *expProb*, the term is multiplied by the root-goal prior R, this is done only when the explanation is first 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* .

At this point in the operation *expProb* is normalized, doing this is a deviation to the design of PHATT. Normalisation in PHATT of an explanations probability is normally only done when sampling of an explanation probability is required, not at the end of every operation. This design choice significantly decelerated the probability of any explanation from dropping to zero very quickly. As if an explanation probability did reach zero, its value would propagate over the data sample and so even if a consistent action were to occur that would normally raise the probability it would be multiplied by zero and so nothing would happen.

3.3.2 Example of Operation

NOT COMPLETE The concepts from this chapter can be tied together with an

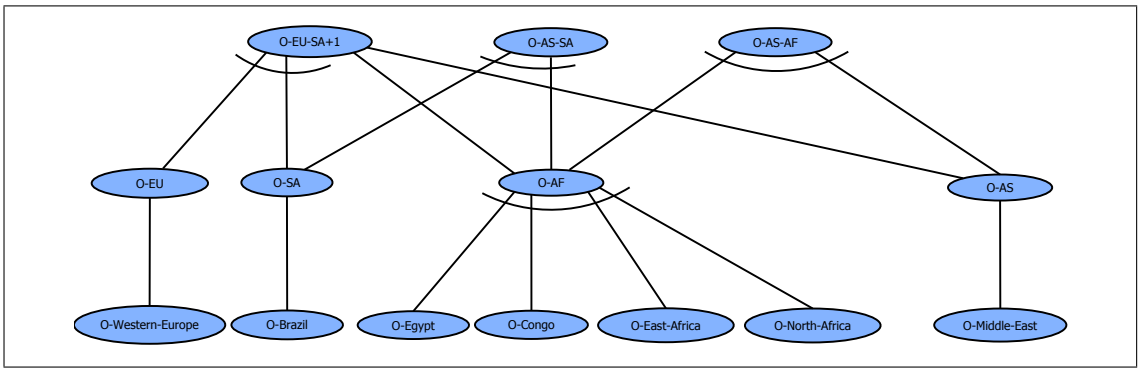

Figure 3.1: Risk Plan Library

arbitrary example.

In this example R.I.S.K map there are two explanations, Occupy-North-America-South-America (O-NA-SA) and Occupy-South-America-Africa (O-SA-A).

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

Some other assumptions for this example:

- The probability of a player choosing a sub-goal is uniform.

Probabilities for each of the explanations = 1 / no of explanations

Player one attacks a territory

3.3.3 Conceptual Issues

NOT COMPLETE

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.

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.

3.4 Summary

NOT COMPLETE

The summary for this section is

4. Implementation

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

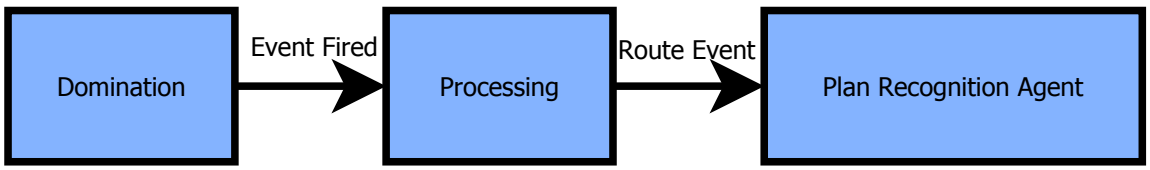

Figure 4.1: Event-Driven Architecture

To operate the plan recognition agent needs to be aware of several important aspects of the game. In particular: how many players there are, when the game begins, the games initial state, any changes that occur in the game and when the game is finished. These aspects can be broken down into individual events:

- Start Game
- Player Initialisation
- Player Removal
- Territory Placement
- Successful Occupation
- Failed Occupation
- Army Movement
- Reinforcement
- End Game

When these occur a distinct event object is fired, these even objects contains important information about the event that is necessary to the plan recognition agent. These events are observed by the processing class which then routes them to the plan recognition agent. The plan recognition agent then looks at the received event object(s) and depending on the type of event object and the information held by it handles the object appropriately. The types of events, what they contain and their purpose are:

The plan recognition agent needs be aware of when the game begins, the games initial state, any changes that occur in the game and when the game is finished.

