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

April 3, 2013

Abstract: This paper we present the design and implementation of a plan recognition agent based on an algorithm published by Christopher Geib and Robert Goldman in 2009 [7] called The Probabilistic Hostile Agent Task Tracker (PHATT). The plan recognition agent behaves as an observer to the game and has the goal to infer the unknown mission card of an agent based on observations of their behaviour in the board game R.I.S.K. EXTEND TO SUMMARISE REST OF REPORT

I dedicate this dissertation to my mother and father. Without you both; nothing; not a single thing would be possible.

Acknowledgements

I would like to give a big thanks to Dr. Michael Rovatsos for agreeing to supervise this project. Dor his herculean efforts in deciphering both my babblings at our weekly meetings and the writings of my cryptic drafts. His involvement has been above and beyond the call of duty. Efcharisto poli!

Thank you to my two little sisters and my family for their unending support in matters both big and small, its always appreciated.

A huge thank you to Christopher Geib for taking his time to meet with me, and his attempts to guide a lost undergraduate in the diverse world of plan recognition. His advice in general and his explanations of the PHATT algorithm were invaluable.

Yura Mamyrin and her team at Yura.Net for their work on Domination. In particular Yura who kindly took her time replying to my endless emails.

I want to give a warm thanks you to all my friends and in no particular order. Nick La Rooy for his great advice in troubled times and as a great source of discussion. Punit Bhudia for his encouragement and in helping me collect the data that fuelled this project. To my friends Akash Pandey and Ammar Khan who been supportive through these four long years.

Last but not least a thanks to all my other friends and acquaintances who participated in my many experiments.

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 [6] 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 likely 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 [14]. 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 [16] and Multi-User Dungeons[5]. 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

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, it was thirty five years earlier that Leonardo Torres y Quevedo had built "El Ajedrecista" a chess automaton which was capable of playing a king and rook endgame against a king from any position. Considered the worlds first computer game, it was arguably the beginning of Artificial Intelligence and board games a relationship older than the term Artificial Intelligence itself.

Since then the relationship has only flourished, many games such as chess, checkers and backgammon have all been the subjects of research. Each area has seen its various triumphs such as, Chess Grand Master Garry Kasparovs defeat to IBM's chess computer DEEP BLUE in 1997 and a solution to the game checkers by Jonathan Schaeffer et al in 2007.

It is apparent that the development of A.I. and its application in any medium of games has been and continues to be a significant area of interest for both commercial and academic fields. Games are considered a good metric for testing the quality of an A.I. due to their potentially challenging game environments and opponents.

There has been academic work done on A.I in R.I.S.K. , in particular various designs of A.I players for R.I.S.K. Michael Wolf at the University of Darmstadt wrote his PhD thesis titled "Development of an Intelligent Artificial Player for the Game of Risk" [17], where he claims that R.I.S.K. is generally well known, but under recognised by academia. He then goes on to prove that the state space for R.I.S.K is in fact infinite. Other research on R.I.S.K includes that of efforts by students at Stanford University on an A.I. [10] agent to play R.I.S.K. In their paper they write that R.I.S.K. is an interesting opportunity for research as it is an "intersection between traditional and modern board games".

1.3 Aims

- To design and implement a plan recognition agent for the board game R.I.S.K. using the PHATT algorithm.
- For the plan recognition agent to be able to perform atleast with twice the accuracy of a random guess of mission card.
- To further understanding in the complexities of performing plan recognition in R.I.S.K.

1.4 Hypothesis

- To determine whether plan recognition algorithms are beneficial in board games, specifically in R.I.S.K.
- If the probability of a mission card is predicted the highest among all the other mission cards for an agent, then it will be the correct mission card.

- To determine whether the length of a game has any bearing on the prediction of the plan recognition agent.

1.5 Paper Structure

Chapter 2 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 introduces the design of the system, beginning with the introduction to the PHATT algorithm then the design of the plan recognition agent is and finally an example of the operation of the plan recognition agent.

Chapter 4 details the implementation of the plan recognition agent. We present a high level overview of the design of the system as well as modifications to the open source project and relevant design concepts.

Chapter 5 presents an evaluation of the plan recognition agent, experiments using the agent are described and experimental findings are presented.

Finally chapter 6 the conclusions of the project are presented. The main insights and outcomes of the project are discussed, future work is described.

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 [15] 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" [11] 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 [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.

For example in 1998, Albrecht, Zukerman and Nicholson [5] did research on using keyhole plan recognition using dynamic Bayesian networks to represent features of an adventure game. The result of which "showed promise" for some domains. Five years later Fagan and Cunningham utilized case-based plan recognition in the classic game Space Invaders [12] producing "good prediction accuracies in real time".

More recently though Synnaeve and Bessiere [16] published a paper of the design and implementation of a plan recognition agent in the Real-Time-Strategy game

Starcraft. Their plan recognition agent through observations of building construction patterns could predict the types of units a player intended 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.

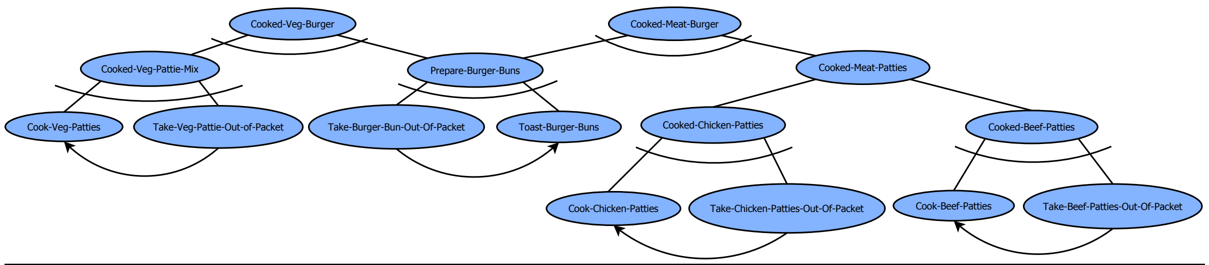

Figure 2.1: Root-goals are nodes that have no parent nodes. Sub-goals are nodes that have both a parent and children and actions are nodes that have no children. And-nodes are represented by an undirected arc across the lines connecting the parent node to its children. Or-nodes do not have this arc.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

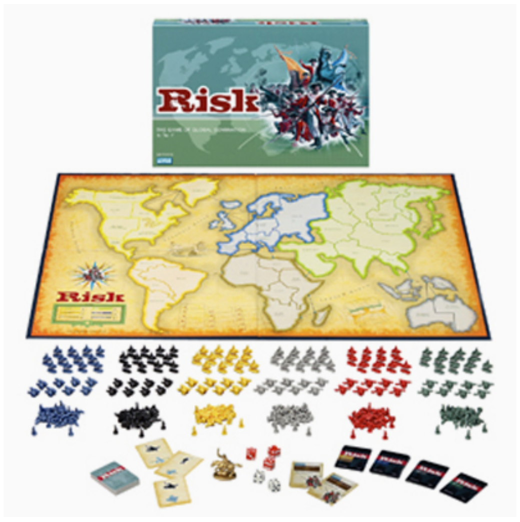

Figure 2.2: Risk Equipment [9]

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 initial setup, which involves:

- Territories being 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

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 *Failed-Occupation* by the attacker as they have only one army remaining in which case they must retreat and a *Successful-Defence* by the defender who retains the territory.
- A *Failed-Occupation* by the attacker as they choose to retreat before having only one army remaining and a *Successful-Defence* by the defender who retains the territory.
- A *Successful-Occupation* 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 a 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

By "knowing a users plan and goals can significantly improve the effectiveness of an interactive system" [2], to this board games are most definitely not exempt. With the success seen in the application of plan recognition to video games, often in complex environments, gives rise to optimism to transferring this success to board games.

Molineaux, Aha and Sukthankar believe that "plan recognition methods can be a powerful ally for machine learning techniques" [13]. Synnaeve and Bessiere have also echo this belief with their work in StartCraft [16]., since their plan recognition agent utilized unsupervised machine learning of common plans which they then could feed into an adaptive A.I. As of today, many program implementations of board games often contain an A.I which is used often as substitutes for human players.

It is an attractive proposition to consider that through the same methods employed particularly by Synnaeve and Bessiere, plan recognition agents for board games could developed to augment board game A.I's, with for example personalized plan library's as Fagan and Cunningham suggest [12] and by doing so build more interesting opponents for people to play against.

