University of Edinburgh
School of Informatics

Plan Recognition in Risk

4th Year Project Report
Artificial Intelligence and Software Engineering

Jibran A.Z. Khan & Dr. Michael Rovatsos

April 4, 2013

Abstract: This paper presents the design and implementation of a plan recognition agent for the board game Risk. The agent is based on Geib and Goldman's Probabilistic Hostile Agent Task Tracker (PHATT) algorithm [8]. The agent acts as an observer to the game with the goal of inferring the unknown mission cards based on player actions in real time. We then go on to evaluate the plan prediction accuracy of the agent with four different experiments. The results of these experiments are presented then analysed. We also discuss the project outcomes and propose future work.

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. For 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 been 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.

I'd also like to thank 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 to all my friends. In particular, Nick La Rooy for his advice in troubled times and as a constant 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 always been supportive through these four long years.

Last but not least thank you 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, commonly termed 'game A.I.', has been an area of research since the beginning of significant work in the Artificial Intelligence field. Though techniques for game A.I. have typically come from academia, as John Funge argues [7] that academic A.I. and game A.I. are notably different in both scope and application.

Since the primary goal of academia is to further human understanding, often by solving particularly complex problems, the scope of its A.I. techniques are more general in their application.

Game A.I., on the other hand, is built to provide an enjoyable experience for those playing the game, usually by creating the illusion of intelligence. The goal of game A.I. is to control the game environment in a manner that pushes it towards states the gameâĂŹs designer has chosen.

Companies in the video game industry who utilise 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 hardware shader techniques and the move from two to three dimensional worlds has kept their customersâĂŹ interest over the years.

Emerging though is the idea that as users become accustomed to high quality graphics, developers will require something new to distinguish their product. This 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 utilises a S.T.R.I.P.S. style planning-architecture, which developer Jeff Orkin termed Goal-Oriented Action Planning [16]. The game gained critical acclaim and is commonly referenced as an example of excellent game A.I.

With this shift in focus, there are growing incentives for developers to create more sophisticated game A.I. As in the past, this will likely using techniques from

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

Plan recognition is the problem of inferring an agent's plans from observed actions. Significant research in plan recognition began in the 1980's, the results of which have had numerous applications in several fields. Nate Blaylock highlights [2] 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 provides the ability to give highly personalized responses and identify threats from malicious agents. As such, plan recognition algorithms continue to receive attention from various Computer Science communities.

In video games, plan recognition has been used to perform dynamic analysis in game environments such as Real Time Strategy [19] and Multi-User Dungeons [6]. More exciting though is the prospect of combining plan recognition algorithms with planning-architectures such as G.O.A.P. or using machine learning to further augment A.I. The desired result is more sophisticated A.I. that is capable of recognising plans and altering its behaviour accordingly.

1.2 Artificial Intelligence and Board Games

In 1950 Alan Turing published the landmark paper "Computing Machinery and Intelligence", establishing the Turing test, marking what many consider to be the birth of the Artificial Intelligence field. 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" [11].

Interestingly though, it was thirty five years earlier that Leonardo Torres y Quevedo built "El Ajedrecista", a chess automaton capable of playing a king and rook endgame against a king from any position. Considered the world's 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 [1] and a solution to the game checkers by Jonathan Schaeffer et al in 2007 [17] .

It is clear that the development of A.I. and its application to the 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 potential for challenging game environments and opponents.

Risk has been the subject of academic work before, with designs for A.I players being a particular focus. In his PhD thesis [20], Michael Wolf of the University of Darmstadt proved that the state space for Risk is infinite. Other research on Risk includes work by students at Stanford University on A.I. players [12]. They summarise Risk as an interesting opportunity for research, sitting at the "intersection between traditional and modern board games".

1.3 Aims

- To design and implement a plan recognition agent for the board game Risk using the PHATT algorithm.
- For the plan recognition agent to have an average plan prediction accuracy at least twice that of a uniform guess.
- To gain further understanding in the complexities of performing plan recognition in Risk
- To determine whether plan recognition algorithms are beneficial in board games, using Risk as an example.

1.4 Hypotheses

The hypothesis for the project is that *plan recognition algorithms are beneficial in the board game Risk.*

Beneficial is defined as a combination of two factors. That the plan prediction accuracy is high and the convergence to the correct explanation is fast.

1.5 Paper Structure

Chapter 2 presents a background to the project where the process of plan recognition and the board game Risk are introduced. Reasons for using plan recognition

in board games are discussed.

Chapter 3 documents the design of the plan recognition agent. It includes an introduction to the PHATT algorithm, along with a worked example.

Chapter 4 details the implementation of the plan recognition agent. It includes the high level architecture of the code, as well as modifications to the open source project and relevant design concepts.

Chapter 5 presents an evaluation of the plan recognition agent, including experiments using the agent. Experimental findings are presented here.

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

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 [18] 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* [5].

They defined each as follows:

- *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* [13] which set the framework of many plan recognition projects that followed. They 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 [4] 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 Bayesian models. Their research continues to be popular in many avenues of research, including its application to games.

For instance, in 1998, Albrecht, Zukerman and Nicholson [6] researched keyhole plan recognition using dynamic Bayesian networks to represent features of an adventure game. The results "showed promise" for various domains. Five years later Fagan and Cunningham utilised case-based plan recognition in the classic game Space Invaders [14], producing "good prediction accuracies in real time".

More recently though Synnaeve and Bessiere [19] published a paper on the design and implementation of a plan recognition agent in the Real-Time-Strategy game StarCraft. The result of their work was a plan recognition agent capable of

predicting the types of units a player intended to produce, through observations of the buildings that they constructed.

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 in a non-adversarial environment.

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

A wants to surprise B, and so will not tell B what his root-goal is. B wants to know what to expect for lunch. Since B cannot know what A is thinking, B must infer A's root-goal through *observing* A cook. In this way, B can consider A's behavioural processes as a Hidden Markov Model. Person B thus models A's behaviour in the following manner.

Person B assumes that A is rational, and that A has a *plan* to achieve their root-goal. A's root-goal can be decomposed into a number of *sub-goals*, such as *Cooked-Beef-Patties*. Sub-goals can often be decomposed further 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 the following assumptions:

- A believes that they can cook a meal, i.e that their goal is achievable.
- 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 certainty what A will cook.
- A cannot hide any actions.
- 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.
- 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.