Event Type	Contains Variables	Purpose
NewPlayer	playerName	Initialises a new player data structure with ID *p* *Name*.
RemovePlayer	playerName	Destroys the player data structure with ID *player*
uccessfulAttack	attackerName, continentName, territoryName	Signals that the player named *attackerName* has formed a SuccessfulAttack.
FailedAttack	attackeName, continentName, territoryName	Signals that the player named *attackerName* has formed a FailedAttack.
ccessfulDefence	defenderName, continentName, territoryName	Signals that the player named *defenderName* has formed a FailedDefence.
FailedDefence	defenderName, continentName, territoryName	Signals that the player named *defenderName* has formed a FailedDefence.
rmyMovement	playerName, continentName, territoryName	Signals that the player named *playerName* has formed an ArmyMovement.
Reinforce	playerName, continentName, territoryName	Signals that the player named *playerName* has formed a Reinforce action.

Table 4.1: Event Object Specification

The architecture is such that the implementation of the plan recognition agent is not as dependant sub-component of the game, but rather a separate entity that operates in parallel to the game which if need be could be replaced by another plan recognition agent.

4.1 Additional Implementation Concepts

4.1.1 Data Structure Re-Use

The plan recognition agent makes use of the data structures from the game itself rather than the alternative of storing its own synchronized copy.

The advantages of this approach is less overhead as well significantly reducing the possibility that the plan recognition agents copy loses sync with the actual game state.

On the other hand the disadvantage is that this makes the agent more dependant on the methods of the developers implementations.

4.1.2 Google Guava Libraries

The plan recognition agent makes 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.

4.2 Summary

NOT COMPLETE

5. Evaluation

5.1 Experiments

5.1.1 A.I. Play

Domination has existing A.I. player implementations for its various game modes. The A.I. players can be substituted for human players to the point where a game can consist of only A.I. players' fighting each other and so can complete a game in a fraction of the time that human players do.

The inner workings of the A.I. player for mission game type according to the developer is that the A.I. does not actually actively follow a plan but that "the mission A.I. is a minimal extension of the core play. Since the core logic is reasonably good at conquering continents and eliminating opponents it just has it's decisions skewed a little to make sure it's going after the right continent/player without causing it to generally play poorly. "

The collection of A.I. play data samples was automated using an free macro program called *AutoHotkey*. By automating the process of initiating A.I. games, it provided an opportunity for the collection of a large number of data samples for evaluation purposes.

5.1.2 Constrained Play

In this form of play, players are asked only to perform actions that are *directly-consistent* with their root-goal. More formally (excluding card related actions) given a set of actions a player will only either:

- Choose 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.
- Reinforce 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.

RETHINK THIS SECTION?

This artificially constructed form of play is designed to be the comparatively easier to perform plan recognition on. By establishing that the algorithm works given that players behave only in a manner that is directly consistent with the mission they have been given, provides grounds to establish that the majority of incorrect classifications arise from players performing actions that are either indirectly-consistent or inconsistent with the players mission.

5.1.3 Free Play

Free play can be described simply as players "playing to win". In this manner it is how players normally play mission RISK.

5.2 Experimental Format

There were two experiment formats, one for constrained play, the other for free play. Before any experiment the rules of the game were explained to participants.

For constricted play, the particular play style required was described to participants.

For free play it was made clear to participants that there were two primary methods of winning in mission risk. Either by eliminating all other participants from the game, or by completing their mission card.

At the end of the game participants were asked to give a short summary of their initial plan and any changes to their plan during the course of the game.

5.3 Data Collection

After a game was finished the program would generate a log file which when loaded through a debug user interface created by the developers, allowed a completely-independent and perfect replay of a full game.

PICTURE OF DEBUG USER INTERFACE

This has significant implications in that the environment can be eliminated as a changing variable in the evaluation process since a game can be reproduced but with the plan recognition agent being implemented differently or with different probability calibrations.

the other a debug user interface created by the developers which allowed the replay of games perfectly. Using this tool meant it was possible to re-observe games and potentially test the plan recongition agent with different models and/or calibrations.

Using tool created by the developer of saving "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.

The plan recognition agent similtaenously would also generate a csv recording the significant features of a game namely:

- Which player won and lost.
- Each players actual mission card.
- The names and probabilities of each players respective explanations over the course of the game.

Python scripts were then used to automate the analysis of the generated csv files, the following section is the findings of that analysis.

5.4 Experimental Findings

5.4.1 Explanation Convergence

Whether is converges?

Find a more complicated graph

TALK ABOUT HOW DOING ACTIONS MAKE SOME EXPLANATIONS MORE LIKELY AND OTHERS LESS LIKELY

THEN AMONGST A GROUP OF LIKELY EXPLANATIONS SOME BECOME MORE LIKELY AND OTHER LESS LIKELY

It will not converge clearly in the case a player performs only actions that are consistent with more than one explanation.