2.4 Summary

Plan recognition is an old area of research dating back to the 1950's. It has seen applications in a variety of fields including video games. R.I.S.K. a popular board game provides an interesting environment to test the application of plan recognition algorithms, having features such as being a multi-agent adversarial environment it intersects between classical board games which are often full observable deterministic environments, with elements of modern games such as stochastic environments. Since plan recognition has been used before to augment A.I. for video games, there rises an opportunity to apply them to board games with techniques such as machine learning in-order to create more interesting A.I. opponents.

3. Design

3.1 Introduction to PHATT

MENTION WHY I CHOOSE PHATT!

PHATT was published by Christopher Geib and Robert Goldman in 2009 [7]. 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. Assuming *blind commitment* on the part of the agent, the agent would then continue to perform actions until it concluded 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:

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

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 $\wedge$ 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:

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

3.2 Environment Modelling and Data-Structure Design

PHATT is designed to operate in an action space and therefore it was a vitally important conceptual issue for the design of plan recognition agent to be able to translate R.I.S.K into such a format as to be able represent the action space of a R.I.S.K game at any time. This process involved modelling and data structure design to exploit the constraints placed by the environment on what a player can do and by doing so, infer the plan of a player by based on what they do, given the choices they had.

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 ∧ 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. A modelling difference followed this assumption.

By hard coding every root-goal as explanations of a players behaviour; then assigning that full set of explanations to each player from the start of the game, provided that action probabilities are computed, multiplied with each respective explanation, normalised and then stored immediately as soon as the action occurs. Removes the need to store a history of pending sets, to save a list of actions a player has taken or to create and calculate new instances of explanations. This method is similar to that of the 'Test and Generate' algorithm proposed in PHATTs paper rather than the dynamic algorithm proposed at the end of the paper.

Additionally this has another implication. Since PHATT computes *Pr(exp* $\wedge$ *obs)* by multiplying three terms together. Choosing to operate on a fully enumerated set of root-goals removes the need to compute player choice as we consider every case from the start hence the term $Pr(plans|goals)$ is equal to one for all explanations.

What is then 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

In R.I.S.K. the only actions players perform are:

- Attacking Territories.
- Defending Territories.
- Occupying a Territory.
- 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**
Successful-Occupation	Yes	A good indication of a players plan is the territories they attack and successful attacks are in themselves the best outcome.
Failed-Occupation	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 Failed-Occupation is in part due to a lack of armies which is suggestive that in cases the action has less significance than Successful-Occupation, 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 *Successful-Defence* 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 *Failed-Defence* 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
Successful-Defence	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.
Failed-Defence	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 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.

Movement actions will be modelled as as a less significant indicator of a players plan due to a much higher proportion of reinforce and attack actions which are already modelled in a manner to indicate a players plan.

3.2.2.4 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.

Since reinforce actions are a unrestrained choice in that players are not being acted upon as in defence or constrained by the territories they own as in movement. Reinforce actions will be considered as consistent or inconsistent with all missions when one is observed.

A reinforce action in R.I.S.K is much more frequent than any other action. Though it is an indicator of a plan ultimately where a player attacks is the crucial decision and therefore will be modelled as a less significant indicator of inferring a players plan.

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**
Successful-Occupation	Attacking and occupying the territory.
Failed-Occupation	Attacking and failing to occupy the territory.
Successful-Defence	Retaining the territory after an attack.
Failed-Defence	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.

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

3.2.6 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 Prediction Agent Design

3.3.0.1 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 $C \in CS$ do
    forall the $E \in ES$ do
        if $C$ $isSubGoalOf$ $E$ then
            forall the $T \in C$ do
                $addAction$(ReinforceT) to $ECA$
                $addAction$(MovementT) to $ECA$
                $addAction$(Successful-Defence) to $ECA$
                $addAction$Successful-OccupationT to $ECA$
                $addAction$(Failed-OccupationT) to $ECA$

                $addAction$(Failed-DefenceT) 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.

3.3.0.2 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(Failed-DefenceT) to playerPS
    addAction(Successful-Defence) to playerPS
    forall the N ∈ TN do
        if playerOwnN then
            addAction(MovementT) to playerPS
        else
            addAction(Failed-OccupationT) to playerPS
            addAction(Successful-OccupationT) 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 [8] 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 when multiplied by with 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 $A \in pTotalAct$ do
    if $A \in expConsAct$ then
        $addAction$(A) to $pConsAct$
    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.

3.3.0.3 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
$playerPS \leftarrow generatePS()$
$totalExpConsAct \leftarrow ECA\ totalExpInconsAct \leftarrow filterPS(EIA, obs)$
$weight \leftarrow arbitraryNumber$

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

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.

3.3.1 Example of Operation

CHECK USE OF TERM EXPLANATION AND ROOT GOAL

The concepts from this chapter can be tied together with an arbitrary example of computing *Pr(exp|obs)* the likelihood of an explanation given an observation. The example will be extended to include the plan recognition agent handle two observations.

Assumptions for this example are:

- Mission cards are unique between players and randomly handed out.
- Consistent attacks are assigned a weight w of 0.02.
- When player p_1 attacks a territory it is always a Successful-Occupation action.
- When player p_1 defends a territory it is always a Successful-Defence action.
- A closed world assumption - The R.I.S.K map consists of only territories, continents and root-goals illustrated on Figure 3.1.

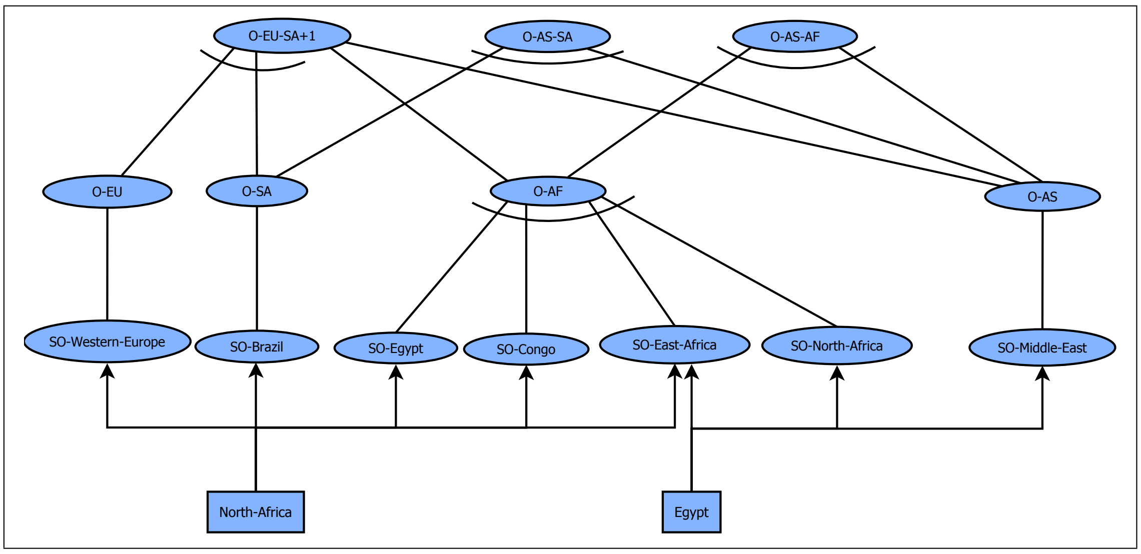

Figure 3.1: Successful-Occupation Plan Library for North-Africa & Egypt, **O** = Occupy, **SO** = Successful-Occupation

The three root-goals of this environment are:

- Occupy-Europe-South-America + One other continent (Occupy-EU-SA+1)
- Occupy-North-America-South-America (O-AS-SA)
- Occupy-South-America-Africa (O-SA-AF)

The first can be expressed in two ways given our example, therefore we must consider four root-goals. Each root-goal has a number of sub-goals. In the context of mission R.I.S.K, for a player to accomplish their root-goal they must have occupied every continent-sub-goal, this is done by performing Successful-Occupation actions on every territory contained in that continent.

For the purposes of this example we introduce another notation which are squares that represent territory entities. Arrows away from these squares are connected to the actions a player can perform when occupying this territory. This is done to visualize the state of the world at the time the action was performed. For example by occupying the territory Egypt, allows a player to perform the actions: SO-East-Africa, SO-North-Africa and SO-Middle-East.