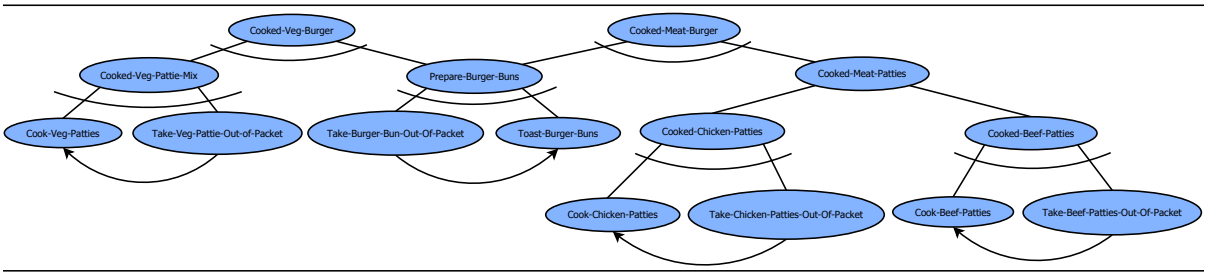


Figure 2.1: Root-goals are nodes that have no parent nodes. Sub-goals are nodes that have both a parent and children and actions are nodes that have no children. And-nodes are represented by an undirected arc across the lines connecting the parent node to its children. Or-nodes do not have this arc.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

RISK is a turn-based board game for two to six players. There are many versions, the first of which was released by french film director Albert Lamorisse in 1957 under the title "La Conquête du Monde". This paper is only concerned with the standard version of Risk, which is an adversarial environment where players vie for control of a fully observable board representing Earth.

2.2.1 Equipment

Figure 2.2: Risk Equipment [10]

The game consists of three pieces of equipment:

- A board portraying a geographical map of the Earth.
- Different coloured tokens called *armies*.
- Two fair 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 grouped into six *continents*, usually corresponding to their actual geographic grouping.

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

Each playerâĂŹs victory conditions depend on the *Game mode*. Game modes significantly impact the behaviour of players and are decided before the game starts. In standard Risk, the possible game modes 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 only describe the rules for that game mode.

After the initial setup, each player's turn is split into three distinct phases which always occur in the same 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 between the players with a random allocation.
- Players being given a number of starting armies. This number is defined in the Risk rule book. Roughly speaking, it is inversely proportional to the number of players.

- 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 up to 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.. The player with the lower value at each comparison loses a single army; 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 the requisite armies.

An attack of a territory has three possible 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 the player chooses to end the attacking phase, or they are no longer able to attack, 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 behind. 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 playerâĂŹs turn ends and another playerâĂŹs reinforcement phase begins.

2.2.4 Cards

Each territory has a corresponding card. As well as displaying the name of that territory, each card either depicts an infantry, cavalry or artillery symbol. By successfully occupying another player's territory in a turn, a player is awarded a single card for that turn. There are also two wild cards which can be assigned a symbol of the playerâĂŹs 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. This trade can be done at any point during the playerâĂŹs turn.

2.3 Why Plan Recognition in Board Games

By "knowing a users plan and goals can significantly improve the effectiveness of an interactive system" [3], 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.

"Knowing a userâĂŹs plan and goals can significantly improve the effectiveness of an interactive system" [3]. The successes seen in the application of plan recognition to video games, many of which have highly complex environments, indicates that similar successes can be had with board games.

Molineaux, Aha and Sukthankar argue that "plan recognition methods can be a powerful ally for machine learning techniques" [15]. Indeed, this was demonstrated by Synnaeve and BessereâĂŹs work with StartCraft [19]. Their plan recognition agent utilised unsupervised machine learning of common plans, which could be used as input for adaptive A.I. Using similar techniques for augmenting board game A.I. is an attractive proposition.

2.4 Summary

Plan recognition is a mature area of research dating back to the 1950's. It has seen applications in a variety of fields including video games. Risk a popular board game that provides an interesting environment to test the application of plan recognition algorithms. It is a multi-agent adversarial environment at the intersection between modern and classical board games. Plan recognition techniques have been used before to augment video game A.I.; there are similar opportunities for applying them to board games. Machine learning can be used to create

more interesting A.I. opponents.

3. Design

3.1 Introduction to PHATT

PHATT is a plan recognition algorithm by Christopher Geib and Robert Goldman [8]. PHATT operates on plans that are executed dynamically, but in discrete time steps. The set of actions that an agent can take each step, their *pending set*, depends on the actions that the agent has previously taken. The model is as follows. An agent first chooses 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, which results in some actions being appended to the pending set and others being removed, thus forming the next pending set. Assuming *blind commitment* on the part of the agent, the agent would then continue to perform actions until its root goal has been satisfied.

From this model, Geib and Goldman proposed an algorithm utilizing a Bayesian approach to perform probabilistic plan recognition.

The algorithm computes *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 BayesâĂŹ Rule they defined *Pr(exp|obs)* as:

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

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

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

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

3.1.1 Computing an Explanation's Probability

Computing the term *Pr(exp ∧ obs)* requires the plan library to be augmented with three probabilistic features:

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

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

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

3.2 Environment Modelling and Data-Structure Design

PHATT is designed to work in an action space. It was therefore it was a vital to translate Risk into such an action space.

Solving this issue involved modelling and data structure design. By exploiting the environmental constraints on what a player can do, and observing the choices they made given these constraints, we attempt infer the plan of a player.

3.2.1 Root-Goals

The mission cards of the Risk 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.

Since mission cards are handed out at random, the prior probability of a root-goal is $1/n$, where n is the number of mission cards. The data structure of a root-goal

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

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

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

A subset of these, where the root-goal involved occupying continents only, were chosen to be the focus of this paper. This choice was made due to time constraints and is an 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, the assumption is made that when 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.

This has another implication. Since PHATT computes *Pr(exp ∧ 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 playerâĂŹs behaviour, and the highest probability by the end of the game is the plan recognition agentâĂŹs prediction.

3.2.2 Actions

In Risk 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 an explanation 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 playerâĂŹs plan.

The pending set of a player's attack actions for any given turn is the aggregate of either a successful or unsuccessful attack on each territory that the game rules will allow them to attack.

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 playerâĂŹs 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. 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 the 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. An attack is required before a defence can occur. In this way, the defence action is modelled as either consistent or inconsistent with only the explanations it is directly related to.

Consider 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 a side-effect which is the increase of other explanations. In PHATT, each explanationâĂŹs probability must be 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 compared to explanations that were not multiplied by x. Since big changes in explanation probabilities are undesirable for defence actions, this effect was minimized by constraining x by $0.9 < x < 1.1$.

As with attacks, a defence 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 missions that require that territory.

Movement actions are modelled as a less significant indicator of a playerâĂŹs plan. This is due to the high proportion of reinforce and attack actions that are already modelled.

3.2.2.4 Reinforce

The reinforce actions that a player p can perform at any turn t is determined by the territories that p owns during turn t. The pending set of any playerâĂŹs reinforce actions is therefore modelled as follows.

For each territory T that p owns at turn t, in p's pending set is an action to reinforce T. If a territory T_l is lost by p to another player, its corresponding reinforce action is removed from the playerâĂŹs pending set for their 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 playerâĂŹs pending set at turn t.