SHOW AN EXAMPLE OF IT NOT CONVERGING

How fast it converges?

At the start of the game fourty-two turns are required before one army is placed on each territory of the R.I.S.K map. After this the remaining initial armies are distributed. During this time the probability of associated explanations will

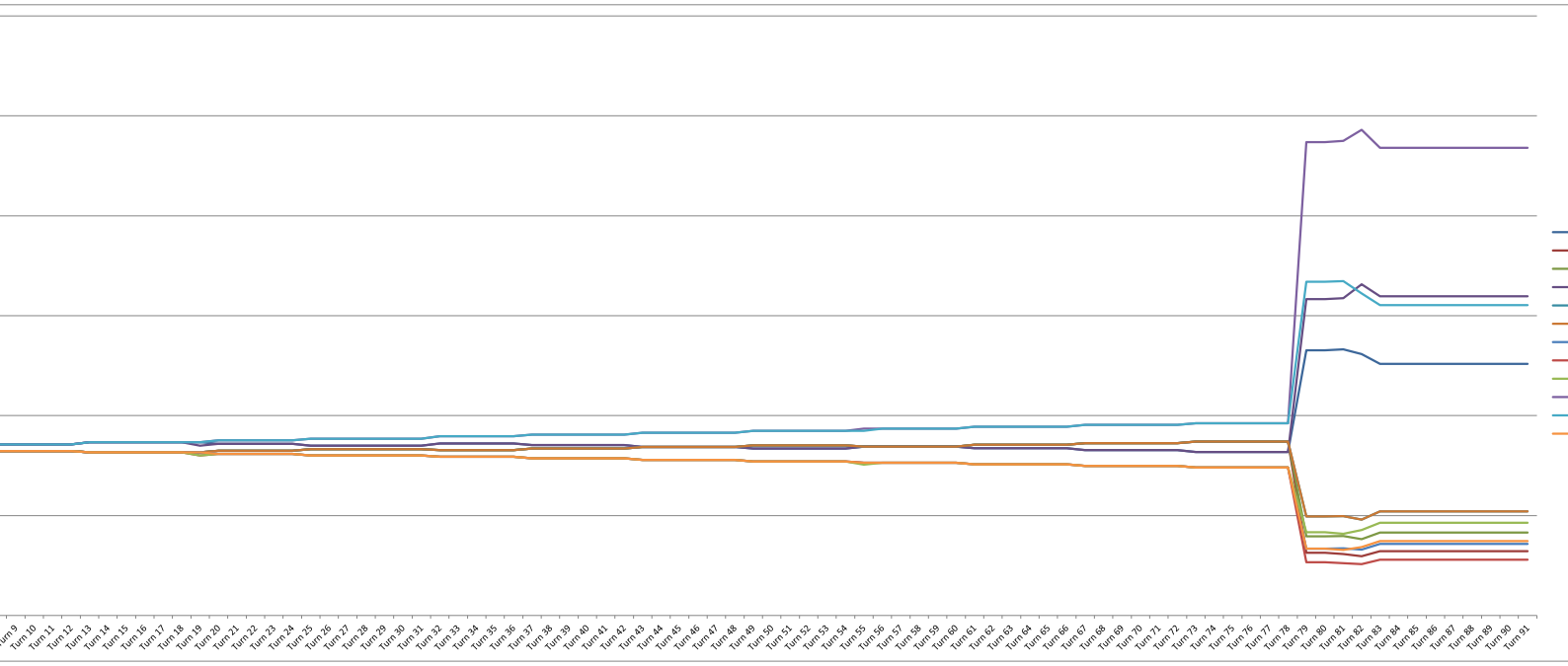

Figure 5.1: Example Convergence

slowly converge upwards or downwards by a small amount depending on whether the reinforcement was consistent or inconsistent.

Once the initial setup is complete, which could be from 80 to 120 turns depending on the number of players, the probabilities move much more rapidly because the prediction agents model has been built to give attack actions the most significance.

5.4.2 Insights

Start with major insights and prove with data

Experiments with the plan recognition revealed the following insights into the data:

1. Constrained play has the best accuracy.
2. The prediction accuracy for winners is better than losers.
3. The difficulty of a mission has an affect on the prediction accuracy of the agent.
4. People perform better than the games A.I.

5. Plan recognition detects that A.I. performs worse.

Do certain explanations perform better? Why?

Why is the accuracy of winners better than losers?