Recall the formulae:

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

The term $Pr(goals)$, the prior probability of each root-goal, is $1/n$ where n is the number of mission cards and therefore for this example is $1/3$ for each root-goal.

The term $Pr(plans|goals)$ is 1.0 as we always consider the the full set of root-goals hence removing the need to calculate a choice on the part of the player.

The term $Pr(obs|exp)$ is computed only when an observation occurs.

Player p_1 initially occupies the territory North-Africa and thus can perform actions connected to the North-Africa square. p_1 then performs a Successful-Occupation-Western-Europe action.

With an observation we can now compute the term *Pr(exp* $\wedge$ *obs)*, the probability of the explanation that the player has a root-goal and O-Western-Europe for p_1.

First we must compute $Pr(obs|exp)$, the probability that p_1 would choose to perform the observation O-Western-Europe given each root-goal.

- Case 1 : $Pr(O\text{-}Western\text{-}Europe|O\text{-}AS\text{-}AF)$
- Case 2 : $Pr(O\text{-}Western\text{-}Europe|O\text{-}EU\text{-}SA\text{-}AF)$
- Case 3 : $Pr(O\text{-}Western\text{-}Europe|O\text{-}EU\text{-}SA\text{-}AS)$
- Cast 4 : $Pr(O\text{-}Western\text{-}Europe|O\text{-}AS\text{-}SA)$

The plan recognition agent first applies the *computeExpProb* operation for each of the above cases. Then using the *generatePS* operation to builds a full pending set for p_1.

After this the plan recognition agent retrieves a list of all consistent actions for each explanation by retrieving the ECA set from each explanations data structure.

Given the observation O-Western-Europe, the complete pending set of the player, and the set ECA. The plan recognition agent then uses the *computeObsProb* operation to compute the probability of choosing the Successful-Occupation-Western-Europe action in the context of each root-goal.

For each root-goal the *computeObsProb* filters out all action types except Successful-Occupation actions, from both p_1 pending set and the ECA set. The result is the Successful-Occupation actions that are consistent with the explanation *expConAct* and a new pending set *pTotalActNum* which contain the following Successful-Occupation actions given the state of the world in this example:

$$p_1PS = \{\text{SO-Western-Europe, SO-Brazil, SO-Egypt, SO-Congo, SO-East-Africa}\}$$

Given these two sets we count the number of consistent actions that the players pending set contains for each respective root-goal, by comparing what actions are contained in *pTotalActNum* to each root-goals *expConAct* set. We also count the total number of actions a player can perform *pTotalActNum*.

In this example it is currently $pTotalActNum = 5$

The counts given the action Successful-Occupation-Western-Europe is as follows:

Explanation	**Number of Consistent Actions**
O-AS-AF	3
O-EU-SA-AF	5
O-EU-SA-AS	2
O-AS-SA	1

Table 3.4: Consistent Action Count

With these counts and our assumed w the plan recognition agent then computes a *base* action probability for each explanation with the $computeBaseWeight$ operation as follows:

Explanation	Total-Weight	Left-Over	Base
O-AS-AF	0.02 * 3	1.0 - 0.06 = 0.94	0.188
O-EU-SA-AF	0.02 * 5	1.0 - 0.10 = 0.9	0.18
O-EU-SA-AS	0.02 * 2	1.0 - 0.04 = 0.96	0.192
O-AS-SA	0.02 * 1	1.0 - 0.02 = 0.98	0.196

Table 3.5: Consistent Action Count

Consistent actions *cons* are weighted so the probability of a consistent action $Pr(cons) = base + w$

Inconsistent actions *incons* are not weighted so the probability of a inconsistent action $Pr(incons) = base$

Explanation	$Pr(cons)$	$Pr(incons)$
O-AS-AF	0.208	0.188
O-EU-SA-AF	0.2	0.18
O-EU-SA-AS	0.212	0.192
O-AS-SA	0.216	0.196

Table 3.6: Computing Probabilities of Consistent/Inconsistent Actions

The plan recognition agent then computes an explanation that the players plan is the root goal given the observation *Pr(exp ∧ obs)* by multiplying the term $Pr(goals)$ with one of the two action probabilities, depending on whether the action that was observed was consistent or not, in other words if was contained in ECA or not.

For example to compute *Pr(exp ∧ obs)* for the explanation O-EU-SA-AF the plan recognition agent would choose to multiply by $Pr(cons)$ and so the calculation would be 1/3 * 0.2

Explanation	**Terms Multiplied**	*Pr(exp* $\wedge$ *obs)*
O-AS-AF	$Pr(goals)$ * $Pr(incons)$	0.0626
O-EU-SA-AF	$Pr(goals)$ * $Pr(cons)$	0.0666
O-EU-SA-AS	$Pr(goals)$ * $Pr(cons)$	0.706
O-AS-SA	$Pr(goals)$ * $Pr(incons)$	0.0653

Table 3.7: Computing Un-Normalised Root-Goal Probabilities

Finally to complete the calculation of *Pr(exp|obs)* the plan recognition agent normalises each with the following formula:

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

For example to compute *Pr(exp|obs)* for O-EU-SA-AF, the plan recognition agent would first sum all explanation probabilities which is:

$0.0626 + 0.0666 + 0.0706 + 0.0653 = 0.2651$

It would then divide the probability of the term *Pr(exp|obs)* for the O-EU-SA-AF root-goal which is:

$0.0666 \ / \ 0.2651 = 0.251$.

Explanation	*Pr(exp|obs)* Pre-Normalisation	*Pr(exp|obs)* Post-Normalisation
O-AS-AF	0.0626	0.236
O-EU-SA-AF	0.0666	0.251
O-EU-SA-AS	0.0706	0.266
O-AS-SA	0.0653	0.246

Table 3.8: Consistent Action Count

At this point we set each normalised value of *Pr(exp|obs)* as the value of each explanation and choose the highest as the prediction of the plan recognition agent. This is done to keep the explanation values from dropping to zero too quickly.

Failed-Occupation actions are computed in the exact same way as Successful-Occupation actions, the difference being that the pre-defined w is set to a lower value than that of Successful-Occupation.

The computation of other actions in the environment is considerably simpler given the assumptions of the model. To compute defence and movement actions the plan recognition agent would first retrieve the current explanation probability. Then firstly taking into consideration, whether it was an action performed on a territory that was contained in the high-level root-goal. If so then whether the action was consistent or not it would multiply it by for example:

- 1.02 for consistent actions.
- 0.98 for in-consistent actions.

The final computed probability would then again be normalised and set to the current explanation probability as normal.

For reinforce actions the plan recognition agent considers each explanations, if the reinforce action is observed as consistent the current explanations probability is multiplied by 1.02 if inconsistent then it is multiplied by 0.98.

3.4 Summary

CHECK SUMMARY TO SEE IF NOT MISSED THINGS

The design of the plan recognition agent is based on the PHATT algorithm. Various environmental modelling and design of environment data structures was necessary in order to allow the PHATT algorithm to be applicable to the R.I.S.K environment.

Each action of the R.I.S.K environment was modelled and various assumptions about each made in-order that each can be used to compute the probability of a players root-goal. Where necessary data structures were designed for the most significant objects in the environment. The purpose of each data structure would be to contain the essential information required by the plan recognition agent.

In particular a key concept in the design of the agent is that the pending set of a player is decided *a priori* based on the territories a player owns and that by combining the action possible from each territory that a player owns into a single set, all the actions a player can perform can be captured.

4. Implementation

The Java programming language was selected for the implementation of the plan recognition agent. Reasons for doing so were:

- The availabilty of an open source project with an active development team.
- Familiarily with the language.
- Cross platform.
- High Quality Integrated Development Environments.

4.1 System Architecture

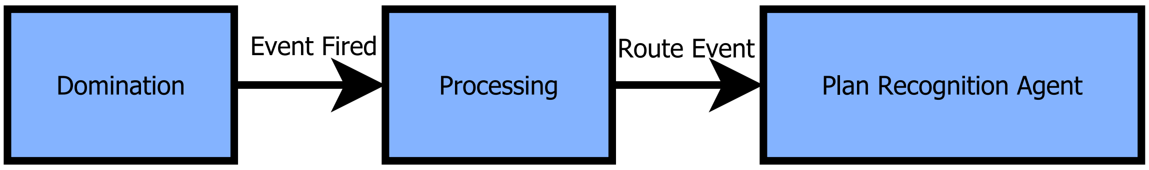

Figure 4.1: System Architecture

The plan recognition agent followed an *event-driven* system architecture.