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

Since reinforce actions are the most frequently observed action in Risk, modelling it as an action could have significant impact on the plan predictions. That said, it is clearly an important indicator of a playerâĂŹs plan. The crucial decision is when a player takes the risk of attacking; not where to place ones reinforcements. Therefore, the reinforce action will be modelled as a less significant indicator.

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

Algorithm 3.3.1: GENERATEEXPLIST(−)

```
forall the C ∈ CS do
    forall the E ∈ ES do
        if C isSubGoalOf E then
            forall the T ∈ C do
                addAction(ReinforceT) to ECA
                addAction(MovementT) to ECA
                addAction(Successful-Defence) to ECA
                addActionSuccessful-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 PR 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 [9] which proposes weighting action probabilities towards consistent

actions, a technique of doing so in the Risk. 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

The concepts from this chapter can be tied together with an arbitrary example. In this example we compute *Pr(exp|obs)* the likelihood of an explanation given an attack observation. Computing *Pr(exp|obs)* with other action types will also be discussed.

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 Risk map consists of only the territories, continents and root-goals illustrated on Figure 3.1.

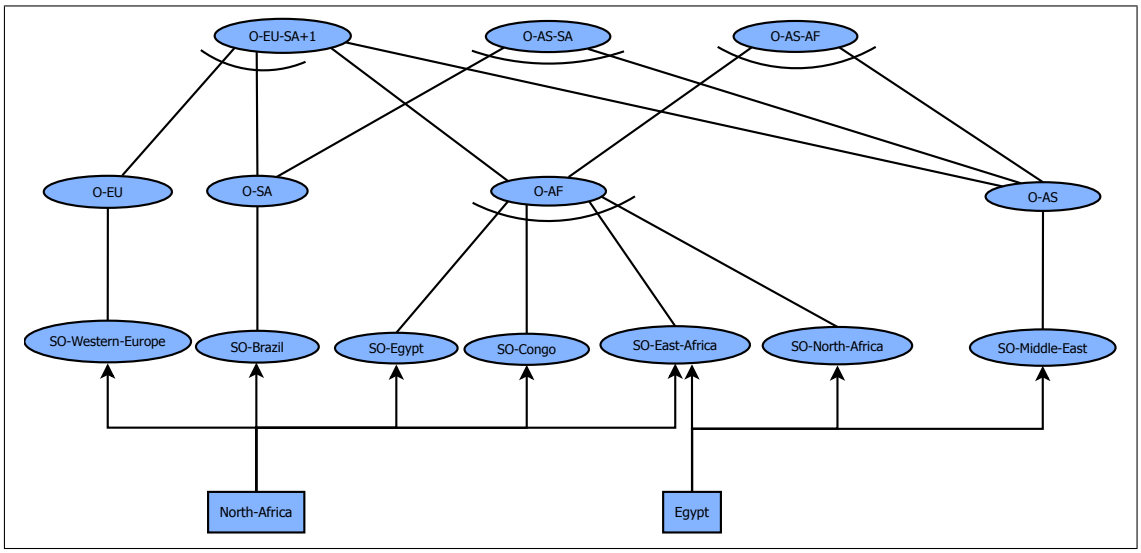


Figure 3.1: Successful-Occupation Plan Library for North-Africa & Egypt, **O** = Occupy, **SO** = Successful-Occupation

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 Risk, 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 PR 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 PR 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 PR 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 = {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 PR 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 PR agent then computes an explanation that the players plan is the root goal given the observation $Pr(exp \wedge obs)$ 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 $Pr(exp \wedge obs)$ for the explanation O-EU-SA-AF the PR 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 PR 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 PR 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 PR 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 PR 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 PR 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

Environmental modelling and a design of environment data structures was necessary in order to allow the PHATT algorithm to operate in the Risk environment.

Each action in the Risk environment was modelled and various assumptions about each were made in-order to compute the probability of a player. 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 PR agent.

A key concept in the design of the agent, is that the pending set of a player is decided *a priori* by the territories that a player owns. And by combining the actions that are 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 PR agent. The reasons for doing so were:

- The availability of an open-source project with an active development team.
- Personal familiarly with the language.
- Java being cross-platform.

4.1 System Architecture

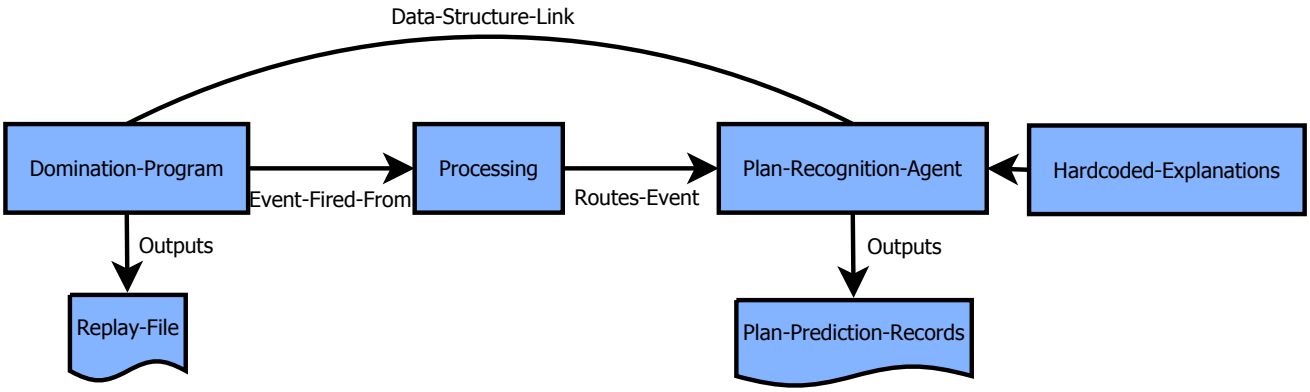


Figure 4.1: System Architecture

The system followed an *event-driven* architecture, comprising of three major components.

The *Domination-Program* which refers to the open-source game developed by Yura.Net. Modifications to the source code of the program were:

- Automating the output of the *Replay-File*.
- Code to fire events into the *Processing* component.

The *Processing* component was responsible for routing events to the *Plan-Recognition-Agent*.

The *Plan-Recognition-Agent* as well as being responsible for had access to the data structures in the Domination program. Rather than storing its own copy

avoided any synchronisation between it and the Domination program. Additionally the advantages of this approach includes less overhead as well significantly reducing the possibility that the PR agents copy loses synchronisation with the Domination-Programs data structures. On the other hand a disadvantage is that this makes the plan-recognition-agent more dependant on the methods of the developers implementation.

The PR agent made extensive use of the freely available Google Guava libraries. These allowed the prediction agent to operate concurrently with the game as well as more efficiently in its operations.