In what way does overlap of missions effect predictions? Is it negative or positive, or is it both?

5.4.2.1 Insight One

From Table 7.1 we see that *constrained play has a significantly higher accuracy than other types of play.* This finding is unsurprising as players only performed directly-consistent actions due to the artificial nature of constrained play.

What is a much more interesting though, is that if this should indeed be the ideal scenerio to detect explanations, then what was the cause for 22.55 % to be incorrectly predicted?

5.4.2.2 Insight Two

If we further break down each of the plays into winners and losers then the picture becomes a little clearer.

From Table 7.2 we notice that in every type of play *the accuracy of winners are better then losers.* This is primarily due to a number of reasons:

Why we get winners right

Winners successfully perform their mission

EXAMPLE

More data for winners, winners also achieve their plan with a greater degree of success meaning more consistent observations BUT winners just like losers may be in positions where they cannot perform actions that are directly-consistent.

EXAMPLE

Winners may occupy a continent in one turn and since sampling is done on a turn by turn basis it may miss a significant change in probabilities as the turn they occupied the continent was the last turn.

EXAMPLE

Why we get winners wrong

Winners may get stuck in situations where they perform inconsistent actions because they are prevented by other players from reaching their goal.

Why we get losers right

Losers

why we get losers wrong

Some reasons for incorrect predictions because of players who lose can be eliminated early, so predictions may be wrong because of a lack of data.

Losers are often prevented from successfully achieving their goal.

Why is the accuracy of free play and A.I. play much lower than constrained play.

Well in free play players there is no requiement for players to be performing directly consistent actons only. Players can be performing actions that are indirectly-consistent which are seen by the plan recognition agent as inconsistent and actions that are actually inconsistent, either reduces the prediction of the correct explanation.

5.4.3 Nature of Game

The nature of R.I.S.K. is adversarial and in-order to achieve ones own mission, players either directly or indirectly eliminate each other and must modify their own plan in order to survive or prevent other players from winning.

Given the model these changes in behaviour are often directly inconsistent with their own mission. Instead this behaviour to the plan recognition agent seems consistent with whatever explanation that those actions *are* directly consistent with instead.

Given this environment and the high-level of overlap between missions the results are misclassification.

For example:

PRESENT EXAMPLE

5.4.4 Simplicity of Model

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.

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.

Calculation problems, give example of a explanation with more territories being higher than one with fewer even though the fewer one was correct.

Misclassification's occur due to three(keep looking in data) main reasons:

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

HOW TO FIX

Solving this problem could be done in potentially two ways either:

Identifying actions are in-directly consistent and modelling them to accurately contribute to explanations.

Giving greater weights to actions that are directly consistent through the state of the game more significant.

For example, knowing that a player owns four out of five of a continents territories when they choose to attack the last territory is intuitively more significant to the explanation of the player wanting to own that continent then if they only owned a single continent of that territory and choose to attack another territory.

Use a system of tiered consistent actions!

ExplanationProb = ExplanationProb * Weight

The weight for each consistent action is flat at the moment or more precisely actions are consistent in themselves but may not be consistent given the *context* of the game state. To deal with this, we should introduce into our calculations data from the game state.

For example

How about making a system of applying weights in a manner that is proportional to the state of the game e.g

Weight = 0.1 * 1- (The proportion of consistent actions for that explanations) * (Proportion of territories a player owns of the continents) <- Proportion of reinforce actions available to the player.

Best case = Low Number of consistent actions and high number of territories owned e.g

Has 1 attack option out of 13 and owns 9 of the ten territories

0.1 * 1-(1/13) * 9/10

Worst Case = High number of consistent actions and low number of territories owned

0.1 * 1-(12/13) * 4/10

Has 12 attack options out of 13 and owns 4 of the ten territories

What is the weight was computed in the normal manner but would be scaled up down by two factors in-order to make an action that appear Plans should consider not only consider what occurs in the action space but the state of the environment as well.

5.5 Outcomes

1) Does it work?

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

2) If not why?

Three types of games to test the algorithm in:

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

Do the accuracy of missions with two continents appear on average to be better?

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!

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!

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.

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

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

6. Conclusions

6.1 Summary of whole report

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

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.

6.2 Main insights and results

6.3 Future Work

Extension of the model to include other mission cards if possible. Due to the design, eliminating players could be modelled but occupy 24, 18 has such high overlap with everything it would be hard to model this.

Eliminating player mission cards - some system where data of who is being consistently attacked is recorded and probabilities are multiplied by a number based on number of attack actions for a player and number of territories that player has remaining.