DESCRIBE THE DIFFERENT PARTS OF THE DIAGRAM ONCE ITS BEEN CHANGED!

The system can be seen as three components.

The Domination Program refers t

Modifications to the Domination program included automating the output of the replay file, as well as lines of code to fire events into the processing class.

The plan recognition algorithm also had access to the data structures in the Domination program rather than storing its own copy avoiding and problems potentially because of losing synchronisation between the two.

The processing class is responsible for routing events to the plan recognition algorithm

When events 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:

Event Type	**Contains Variables**	**Purpose**
New-Player	playerName	Initialises a new player data structure with ID *playerName*.
Remove-Player	playerName	Destroys the player data structure with ID *playerName*.
Successful-Occupation	attackerName, cName, tName	Signals that the player named *attackerName* performed a Successful-Occupation.
Failed-Occupation	attackerName, cName, tName	Signals that *attackerName* performed a FailedOccupation.
Successful-Defence	defenderName, cName, tName	Signals that *defenderName* performed a SuccesfulDefence.
Failed-Defence	defenderName, cName, tName	Signals that *defenderName* performed a FailedDefence.
Army-Movement	playerName, cName, tName	Signals that *playerName* performed an ArmyMovement.
Reinforce	playerName, cName, tName	Signals that *playerName* performed a Reinforce action.

Table 4.1: Event Object Specification, c = Continent, t = Territory

At the end of a game the domination program and the plan recogntion agent would output two files:

The domination program would output a file that would be used for data collection.

The plan recognition agent would output a csv file containg the prediction of the plan recogntion agent over the course of the game.

TALK MORE ABOUT THE SYSTEM!

The architecture of the system is such that 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 implementation of a plan recognition agent.

4.2 Plan Recognition Agent Architecture

PLAN RECOGNTION AGENT ARCHITECTURE-EXTENSION OF DIAGRAM INCLUDE PLAN RECOGNITION AGENT SUB DATA STRUCTURES - EXTENSION OF DIAGRAM TO INCLUDE OUTPUTTING OF CSV FILE

The plan recognition agent has a data structure for each player which in turn has several data structures of the explanation that are initially hardcoded. Each explanation object calculate its own probability.

When an event arrives from the processing class it is routed to its specified player data structure and then to its specifiied explanation data structure.

Data structure that stores the probabilities of each players expalantion structure once this game ends the data structure writes it out to a data folder as a csv file.

Given a list of hardcoded explanations the plan recognition could be used on any map that might be developed for the Domination Program.

4.3 System Operation

To operate the plan recognition agent needs to be aware of several 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 a game begins, the plan recognition agent initalises a list of hardcoded explanations, these objects contain all the necessary information from the environment to computes its own probability.

The plan recognition agent also then initialises a set of player data strucutres. This complete list of explanations is cloned into each players data structure.

4.4 Additional Implementation Concepts

4.4.1 Plan Recognition Agent Map Portability

Since explanations in the environement can be hardcoded without affecting the plan recogntion agents core structures, given a different list of hardcoded explanations the plan recognition can be used on any map that might be developed for the Domination Program.

4.4.2 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.

4.4.3 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.

4.5 Summary

The plan recognition agent was written in the Java programming language following an event-driven system architecture. It operated in parallel to the Domination program receiving events fired from the program to update data structures it held locally that contained information about each players explanation predictions. At the end of a game the domination program as well as the plan recognition agent would output two files allowing the replay of a game and a record of the plan recognition agents prediction probabilities for each players explanations over the couse of the game.

5. Evaluation

The evaluation plan consisted of three experiments with the plan recognition agent:

1. Free Play
2. Constrained Play
3. A.I Play

Free Play can simply be described as human players "playing to win". In this manner it is how people would normally play mission R.I.S.K.

Constrained Play consists of human players only performing 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.

This artificially constructed form of play is designed to be the easiest to perform plan recognition on due to the lack of environmental noise fron in-directly consistent and inconsistent actions found in other forms of play. In doing so a high correct prediction accuracy is expected and if this is not the case then the resulting experiments will provide a good indication into why.

A.I. Play is games using the Domination programs existing implementations of A.I. players for its various game modes only. These A.I. players can be substituted for human players to the point where a game can consists of only A.I. players' fighting each other, and by doing so can complete a game in a fraction of the time that human players do. By automating the collection of these data samples using an free macro program called AutoHotkey **[ref to autohotkey]**, provided an opportunity for the collection of a large number of data samples to analyse.

5.1 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 R.I.S.K. either by:

1. Eliminating all other players from the game.
2. 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 that plan during the course of the game.

Since the A.I. for the game has its decisions skewed, varying this should in theory cause the A.I. to perform either better or worse depending on whether decisions are skewed towards decisions that are more consistent with the mission or away from decisions that are consistent with the mission but are rather focused on occupying continents that the A.I. has advantages in or eliminating players who are weak.

Experiments were performed with with different values of w to confirm this.

For the A.I. two experiments were performed with different values of w, 4 its normal setting and 100 a significantly higher value.

5.2 Data Collection

After a game had finished the Domination program would generate a replay file, which if saved could be loaded at a later point through a specialised user interface for debugging created by the developers allowing a completely-independent and exact replay of the game.

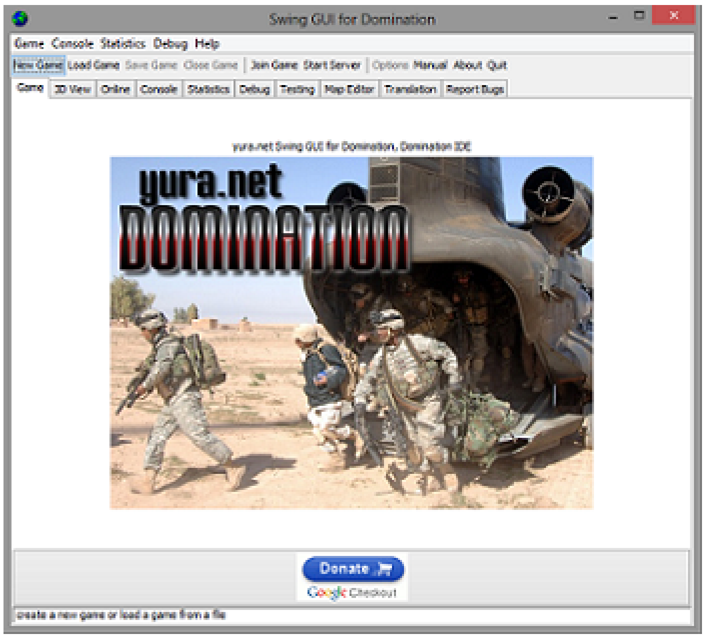
Swing GUI for Domination
Game Console Statistics Debug Help
New Game Load Game Save Game Close Game Join Game Start Server Options Manual About Quit
Game 3D View Online Console Statistics Debug Testing Map Editor Translation Report Bugs
yura.net Swing GUI for Domination, Domination IDE
yura.net
DOMINATION
Donate
Google Checkout
create a new game or load a game from a file

Figure 5.1: Domination Debug Graphical User Interface

Using this tool had significant implications, in that it would possible to re-observe games to assist with evaluation. It would also allow for testing of the plan recognition agent again with different modelling and/or calibrations in an exact same game and by doing so eliminate the environment as a variable between comparisons of the plan recognition agent.

In addition to Dominations replay file, the plan recognition agent would simultaneously generate a csv file recording the significant features of a game, namely:

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

Scripts written in the Python programming language were then used to automate the analysis of these generated csv files.

5.3 Experimental Findings

Experiments with the plan recognition agent revealed the following insights:

1. The prediction accuracy of the plan recognition agent is highest in constrained play.
2. The prediction accuracy of the plan recognition agent is HOW MUCH PERCENT higher for players who won than players who lost.
3. The prediction accuracy of the plan recognition agent is significantly higher for human players than the domination A.I. players.
4. From the prediction accuracy of the plan recognition agent, we can infer when the domination A.I. gives less significance to completing its missions.
5. The length of game has a positive effect on the accuracy of the prediction agent.

Keeping in mind the initial hypothesis, we attempt to evaluate whether plan recognition algorithms are beneficial in games, specifically R.I.S.K.