A set of explanation objects made up the *Hard-coded-Explanations*. Explanation objects had hard coded root and sub goals. Using these sub-goals, the system could build a list of consistent and inconsistent actions using the data structures stored by the Domination program.

The *Hard-coded-Explanations* were implemented as separate to the core structure of the PR agent. As given a complete list of hard-coded explanations for another map the plan recognition can be used successfully once again.

The Domination-Program is capable of outputting a Replay-File that could be used to replay a game.

The PR agent would output a csv file containing the plan predictions for each player over the course of the game.

The architecture of the system is such that the PR 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 PR agent.

4.2 Plan Recognition Agent Architecture

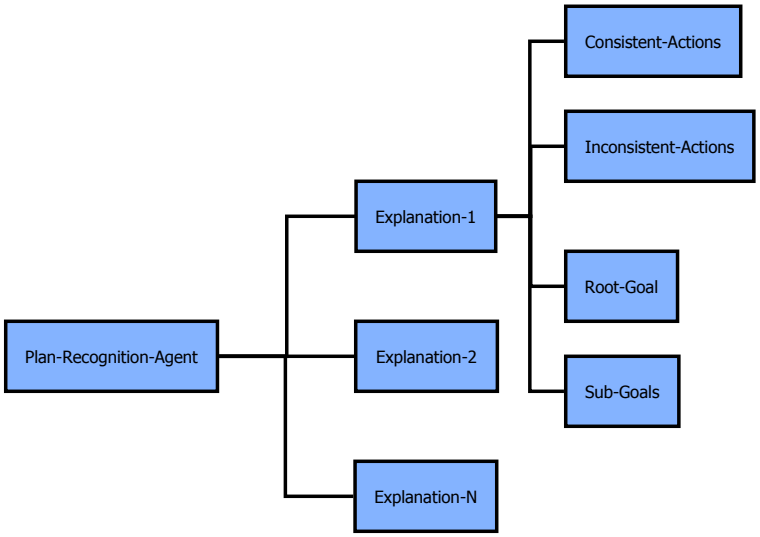


Figure 4.2: PR Agent Architecture

The Plan-Recognition-Agent has a data structure for each player, which in turn has several explanation data structures that are initially hard-coded. Each explanation object calculates its own probability.

The Root-Goal and Sub-Goal data structures must be hard-coded

When an event arrives from the processing class it is routed to its specified player data structure and then to its specified explanation data structure which would use the observed action to compute its new probability.

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

4.3 Event Design

To operate the PR 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 these events occur a distinct event object is fired, these even objects contains important information about the event that is necessary to the PR agent. These events are observed by the processing class which then routes them to the PR agent. The PR 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:

More specifically other than player initialisation or removal events, each event will trigger the computation of explanation probabilities, which players explanation probabilities are decided for example by the *attacker-name* and what effect, is decided by the state of the Risk world namely the respective players pending set which can be built from the data structures of the Domination-Program and the continent and territory name of the event.

4.4 System Operation

Before a game begins, based on the players registered in the game lobby screen the PR agent initialises a set of player data structures.

The PR agent first initialises the list of *Hardcoded-Explanations* which involves building the set of consistent and inconsistent actions based on the explanations sub-goals. These explanation objects contain all the necessary information from

Event Type	Contains Variables	Purpose
New-Player	player-name	Initialises a new player data structure with ID *player-name*.
Remove-Player	player-name	Destroys the player data structure with ID *player-name*.
Successful-Occupation	attacker-name, cName, tName	Signals that the player named *attacker-name* performed a Successful-Occupation.
Failed-Occupation	attacker-name, cName, tName	Signals that *attacker-name* performed a Failed-Occupation.
Successful-Defence	defender-name, cName, tName	Signals that *defender-name* performed a Successful-Defence.
Failed-Defence	defender-nName, cName, tName	Signals that *defender-name* performed a Failed-Defence.
Army-Movement	player-name, cName, tName	Signals that *player-name* performed an Army-Movement.
Reinforce	player-name, cName, tName	Signals that *player-name* performed a Reinforce action.

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

the environment to computes its own probability. Once the complete list of explanations has been fully initialised it is cloned into each players data structure.

Once action events start firing from the Domination-Program and was routed by the Processing component, the Plan-Recognition-Agent will depending on the event compute new explanation probabilities for a player.

This would continue till the end of a game the domination program as well as the PR agent would output two files allowing the replay of a game and a record of the PR agents prediction probabilities for each players explanations over the course of the game.

4.5 Summary

The Plan-Recognition-Agent was written in the Java programming language following an event-driven system architecture. It was built to be able to operate on any map developed for the Domination-Program and using the Google Guava libraries could operate in parallel to the Domination program.

By receiving events fired from the program it would update locally held player data structures that contained information about each players explanation predictions and any other required would be retrieved from the Domination-Program.