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.

The model can be augmented further as any action could in fact be further decomposed into two types,*directly-consistent* or *indirectly-inconsistent*. For example a directly-consistent action would be to occupy a territory of an explanation, where as an indirectly-consistent action would be a player breaking another players continent by occupying a territory in that continent, as preventing other players from completing their mission makes completing ones own mission more likely. This distinction has not been taken into account as any action that is not directly-consistent is classified as an inconsistent action in the model and investigation into the effects of doing some may yield improvements.

Only attack actions have been modelled in a manner other than scalin up by 1.02 and down by 0.98 to simulate the effects of small decreases and increases in likelihood. More sophisticated computation models of action probabilities could

be investigated to capture the state of the game to a greater degree in each calculation rather than scaling.

Other than modelling attacks as two events where the result affect both involved players, the model does not considered the effects. This could be improved by

Model does not take into account situations where players cannot perform any consistent actions and therefore any action that a player performs is considered inconsistent but actually could be indirectly-consistent. This could be improved by

The model does not take into account the importance of territories. Each continent has a number of terriories which control entry into the continent. To elaborate each continent has a fixed number of these entry territories and if it were possible for a player to perfectly defend only the entry territories and occupy the remaining terriotries of the continent they would in theory never lose the continent as players must occupy the entry territory to be able to occupy other territories in that continent. The model does not take into consideration this.

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

Using genetic algorithms to perform meta optimisation this is done by randomly assigning a choice weight to the ai then looking at the numbers returned by the plan recognition algorithm and choosing the best configuration that survived and won.

Allow for method choice probabilities again and use machine learning to computer better sub-goal choice probabilities based on preferences shown in training data.

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

MODEL AUGMENTATION

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

Augmentation possibility - look at where player has already placed armies and figure out a probability based on where they place armies. Argument though is placement is not restricted an form.

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. <- MODELLING DIFFERENCE

IMPROVEMENTS TO MODEL

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

7. Appendix

7.1 Accuracy Count Measurements

7.1.1 General Prediction Accuracy

Type of Play	Correct	Incorrect	% Accuracy
Free	29	32	47.54
Constrained	79	23	77.45
A.I. - Weight 100	390	714	35.33
A.I. - Weight 4	89	265	25.14

Table 7.1: General Game Accuracy

Type of Play	Player End State	Correct	Incorrect	% Accuracy
Free	Winner	11	7	61.11
	Loser	18	25	41.86
Constrained	Winner	14	3	82.35
	Loser	65	20	76.47
A.I. - Weight 100	Winner	80	104	43.48
	Loser	310	610	33.70
A.I. - Weight 4	Winner	16	43	27.12
	Loser	73	222	24.75

Table 7.2: General Winner-Loser Accuracy

7.1.2 Free Play

7.1.3 Constrained Play

General Accuracy for each explanation

Mission	Correct	Incorrect	% Accuracy
Occupy-Europe-Australia-One	11	6	64.71
Occupy-Asia-South-America	16	1	94.12
Occupy-Europe-South-America-One	9	8	52.94
Occupy-North-America-Africa	13	4	76.47
Occupy-Asia-Africa	15	2	88.24
Occupy-North-America-Australia	15	2	88.24

Table 7.3: Constrained Play General Explanation Accuracy

Accuracy of two-continent missions appear in general to be better.

General Accuracies for each explanation type separated into winners and losers.

Mission	Correct	Incorrect	% Accuracy
Occupy-Europe-Australia-One	1	1	50.00
Occupy-Asia-South-America	0	0	0.00
Occupy-Europe-South-America-One	4	0	100.00
Occupy-North-America-Africa	5	1	83.33
Occupy-Asia-Africa	0	0	0.00
Occupy-North-America-Australia	4	1	80.00

Table 7.4: Constrained Play Winner Accuracy

The number of data samples are quite small. Why is it that there are no winners for the missions Asia-South-America and Asia-Africa. Is this because these mission cards are harder?

Would make sense seeing as there is a trend in other data showing lower percentage accuracy for that mission.