To do this we first comment on general trends, then we explore two questions for both human and A.I players:

The following sections will be made up of an analysis of the prediction accuracy of the plan recognition agent for players in *General*, then we analyse the data further by decomposing players into players who won *Winners* and players who lost *Losers*.

- Does the plan recognition agent converge on the correct explanation?
- How fast does the plan recognition agent converge on the correct explanation?

How the game is played out for each player or their *game-scenarios*, deciedes whether a player is a winner or loser. The main game-scenarios that the plan recognition agent faces are the following:

Winners who won *quickly* or *slowly* and who spent the majority of their game:

- fighting over non-mission continents.
- fighting over mission continents.

Losers who were eliminated *quickly* or *slowly* or who were *not-eliminated* and who spent the majority of their game:

- fighting over non-mission continents.
- fighting over mission continents.

		Mission	Non-Mission
Winner	Quickly		
	Slowly		
Loser	Quickly		
	Slowly		
	Not-Eliminated		

Table 5.1: Game-Scenario Table

These cases can be compactly represented in a definitional framework.

Using table 5.1 we classify players into their game-scenarios on game-by-game basis. As there can be only one winner, the winner row only ever has one entry, whereas the loser row can have 1 to 5 entries.

5.3.1 Explanation Convergence Format

EXAMPLE OF STANDARD SUCCESSFUL CONVERGENCE OF AN EXPLANATION

At the start of a game forty-two turns are required to place one army on each territory of the R.I.S.K map. These observations are not taken into account by the plan recognition agent as the distribution of territories between players is automatic and random.

After this the remaining initial armies are distributed by players on to their territories. During this the probability of explanations that are consistent with the actions a player is performing will slowly rise and conversely those that explanations which with the actions are inconsistent will slowly fall.

Once this initial setup is complete (which could be from 80 to 120 turns depending on the number of players) the attack phase begin for the first player p_c, at this time the probabilities of p_c explanations may change rapidly depending on the number of actions that p_c performs in their turn. The most significant change occurs when there is a high number of actions. This is due to the plan recognition agents model which has been built to give attack actions the greatest significance.

After p_c turn is over, other players begin their attack phases in-turn, and unless the action another player performs is to attack a territory owned by p_c, the probabilities of p_c explanations remained unaffected.

This continues till the end of the game at which point the probability of a players explanation will have either converged successfully or unsuccessfully to the correct explanation.

5.3.2 Prediction Accuracy in General

Given there are six mission cards, the accuracy of a random guess of a players mission card is 16.67%. According to Table 7.1, at its worse the plan recognition agent in general performs 8.47% better than this, and at its best 63.72% better.

Of free and constrained play table 7.1 reveals that *constrained play had a significantly higher accuracy than both free and A.I. play.* This is unsurprising given its artificial nature. Designed to be ideal the scenario for the plan recognition agent, it is therefore in itself not a good measure of evaluating the success of the plan recognition agent, but in conjunction with Free Play gives us a better picture of how well the plan recognition agent performed.

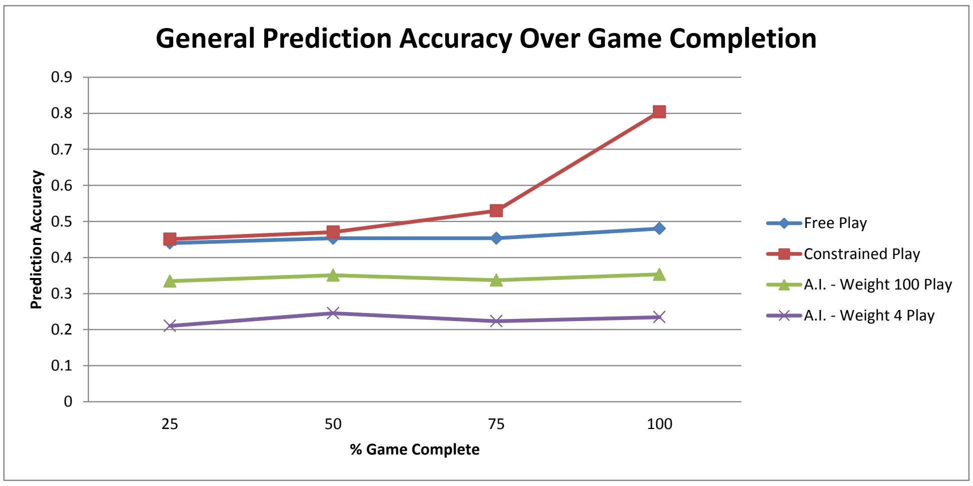

Figure 5.2: General Prediction Accuracy

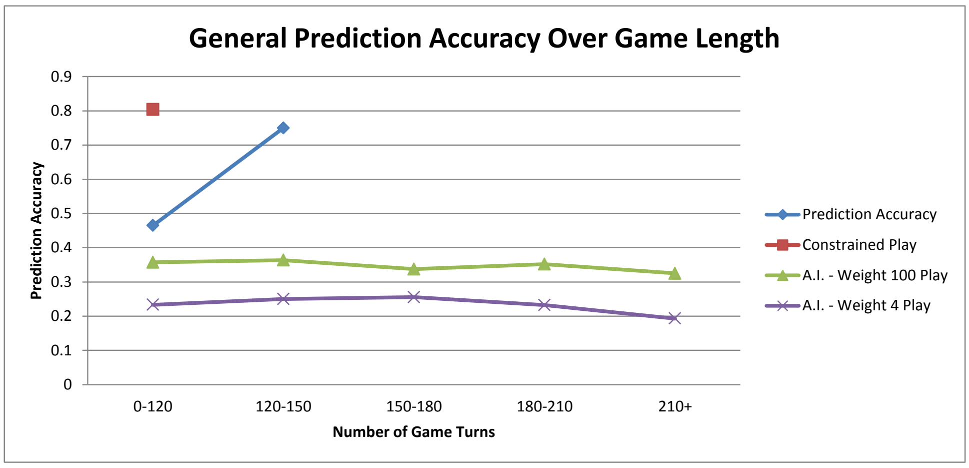

Figure 5.3: General Prediction Accuracy

Figure 5.2 details the general prediction accuracy of the plan recognition agent at intervals of 25% of the games duration.

Free and A.I. play are definitely a more natural form of playing. Players in these perform actions that are either indirectly-consistent or inconsistent primarily because their plan at that point may be different to that of their mission. So a better measure for the general accuracy of the plan recognition agent for humans is rather Ftree Play as prediction accuracies of human players is most significant. Therefore *the plan recognition agent is 32.61% better than random guess* and so we can establish that the plan recognition agent indeed does converge on the correct explanation 49.28% of the time.

From this we can infer several important trends.

In general the plan recognition of the different types of play the plan recognition agent performed significantly better in *constrained play had a than both free and A.I. play.* This is unsurprising given its artificial nature. Designed to be ideal the scenario for the plan recognition agent, it is therefore in itself not a good measure of evaluating the success of the plan recognition agent, but in conjunction with Free Play gives us a better picture of how well the plan recognition agent performs for human players.

In general the plan recognition agent performed better on human players than A.I. players. The reason for this is due primarily to the way in which the humans and the A.I. differ in operation. According to the A.I developer at Yura.Net "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. "

There is a significant improvements in the prediction accuracy of the plan recognition agent from $w4$ to $w100$ and if the aim of the A.I. was to perform its mission quickly and in doing so effectively then this would be confirmed by the readings from the plan recognition agent.

Changing the variable w hence explains the plan recognition agents increase in accuracy of up to %10 as the A.I.'s decisions becomes more skewed towards occupying mission continents and less towards occupying accessible non mission-continents and eliminating opponents both of which are observed as inconsistent actions by the plan recognition agent.

There is a significant increase in the prediction accuracy of the plan recognition agent in constrained play in the 75 to 100% of game play complete range. This is because in the later part of any game players can perform more actions per turn due to the increasing number of armies on the board.

The general prediction accuracy over game length appears to fall for A.I over time whereas human players appear to be increasing. Though this is an interesting contrast from figure 5.2 which indicates that prediction accuracy over game length appear to be stable and in the case of constrained play increases significantly.

In general it appears the shorter the game the better the prediction of the plan recognition will be, but

The length of all human games is much lower that that of A.I. games this is either one of two reasons:

- A.I. prevents other A.I. from winning and therefore game is longer.
- A.I is worse and therefore game is longer because they cannot win.

With an example of the longest game we can see that this is not the case.