At the end of a game the Plan-Recognition-Agent would output two files. One was a Replay-File that would allow a complete replay of the game, the other was the Plan-Prediction-Record which detailed the probabilities of each players explanations over the course of the game.

5. Evaluation

The evaluation plan consists of four experiments on the Plan Recognition agent:

1. Free Play
2. Constrained Play
3. A.I Play - 100
4. A.I Play - 4

5.1 Experimental Format

5.1.1 Participant Experiments

Of the four experiments, two were conducted with human players:

Free Play, where human players "play to win", in any manner they see fit. This is how people would normally play mission Risk.

Constrained Play consists of players only performing actions that are *directly-consistent* with their root-goal.

An example of a directly-consistent action is occupying a territory required by a root-goal. An example of the converse, a *indirectly-consistent* action, would be breaking another player's continent by occupying a territory in that continent. It is indirectly-consistent because whilst not satisfying the player's goal, it prevents other player's from satisfying theirs. And only one player can win. More formally, given a set of actions (excluding card related 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 artificial form of play is designed to give optimal performance from the plan recognition agent. This is achieved by minimizing environmental noise from indirectly-consistent and inconsistent actions found in the other forms of play. Under this form of play, a high prediction accuracy is expected. If this is not the case then the resulting experiments should provide a good indication as to why.

The A.I. players used were those provided with the Domination implementation of Risk. For experiments involving human players, the game rules were explained before the start of the experiment.

For constrained 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 the game:

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 they to it during the course of the game.

In total, nineteen people participated in the experiments. Some took part multiple times.

The final count of participant data samples was:

- 75 data samples collected for free play.
- 100 samples collected for constrained play.

5.1.2 A.I. Experiments

A.I. Play are games played using only the game A.I provided with the Domination Risk implementation. These A.I. players can be substituted for human players to the point where a game can consists of only A.I. players battling each other. A.I. games can finish in a fraction of the time that it takes to complete a game with human players.

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."

Looking at the Domination program, the degree to which the A.I.'s decisions are skewed, is dependant on a variable w. By increasing this value, it causes the A.I.

to more actively pursue its assigned mission card.

Using this we can evaluate the plan prediction accuracy of the PR agent, and perhaps more importantly see whether the PR agent is capable of detecting the behavioural change that follows a change in w.

Automating the collection of these data samples using the free macro program AutoHotkey, provided an opportunity for the collection of a large number of data samples for analysis.

The final count of A.I. data samples was:

- 1004 A.I. data samples set at w 4.
- 1004 A.I. data samples set at w 100.

5.2 Data Collection

After a game had finished the Domination program would generate a replay file. This file can be loaded later through a specialised interface, allowing an exact replay of the game.

Figure 5.1: Domination Debug Graphical User Interface

Using this tool made it possible to re-observe games for their evaluation. It also allows the PR agent to be tested with different modelling and calibrations in the same game twice. This eliminates environmental variables between comparisons of the PR agent.

In addition to the Domination replay file, the PR agent would also generate a CSV file containing the significant features of a game, namely:

- Which players took part
- Which player won
- Each player's mission card
- The probabilities of each player's explanations over the course of the game

Scripts written in Python were used to automate the analysis of the generated CSV files.

5.3 Experimental Findings

Experiments with the PR agent revealed the following insights:

1. The prediction accuracy of the PR agent is highest in constrained play.
2. The prediction accuracy of the PR agent is higher for winning players than losing players.
3. The prediction accuracy of the PR agent is higher for human players than the Domination A.I. players.