Mission	Correct	Incorrect	% Accuracy
Occupy-Europe-Australia-One	10	5	66.67
Occupy-Asia-South-America	16	1	94.12
Occupy-Europe-South-America-One	5	8	38.46
Occupy-North-America-Africa	8	3	72.73
Occupy-Asia-Africa	15	2	88.24
Occupy-North-America-Australia	11	1	91.67

Table 7.5: Constrained Play Loser Accuracy

7.1.4 A.I. Play - Weight 100

General Accuracy for each explanation

Mission	Correct	Incorrect	% Accuracy
Occupy-Europe-Australia-One	51	133	27.72
Occupy-Asia-South-America	50	134	27.17
Occupy-Europe-South-America-One	127	57	69.02
Occupy-North-America-Africa	58	126	31.52
Occupy-Asia-Africa	37	147	20.11
Occupy-North-America-Australia	67	117	36.41

Table 7.6: A.I. Play General Explanation Accuracy - Weight 100

Why is it that Europe-South-America-One is better than the rest when trend is normally two-continent accuracy is better?

General Accuracies for each explanation type separated into winners and losers.

Mission	Correct	Incorrect	% Accuracy
Occupy-Europe-Australia-One	15	21	41.67
Occupy-Asia-South-America	9	9	50.00
Occupy-Europe-South-America-One	9	8	52.94
Occupy-North-America-Africa	15	21	41.67
Occupy-Asia-Africa	14	28	33.33
Occupy-North-America-Australia	18	17	51.43

Table 7.7: A.I. Play 100 Winner Accuracy

Mission	Correct	Incorrect	% Accuracy
Occupy-Europe-Australia-One	36	112	24.32
Occupy-Asia-South-America	41	125	24.70
Occupy-Europe-South-America-One	118	49	70.66
Occupy-North-America-Africa	43	105	29.05
Occupy-Asia-Africa	23	119	16.20
Occupy-North-America-Australia	49	100	32.89

Table 7.8: A.I. Play Weight 100 Loser Accuracy

Why is it that Europe-South-America-One is better than the rest when trend is normally two-continent accuracy is better?

7.1.5 A.I. Play - Weight 4

General Accuracy for each explanation

Mission	Correct	Incorrect	% Accuracy
Occupy-Europe-Australia-One	34	133	20.36
Occupy-Asia-South-America	27	140	16.17
Occupy-Europe-South-America-One	88	79	52.69
Occupy-North-America-Africa	28	139	16.77
Occupy-Asia-Africa	9	158	5.39
Occupy-North-America-Australia	49	118	29.34

Table 7.9: AI Play General Explanation Accuracy - Weight 4

General Accuracies for each explanation type separated into winners and losers.

Mission	Correct	Incorrect	% Accuracy
Occupy-Europe-Australia-One	6	27	18.18
Occupy-Asia-South-America	8	21	27.59
Occupy-Europe-South-America-One	14	20	41.18
Occupy-North-America-Africa	3	13	18.75
Occupy-Asia-Africa	1	28	3.45
Occupy-North-America-Australia	5	21	19.23

Table 7.10: AI Play Winner Explanation Accuracy - Weight 4

Mission	Correct	Incorrect	% Accuracy
Occupy-Europe-Australia-One	28	106	20.90
Occupy-Asia-South-America	19	119	13.77
Occupy-Europe-South-America-One	74	59	55.64
Occupy-North-America-Africa	25	126	16.56
Occupy-Asia-Africa	8	130	5.80
Occupy-North-America-Australia	44	97	31.21

Table 7.11: AI Play Winner Explanation Accuracy - Weight 4

7.1.6 Graphs

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] Nate Blaylock. Retroactive Recognition of Interleaved Plans for Natural Language Dialogue, December 11 2001.

[2] Eugene Charniak and Robert P. Goldman. A Bayesian Model of Plan Recognition. 64(1):53–79, 1993.

[3] 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.

[4] David W. Albrecht, Ingrid Zukerman and Ann E. Nicholson. Bayesian Models for Keyhole Plan Recognition in an Adventure Game. *User Modeling and User-Adapted Interaction*, 8(1-2):5–47, January 1998.

[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] Image taken from. *http : //www.insurgencygaming.com*.

[9] Henry A. Kautz and James F. Allen. Generalized Plan Recognition. In Tom Kehler, editor, *Proceedings of 1986 Conference of the American Association for Artificial Intelligence*, pages 32–37. Morgan Kaufmann, 1986.

[10] Jeff Orkin. Symbolic Representation of Game World State: Toward Real-Time Planning in Games. In Dan Fu and Jeff Orkin, editors, *Proc. of the 2004 AAAI Workshop*, Menlo Park, CA, 2004. AAAI, AAAI Press.

[11] 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.

[12] 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, *Proceedings of the Seventh AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, AIIDE 2011, October 10-14, 2011, Stanford, California, USA*. The AAAI Press, 2011.