GET EXAMPLE OF LONG GAME

In this example the A.I. could have won but did not because it considered eliminating its opponents and occuping neibouring continents more significant that completing its mission despite owning all territories except one.

This observation also happens to coincide with the signifanct drop of A.I. - 4 prediction accuracy in games longer than 210+, but looking at figure 1 we must conclude that the number of data samples of this length is relatively small given that figure 5.3

5.4 Plan Recognition Agent Winner Prediction Accuracy

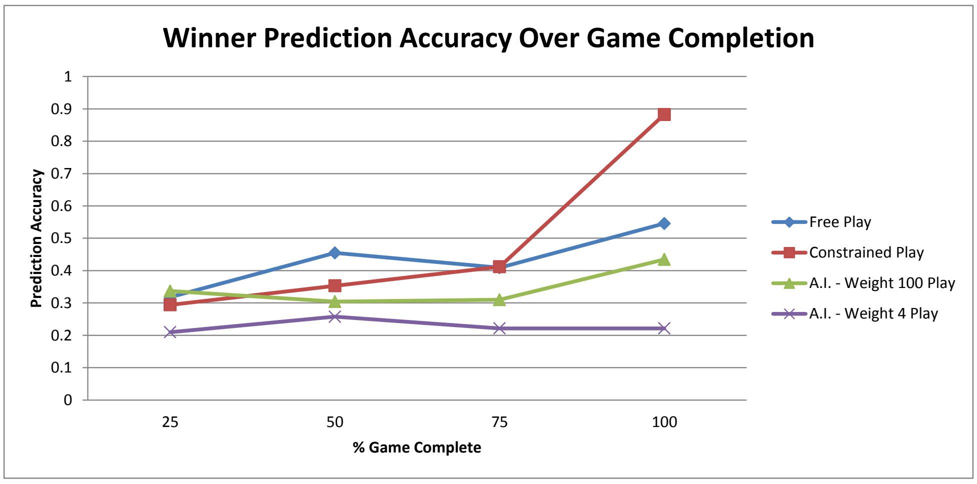

Figure 5.4: Winner Prediction Accuracy

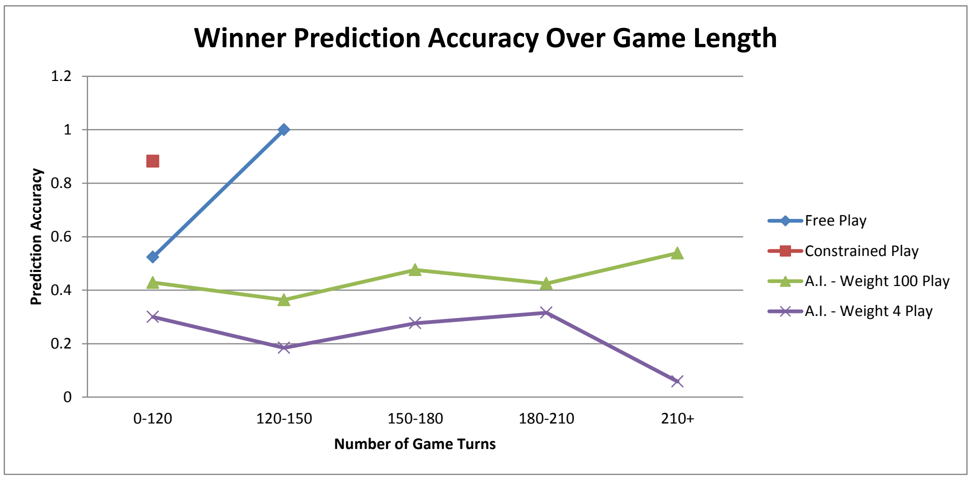

Figure 5.5: Winner Prediction Accuracy

MAKE TABLE OF AVERAGE ACCURACY

Most prominantly we must point out that the prediction accuracy of is 1.0 is due to the number of data samples, there was only a single data sample of a winner that was correctly predicted between 120-150 turn long games.

Winners by definition have successfully performed their mission or will have eliminated every player, in either case they will have performed each consistent action for a mission atleast once. Making on average as the data reflects the number of consistent actions performed by the winner, on average higher than that of losers hence making it easier to detect.

On the other hand though there are of course cases where the plan recognition agent is unsuccessful:

There are cases where a winners has occupied the entire last continent for their mission in one turn and since sampling is done on a turn by turn basis the plan recognition agent will miss the significant change in probabilities as the system does not register an end of turn but rather a different event which is the end of the game.

Except for const

Therefore the shorter the game the more likely the prediction will be correct for the winner

Comparison of the Figure 5.4 to figure 5.6 shows that in every type of play the plan recognition agents *prediction accuracy of winners is better than that of losers.* The plan recognition agent is more successful at predicting due to a number of reasons:

5.5 Plan Recognition Agent Loser Prediction Accuracy

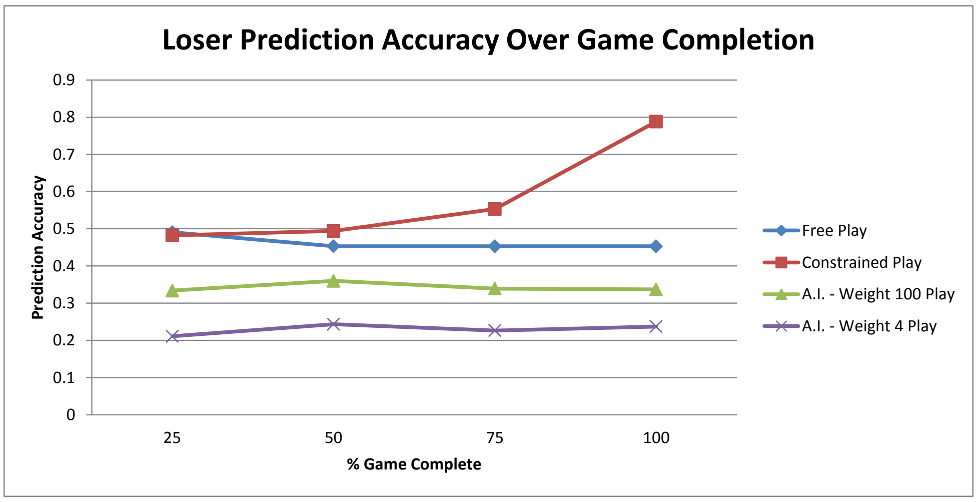

Figure 5.6: Domination Debug Graphical User Interface

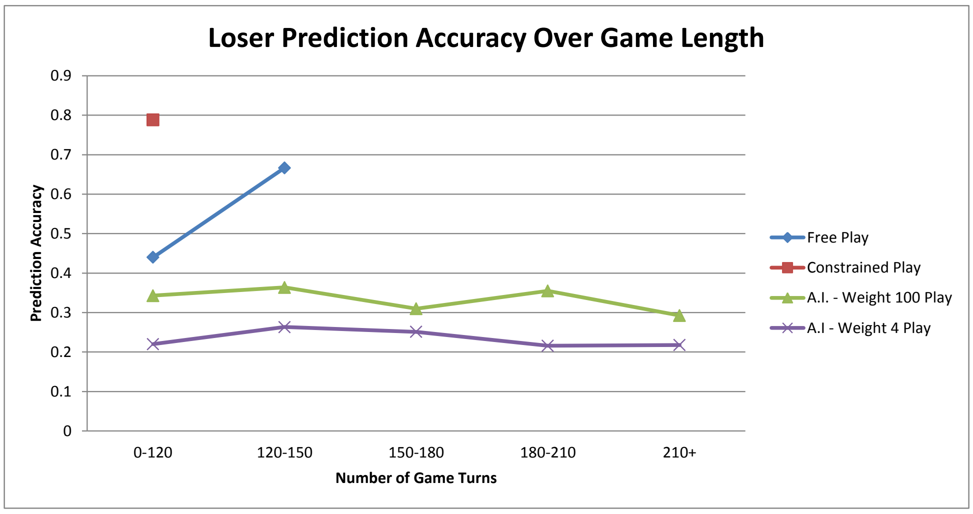

Figure 5.7: Winner Prediction Accuracy

MAKE TABLE OF AVERAGE ACCURACY

GENERAL REASONS WHY LOSER ACCURACY IS WORSE

Losers who were eliminated early

Losers are often prevented from successfully achieving their goal.

Present typically examples where this is the case.