4. The prediction accuracy of the PR agent reflects the failings of the Domination A.I. in completing its mission.
5. There is a positive correlation between game length and prediction accuracy.
6. This is most true of winning players.
7. Constrained play games were never longer than 120 turns, making them the shortest of the four game types.

Keeping in mind the initial hypotheses, we attempt to evaluate the claim that plan recognition algorithms are beneficial in Risk.

To do this, we explore the two questions that define what we consider beneficial:

- Is the PR agent accurate in its predictions?
- How quickly does the PR agent converge on a correct prediction?

The term *General* player is used to refer to both human and A.I players. Players are further decomposed into players who won, *Winners*, and players who lost, *Losers*.

5.3.1 Format of Explanation Convergence

At the start of a game, forty two turns are required to place one army on each territory of the Risk map. These observations are not taken into account by the PR agent, as the distribution of territories between players is automatic and random.

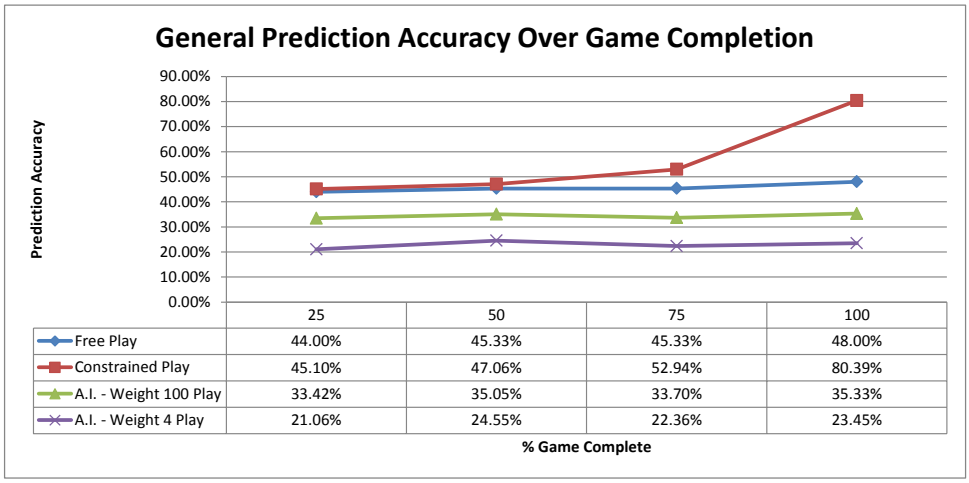
After this, the remaining initial armies are distributed by players onto their territories. During this, the probability of explanations that are consistent with a player's actions will slowly increase. Conversely, those explanations with which the actions are inconsistent with will slowly decrease.

Once the initial setup is complete (which ranges from 80 to 120 turns, depending on the number of players), the attack phase begins 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 in explanation probability occurs when there are many Successful-Attack actions in a single turn. This is due to the design of the PR agent. After p_c's turn is over, other players begin their attack phases in turn. Unless the action another player performs is to attack a territory owned by p_c, the probabilities of p_c explanations remained unaffected.

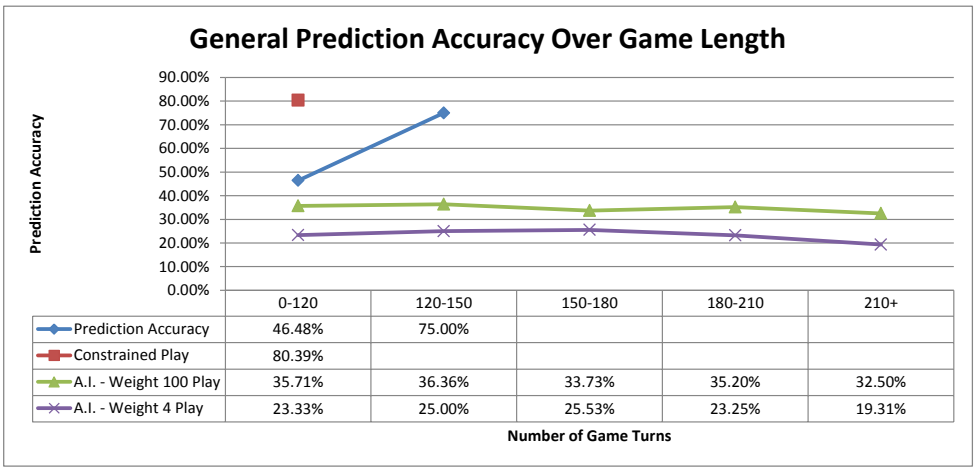
This pattern continues till the end of the game, at which point the PR agent will have determined its predictions. The level of convergence differs between types of play, although generally significant convergence only occurs after the attack phases have begun.

5.3.2 Prediction Accuracy in General



	25	50	75	100
Free Play	44.00%	45.33%	45.33%	48.00%
Constrained Play	45.10%	47.06%	52.94%	80.39%
A.I. - Weight 100 Play	33.42%	35.05%	33.70%	35.33%
A.I. - Weight 4 Play	21.06%	24.55%	22.36%	23.45%

Figure 5.2: General Prediction Accuracy



	0-120	120-150	150-180	180-210	210+
Prediction Accuracy	46.48%	75.00%			
Constrained Play	80.39%				
A.I. - Weight 100 Play	35.71%	36.36%	33.73%	35.20%	32.50%
A.I. - Weight 4 Play	23.33%	25.00%	25.53%	23.25%	19.31%

Figure 5.3: The general prediction accuracy of the PR agent at various game lengths.

The data has several trends with regards to player prediction accuracy.

The PR agent plan prediction accuracy was significantly better in *Constrained play than in Free or A.I. play* in both interval of game completion, and game length metrics. This is unsurprising given its artificial nature. Designed to be ideal the scenario for the PR agent, it is not a good measure of evaluating the success of the PR agent in general. Instead, in conjunction with Free Play, it gives us a better picture of how well the PR agent performs on human players.

The PR agent's plan prediction accuracy was on average better for human players than A.I. players. The reason for this is the degree with which human and A.I. players actively follow their mission card. The data suggests that, since the core A.I. is a modification to the domination mode A.I., it is not specifically built to actively follow a plan. Thus we can confirm that human players followed their mission cards more actively than A.I. players.

Comparing the prediction accuracy at $w4$ and $w100$, we see up to a 10% improvement in the plan prediction accuracy of the PR agent. This confirms that as the A.I.'s decisions become more skewed towards occupying mission continents and less towards other activities, prediction accuracy increases.

There is a 25% increase in the plan prediction accuracy of the PR agent for constrained play in the 75% to 100% game completion category. In general there is a rise in the prediction accuracy in this game completion category across all forms of play, though the most significant rise remains exclusive to constrained play. Again, this is likely due to the nature of constrained play. As we shall see with further analysis of winners and losers, both categories of constrained play yield high prediction accuracy.