Winners who win quickly

Winners who win slowly

Though there were exceptions to this? Misclassification due to association?

We will describe each case, present a hypothesis for the plan recognition agents prediction accuracy then present examples from the data for most significant of these cases to support the hypothesis.

but this is not necessarily true winners just like losers may be in positions where they cannot perform actions that are directly-consistent.

A proportion of incorrect predictions occur because losers can be eliminated resulting in a lack of data about the losers behaviour. If a loser was to be eliminated early on in the game then their explanations probabilities would *flat-line*, in that the prediction does not change for the remainder of the game. If the loser had been performing more inconsistent actions then consistent actions up until that point then the prediction will be wrong, inversely it may be correct for the very same reason.

WINNERS CONSTRAINED PLAY

Table 7.4 shows two incorrect predictions for winners in constrained play. Both were *Two-Continent+1* mission types and from analysis of each both are shown to be incorrect for the same reason.

SHOW A GRAPH OF ONE EXAMPLE OF WINNER CONSTRAINED WRONG

The final prediction probabilities of the above graph was split exactly equally between two different children-root-goals as their sub-goals were identical, thus resulting in an incorrect or argueably *inconclusive* prediction.

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

In the context of constrained play, players do not purposely seek to hinder the progress of other players' missions. The winner of constrained play is often the player with the least overlapping mission with other players and the smallest number of territories to occupy. This situation allows for a player to quickly complete their mission with little opposition from other players who do not interfere.

SHOW THAT CONSTRAINED PLAY GAMES ARE ON AVERAGE SHORTER THAN OTHER GAME TYPES CAN DO WITH GRAPH

AI 100 LOSERS GETS WORSE OVER GAME LENGTH

BIG DROP IN PREDICTION ACCURACY IN A.I 4 GAME OVERS 210 IN TURNS BUT SINCE THE NUMBER OF SAMPLES MUST BE SMALL HAS A LOW EFFECT ON THE GENERAL ACCURACY

5.6 Summary

Why is the accuracy of winners better than losers? May be because weighting of attack actions is towards successful attacks, of which winners perform more.

6. Conclusion

6.1 Main insights and results

In R.I.S.K.there is only one winner. R.I.S.K is an adversarial environment where right from the start you are attacked. Given that you can survive an onslaught of attacks in-order to achieve ones own mission, players must either directly or indirectly eliminate each other and often modify their own plan in order to continue surviving or prevent other players from winning.

This environment and the high-level of overlap between missions the results are many misclassifications.

Given the design of the plan recognition agent these changes in behaviour are see as directly-inconsistent with their own mission and rather seems consistent with whatever explanation that those actions *are* directly-consistent with instead.

The main issue facing the plan recognition agent is differentiating between noise from the environment.

The plan recognition agent should not only consider what occurs in the action space but the state of the environment as well as the state of the environment further indicates the plans of players.

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 as 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.

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

6.1.1 Improvement of Model

Currently the system suffers from the problem of distinguishing noise from meaninful actions.

Solving this problem could be taclked in two ways:

Identifying actions are in-directly consistent and modelling them to accurately contribute to explanations - further modelling to further diffferentiate noise from meaninggul behaviour.

Making the meaningful actions that one can already identify greater significance to actions that are directly consistent through the state of the game more significant - making noise less significant.

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.

Making a system of dynamically applying weights in a manner that is proportional to the state of the game e.g:

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.

ExplanationProb = ExplanationProb * Weight * Fraction representing state of environment

For example

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

The weight was computed in the normal manner but would be scaled up down by two factors in-order to make an actions more a less significant given the state of the environment.

6.2 Outcomes

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.

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!

6.3 Criticism

ACCURACY OVER LENGTH OF GAME SHOULD HAVE A SMALLER UNIT SO AS TO SEE TRENDS IN HUMAN DATA

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

Small number of data samples for free play and constrained winners, therefore analysis will likely not have covered all the cases.

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

How to differentiate between two identical alternates? E.g prediction is equal between EU SA AU and EU AU SA

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.

6.4 Future Work

A.I. play was done with six players, would be interesting to see the effects on prediction accuracy on games with different number of players.

THINK ABOUT WHY IT WOULD IMPROVE

6.4.1 Model Augmentation

ANALYSIS OF EACH EXPLANATION AND BUILD MODELS FOR EACH, DATA COLLECTED TO PERFORM BUT DID NOT HAVE TIME

6.4.1.1 Application of Machine Learning

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

6.4.1.2 More Sophisticated Action Probability Computations

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.

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. REMODEL MOVEMENT TO BE:

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

6.4.1.3 More Sophisticated Modelling of Current Model

Players in these perform actions that are either indirectly-consistent or inconsistent primarily because their plan at that point may be different to that of their mission. Either type of action reduces the probability of the correct explanation given the current model. MODEL TO TAKE THIS INTO ACCOUNT

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 scaling 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 territories 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 territories 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.

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.

6.4.1.4 Extension to Include Other Types of Mission Cards

Extension of the model to include other mission cards. Due to the design, eliminating players could be modelled but modelling Occupy 24, 18 due to design decision has such high overlap with all other missions it would be hard to model this.

A proposed method of modelling eliminating player missions

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.

6.4.2 Discussion of Possible Applications

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.

6.5 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.

WHY USE THIS ALGORITHM WHY IS SUITABLE END ON A POSITIVE NOTE

7. Appendix

7.1 Accuracy Count Measurements

7.1.1 General Prediction Accuracy

		% Game Played							
		25		50		75		100	
Type of Play	**Player Type**	**C**	**I**	**C**	**I**	**C**	**I**	**C**	**I**
Free	Winner	7	15	10	12	9	13	12	10
	Loser	26	27	24	29	25	28	24	29
Constrained	Winner	5	12	6	11	7	10	15	2
	Loser	41	44	42	43	47	38	67	18
A.I. - Weight 100	Winner	62	122	56	128	57	127	80	104
	Loser	307	613	331	589	312	608	310	610
A.I. - Weight 4	Winner	35	132	43	124	37	130	37	130
	Loser	176	659	203	632	189	646	198	637

Table 7.1: Winner-Loser Game-Length Correct-Incorrect Prediction Count, **C** = Correct, **I** = Incorrect

	% Game Played							
	25		50		75		100	
Type of Play	**C**	**I**	**C**	**I**	**C**	**I**	**C**	**I**
Free	33	42	34	41	34	41	36	39
Constrained	46	56	48	54	54	48	82	20
A.I. - Weight 100	369	735	387	717	372	732	390	714
A.I. - Weight 4	211	791	246	756	224	778	235	767

Table 7.2: General Game-Length Correct-Incorrect Prediction Count, **C** = Correct, **I** = Incorrect

7.2 Game Length Correct-Incorrect Prediction Count

		Number of Game Turns									
		< 120		120-150		150-180		180-210		210+	
Type of Play	**Player Type**	**C**	**I**	**C**	**I**	**C**	**I**	**C**	**I**	**C**	**I**
Free	Winner	11	10	1	0	0	0	0	0	0	0
	Loser	22	28	2	1	0	0	0	0	0	0
Constrained	Winner	15	2	0	0	0	0	0	0	0	0
	Loser	67	18	0	0	0	0	0	0	0	0
A.I. - Weight 100	Winner	9	12	20	35	20	22	17	23	14	12
	Loser	36	69	100	175	65	145	71	129	38	92
A.I. - Weight 4	Winner	3	7	7	31	13	34	12	26	2	32
	Loser	11	39	50	140	59	176	41	149	37	133

Table 7.3: Game-Length Correct-Incorrect Prediction Count, **C** = Correct, **I** = Incorrect

	Number of Game Turns									
	< 120		120-150		150-180		180-210		210+	
Type of Play	**C**	**I**	**C**	**I**	**C**	**I**	**C**	**I**	**C**	**I**
Free	33	38	3	1	0	0	0	0	0	0
Constrained	82	20	0	0	0	0	0	0	0	0
A.I. - Weight 100	45	81	120	210	85	167	88	162	52	108
A.I. - Weight 4	14	46	57	171	72	210	53	175	39	163

Table 7.4: General Game-Length Correct-Incorrect Prediction Count, **C** = Correct, **I** = Incorrect

		Mission	Non-Mission
Winner	Quick		
	Slow		
	Quick		
Loser	Slow		
	Till End		

7.3 Explanation Data

7.4 Template Tables

		% Game Played							
		25		50		75		100	
Type of Play	**Mission Card**	**C**	**I**	**C**	**I**	**C**	**I**	**C**	**I**
Free	Occupy-Europe-Australia-One	4	1	4	1	3	2	3	2
	Occupy-Asia-South-America	0	3	0	3	0	3	2	1
	Occupy-Europe-South-America-One	2	1	2	1	1	2	0	3
	Occupy-North-America-Africa	0	5	3	2	3	2	4	1
	Occupy-Asia-Africa	0	2	0	2	1	1	1	1
	Occupy-North-America-Australia	1	3	1	3	1	3	2	2
Constrained	Occupy-Europe-Australia-One	2	0	2	0	2	0	1	1
	Occupy-Asia-South-America	0	0	0	0	0	0	0	0
	Occupy-Europe-South-America-One	3	2	3	2	4	1	4	1
	Occupy-North-America-Africa	0	5	1	4	1	4	5	0
	Occupy-Asia-Africa	0	0	0	0	0	0	0	0
	Occupy-North-America-Australia	0	5	0	5	0	5	5	0
A.I. - Weight 100	Occupy-Europe-Australia-One	32	4	20	16	15	21	15	21
	Occupy-Asia-South-America	3	15	3	15	6	12	9	9
	Occupy-Europe-South-America-One	13	4	8	9	8	9	9	8
	Occupy-North-America-Africa	2	34	7	29	6	30	15	21
	Occupy-Asia-Africa	9	33	7	35	9	33	14	28
	Occupy-North-America-Australia	3	32	11	24	13	22	18	17
A.I. - Weight 4	Occupy-Europe-Australia-One	21	12	15	18	8	25	6	27
	Occupy-Asia-South-America	1	28	8	21	8	21	8	21
	Occupy-Europe-South-America-One	11	23	13	21	13	21	14	20
	Occupy-North-America-Africa	0	16	1	15	3	13	3	13
	Occupy-Asia-Africa	1	28	1	28	0	29	1	28
	Occupy-North-America-Australia	1	25	5	21	5	21	5	21

Table 7.5: Winner Explanation Game-Length Correct-Incorrect Prediction Count, **C** = Correct, **I** = Incorrect

		% Game Played							
		25		50		75		100	
Type of Play	**Mission Card**	**C**	**I**	**C**	**I**	**C**	**I**	**C**	**I**
Free	Occupy-Europe-Australia-One	7	3	5	5	6	4	3	7
	Occupy-Asia-South-America	2	7	3	6	4	5	6	3
	Occupy-Europe-South-America-One	13	3	12	4	11	5	8	8
	Occupy-North-America-Africa	0	6	0	6	0	6	1	5
	Occupy-Asia-Africa	3	5	3	5	3	5	4	4
	Occupy-North-America-Australia	1	3	1	3	1	3	2	2
Constrained	Occupy-Europe-Australia-One	11	4	11	4	10	5	12	3
	Occupy-Asia-South-America	7	10	8	9	9	8	16	1
	Occupy-Europe-South-America-One	6	6	5	7	5	7	6	6
	Occupy-North-America-Africa	5	7	6	6	6	6	7	5
	Occupy-Asia-Africa	9	8	9	8	11	6	15	2
	Occupy-North-America-Australia	3	9	3	9	6	6	11	1
A.I. - Weight 100	Occupy-Europe-Australia-One	128	20	90	58	43	105	36	112
	Occupy-Asia-South-America	6	160	26	140	40	126	41	125
	Occupy-Europe-South-America-One	119	48	113	54	107	60	118	49
	Occupy-North-America-Africa	20	128	32	116	46	102	43	105
	Occupy-Asia-Africa	26	116	27	115	25	117	23	119
	Occupy-North-America-Australia	8	141	43	106	51	98	49	100
A.I. - Weight 4	Occupy-Europe-Australia-One	77	57	51	83	31	103	28	106
	Occupy-Asia-South-America	5	133	16	122	16	122	19	119
	Occupy-Europe-South-America-One	71	62	66	67	67	66	74	59
	Occupy-North-America-Africa	8	143	21	130	23	128	25	126
	Occupy-Asia-Africa	10	128	14	124	9	129	8	130
	Occupy-North-America-Australia	5	136	35	106	43	98	44	97

Table 7.6: Loser Explanation Game-Length Correct-Incorrect Prediction Count, **C** = Correct, **I** = Incorrect

		% Game Played							
		25		50		75		100	
Type of Play	**Mission Card**	**C**	**I**	**C**	**I**	**C**	**I**	**C**	**I**
Free	Occupy-Europe-Australia-One	11	4	9	6	9	6	6	9
	Occupy-Asia-South-America	2	10	3	9	5	7	8	4
	Occupy-Europe-South-America-One	15	4	14	5	12	7	8	11
	Occupy-North-America-Africa	0	11	3	8	3	8	5	6
	Occupy-Asia-Africa	3	7	3	7	3	7	5	5
	Occupy-North-America-Australia	2	6	2	6	2	6	4	4
Constrained	Occupy-Europe-Australia-One	13	4	13	4	12	5	13	4
	Occupy-Asia-South-America	7	10	8	9	9	8	16	1
	Occupy-Europe-South-America-One	9	8	8	9	9	8	10	7
	Occupy-North-America-Africa	5	12	7	10	7	10	12	5
	Occupy-Asia-Africa	9	8	9	8	11	6	15	2
	Occupy-North-America-Australia	3	14	3	14	6	11	16	1
A.I. - Weight 100	Occupy-Europe-Australia-One	59	125	160	24	110	74	51	133
	Occupy-Asia-South-America	47	137	9	175	29	155	50	134
	Occupy-Europe-South-America-One	116	68	132	52	121	63	127	57
	Occupy-North-America-Africa	53	131	22	162	39	145	58	126
	Occupy-Asia-Africa	34	150	35	149	34	150	37	147
	Occupy-North-America-Australia	63	121	11	173	54	130	67	117
A.I. - Weight 4	Occupy-Europe-Australia-One	98	69	66	101	37	130	34	133
	Occupy-Asia-South-America	6	161	24	143	24	143	27	140
	Occupy-Europe-South-America-One	82	85	79	99	80	87	88	79
	Occupy-North-America-Africa	8	159	22	145	26	141	28	139
	Occupy-Asia-Africa	11	156	15	152	9	158	9	158
	Occupy-North-America-Australia	6	161	40	127	48	119	49	118

Table 7.7: General Game-Length Correct-Incorrect Prediction Count, **C** = Correct, **I** = Incorrect

		Turns									
		< 120		120-150		150-180		180-210		210+	
Type of Play	**Player Type**	**C**	**I**	**C**	**I**	**C**	**I**	**C**	**I**	**C**	**I**
Free	Winner										
	Loser										
Constrained	Winner										
	Loser										
A.I. - Weight 100	Winner										
	Loser										
A.I. - Weight 4	Winner										
	Loser										

Table 7.8: General Game-Length Correct-Incorrect Prediction Count, **C** = Correct, **I** = Incorrect

Bibliography

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

[2] Carberry, Sandra. Techniques for Plan Recognition. *User Modeling and User-Adapted Interaction*, 11(1-2):31–48, March 2001.

[3] Eugene Charniak and Robert P. Goldman. A Bayesian Model of Plan Recognition. 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] 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.

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

[7] 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.

[8] Robert P. Goldman, Christopher W. Geib, and Christopher A. Miller. A New Model of Plan Recognition. *Artificial Intelligence*, 64:53–79, 1999.

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

[10] Juan Lozano, Dane Bratz. A Risky Proposal: Designing a Risk Game Playing Agent. Technical report, Stanford, 2012. (Machine Learning Final Project).

[11] 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.

[12] Michael Fagan and PÃądraig Cunningham. Case-Based Plan Recognition in Computer Games. In *Proceedings of the Fifth ICCBR*, pages 161–170. Springer, 2003.

[13] Matthew Molineaux, David W. Aha, and Gita Sukthankar. G.: Beating the defense: Using plan recognition to inform learning agents. In *In: Proceed-*

ings of Florida Artifical Intelligence Research Society, AAAI, pages 337–343. Press, 2009.

[14] 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.

[15] 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.

[16] 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.

[17] Michael Wolf. *An Intelligent Artificial Player for the Game of Risk*. PhD thesis, Darmstadt University of Technology, 2005. (Unpublished Doctoral Dissertation).