A trend can be seen of an increase in plan prediction accuracy towards the end of a game. As before, this is primarily due to how winners perform a large number of consistent actions in a short span of time at the end of a game.

We can look at an example of constrained play to confirm this.

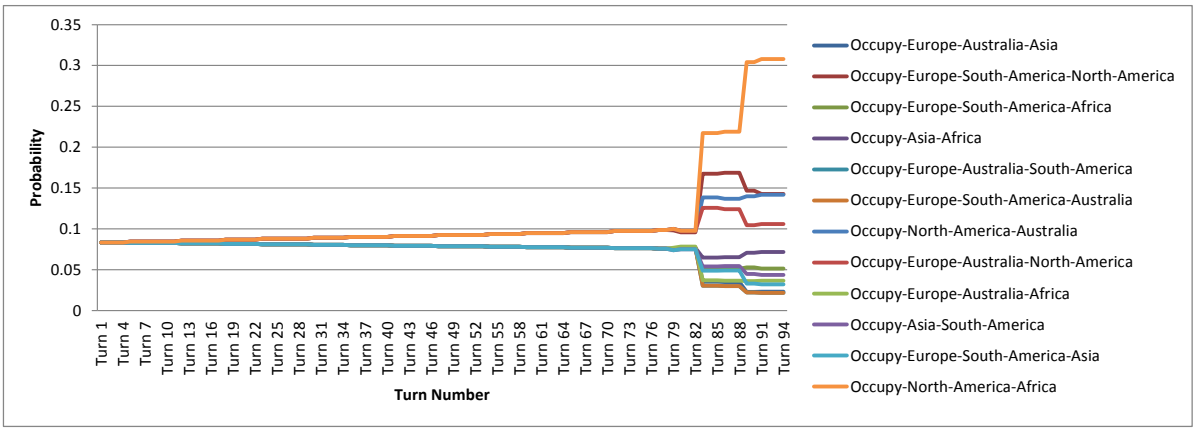


Figure 5.4: A.I. 4 - Example Game

The winning players mission was to capture North-America and Africa. For the majority of the game - during the placement phase and up until turn 82 - the probability does not rise significantly. After turn 82, when the attack phases begin, the increase in explanation probability is substantial. Convergence to

the correct explanation occurs within 10 turns, reaching a significantly higher explanation probability soon after attacks have begun.

The general prediction accuracy over game length falls for A.I. players over time, whereas for human players it increases. Figure 5.2 does not echo this trend though which indicates that prediction accuracy over game length appear to be stable which is indicative that the number of samples from Figure 5.3 reaching game length of over 210+ are relatively small as to not affect the average accuracy so significantly.

The of length the human games ranged between 120 and 150 turns, which is far lower than that of A.I. games. This suggests that either:

- The A.I. is good at preventing other A.I. players from winning.
- Games are longer because the A.I. is ineffective at finishing their mission.

With an example we can see evidence to suggest the latter point.

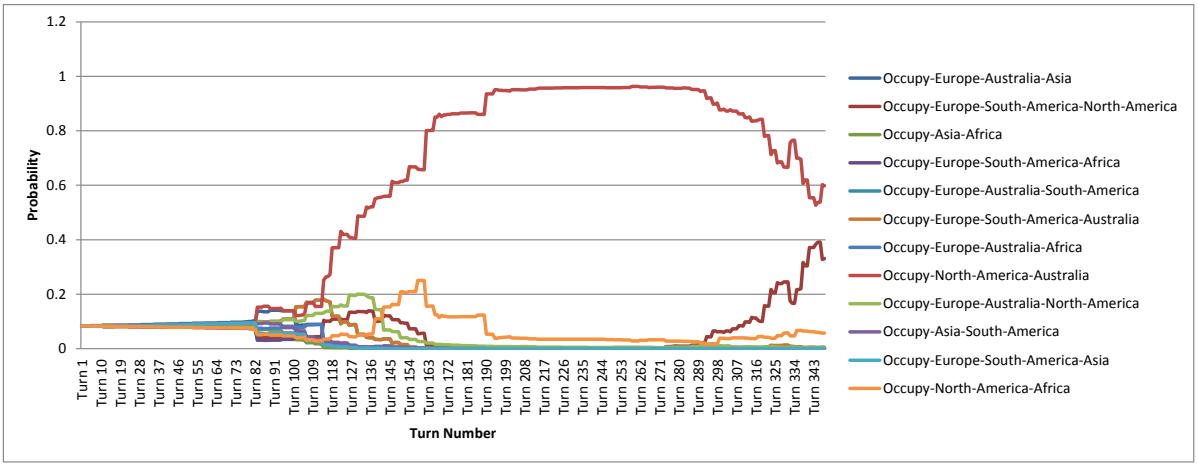


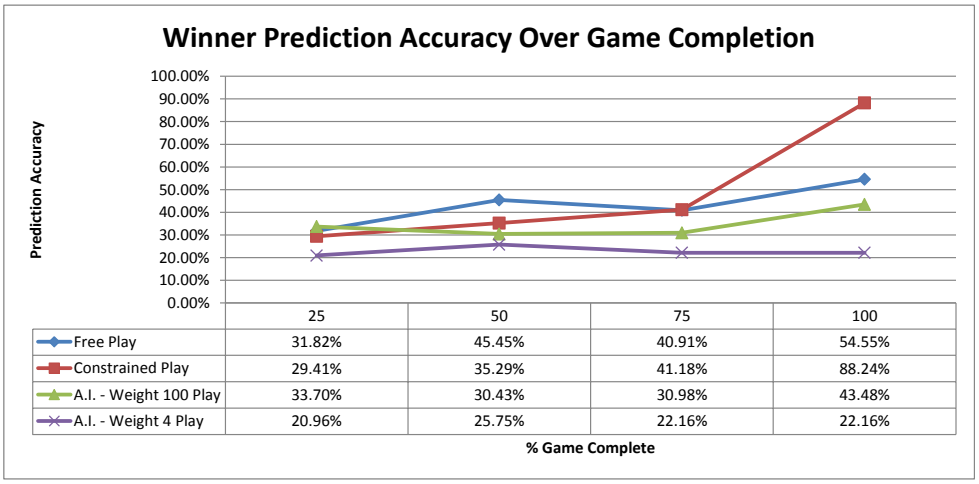
Figure 5.5: A.I. 4 - Example Game

Figure 5.5 illustrates the probabilities of explanations for a 4 player game between A.I.s. Player 4, who has the mission of conquering Asia and South-America, won in approximately 342 turns. The A.I. players were caught in a stalemate between each other for many turns, arguably not out of skill but by a lack of strategic diversity.

For instance, they make no attempt to fortify and hold continents when they were weak, or to consolidate their armies in strategic territories. Instead, the players opted for a strategy of all-out attack. Most of player 4's engagements occurred in North-America and Australia - continents outside of its mission. At one point, the player had a clear advantage but chose to attack a non-mission continent to eliminate a weaker player instead of taking Asia, its final mission continent.

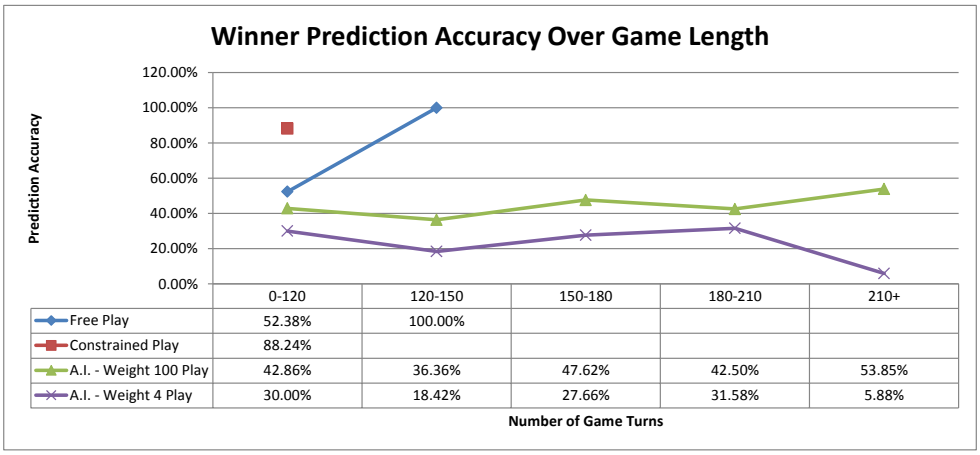
This observation coincides with the significant drop of the PR agents prediction accuracy for A.I. - 4 games longer than 210 turns. The number of such examples must be relatively small because we do not see a significant drop in prediction accuracy of A.I -4 games between 75 to 100 % game completion.

5.4 Plan Recognition Agent Winner Prediction Accuracy



	25	50	75	100
Free Play	31.82%	45.45%	40.91%	54.55%
Constrained Play	29.41%	35.29%	41.18%	88.24%
A.I. - Weight 100 Play	33.70%	30.43%	30.98%	43.48%
A.I. - Weight 4 Play	20.96%	25.75%	22.16%	22.16%

Figure 5.6: Winner Plan Prediction Accuracy



	0-120	120-150	150-180	180-210	210+
Free Play	52.38%	100.00%			
Constrained Play	88.24%				
A.I. - Weight 100 Play	42.86%	36.36%	47.62%	42.50%	53.85%
A.I. - Weight 4 Play	30.00%	18.42%	27.66%	31.58%	5.88%

Figure 5.7: Winner Plan Prediction Accuracy

First, we must point out that the prediction accuracy of the PR agent is 1.0 only due to the number of data samples of winners being small. There was only a single data sample of a winner between 120-150 turn long game, and that sample was correctly predicted.

Games of constrained play were never longer than 120 turns.

The highest plan prediction accuracy of the plan recognition is 88.24%, which is for winners in constrained play games of length 0-120.

What is striking about the data for winners, is that for A.I. - 4 games, there is a significant drop in plan prediction accuracy from 31.58% to 5.88% in from the 180-210 to the 210+ game length category.

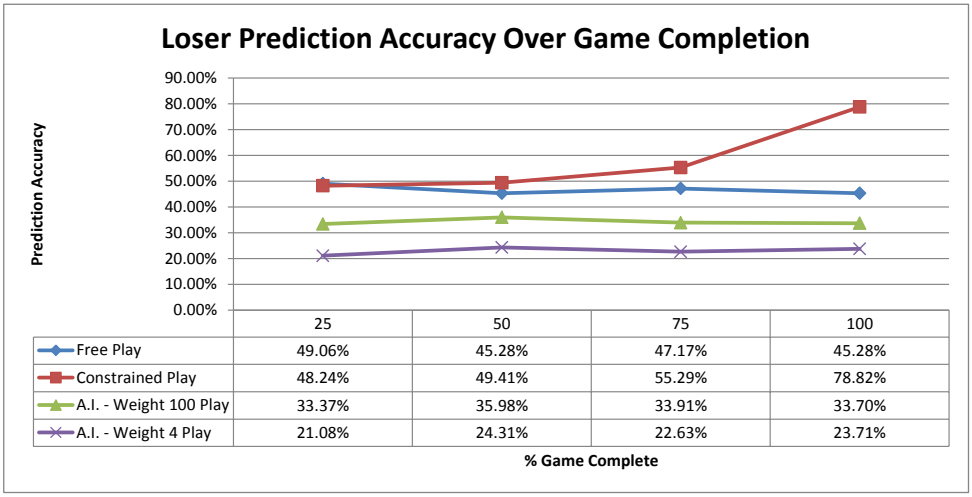
As we saw from the example of a long A.I. game in the previous section, this can be explained by the majority of the losers having been eliminated at that stage and the winner was focused on eliminating the last player. This involved performing many inconsistent actions to eliminate them, and the losers that did survive did not perform many actions because of a comparatively low level of reinforcement armies. Once again, though the number of these cases are small given the winnerâĂŹs plan prediction accuracy remains stable on average through a game.

For A.I-100, there is a positive trend that as the game length increases, so does the plan prediction accuracy.

The trend of increase towards the end of the game is much more apparent for winners, as seen by significant increases of plan prediction accuracy between 75% and 100% game completion. This is due to winners performing more directly-consistent actions at the end of a game.

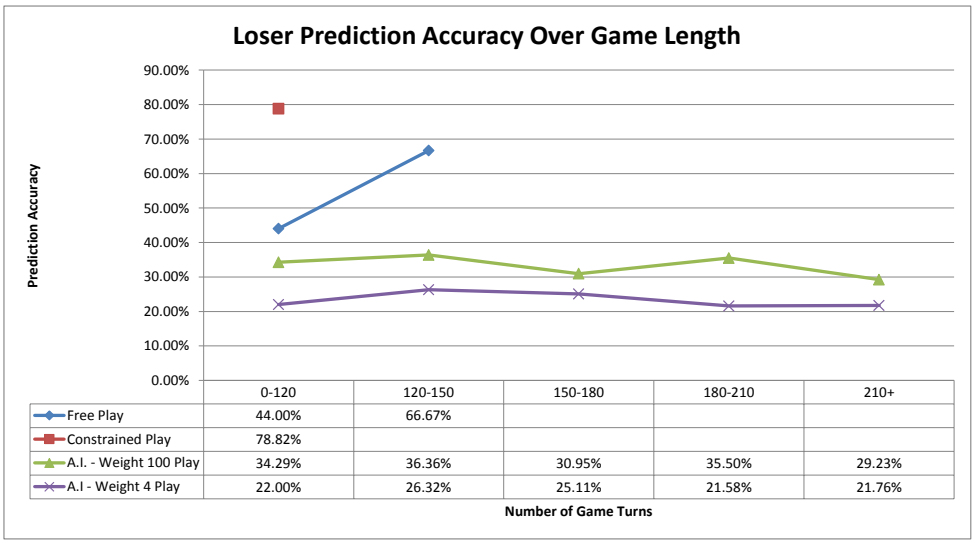
There are cases where winners have occupied an entire continent and completed their mission in one turn. Since sampling is done on a turn by turn basis, the PR agent will miss the significant change in probabilities. This is a consequence of the system not registering an end of turn, but rather a different event which is the end of the game.

5.5 Plan Recognition Agent Loser Prediction Accuracy



	25	50	75	100
Free Play	49.06%	45.28%	47.17%	45.28%
Constrained Play	48.24%	49.41%	55.29%	78.82%
A.I. - Weight 100 Play	33.37%	35.98%	33.91%	33.70%
A.I. - Weight 4 Play	21.08%	24.31%	22.63%	23.71%

Figure 5.8: Loser Plan Prediction Accuracy



	0-120	120-150	150-180	180-210	210+
Free Play	44.00%	66.67%			
Constrained Play	78.82%				
A.I. - Weight 100 Play	34.29%	36.36%	30.95%	35.50%	29.23%
A.I - Weight 4 Play	22.00%	26.32%	25.11%	21.58%	21.76%

Figure 5.9: Winner Prediction Accuracy

Losers in A.I - 100 appear to show negative trend in that as the game length increases the prediction accuracy falls. As we saw from an analysis of winners the inverse to this was true, as game length increases the plan prediction accuracy increases as well. This is an interesting correlation as it suggests that the prediction accuracy of winners at least for A.I. - 100 play gets better as losers get worse. This makes sense; by definition, a loser cannot complete their mission.

Losers are often prevented from successfully achieving their goal as they may be eliminated early or may be pushed out of their mission continents. The following is an example of a player with the mission of taking North-America and Africa being eliminated from a game at approximately turn 75.

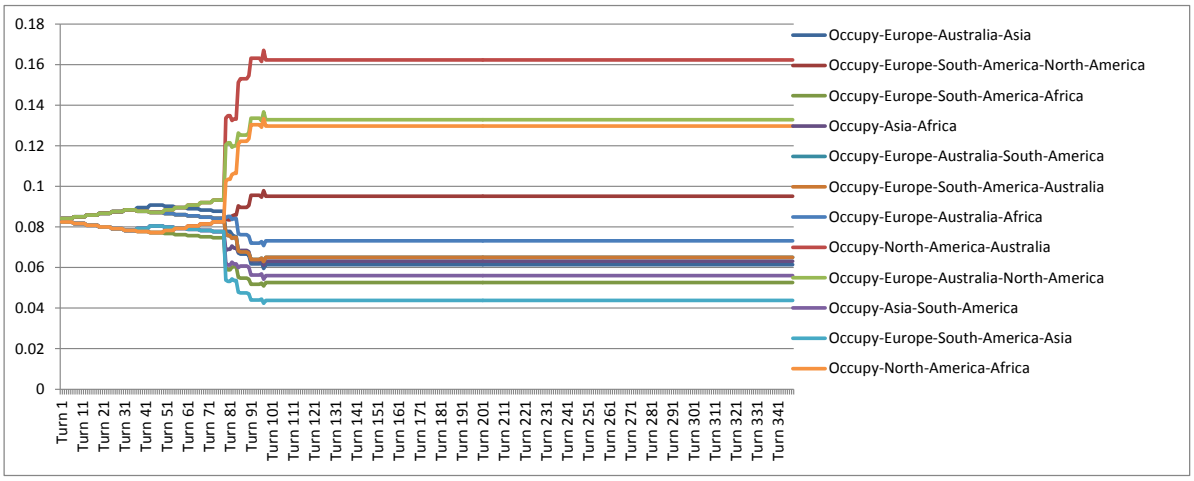


Figure 5.10: Example Player Eliminated

A number of incorrect predictions occurred due to the lack of data about players that are eliminated early on. If a loser is eliminated early in the game, then their explanations probabilities would *flat-line*. This can be seen clearly in Figure 5.10. If the loser had been performing more inconsistent actions than consistent actions up to that point, then the prediction will be incorrect and will remain so for the rest of the game. Conversely, it may remain correct for the same reason.

5.6 Outcomes

The PR agent has shown to perform better than uniform choice of mission card, but the prediction accuracy is still below 50% and significant successful convergence does not occur typically until after 75% of the game is complete. This does not meet the projects definitions of beneficial. In defence to this, the PR agent did perform better on players than on A.I. which for our purposes is significant.

The data that we have collect supports the theory that unless players behaved as in constrained play, the intense adversarial atmosphere, together with the high level of overlap between mission cards causes significant noise for the PR agent, which is detrimental to converging on the correct explanation but rather results in on average high levels of incorrect plan prediction's.

Particularly difficult areas of this project have been, deciding on methods of computing action probabilities and how much if any to weight each of those actions.

There was trouble working with an algorithm that was made to deal with an

action space rather than such a significant state based environment. Therefore having to figure out how to represent the environment as an action space so that the algorithm could operate was particularly difficult.

Finding plan recognition algorithms that are made to cope with such noisy environments may be beneficial in future research. Also given that it works well in constrained play, the design has some in less complex environments such as the card game Pit.

Constrained play games were never longer than 120 turns, thus making them the shortest of all game types. In constrained play, players do not purposely seek to hinder the progress of other players' missions. So therefore winner of constrained play tends to be the player with the least overlapping missions 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 due to the design of constrained play. Hence games are shorter.

For A.I. 100 there is a inverse correlation between the plan prediction accuracy of winners and the plan prediction accuracy of losers.

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 at least 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. This proposition is supported by how the PR agents *prediction accuracy of winners is better than that of losers.*

In Risk.there is only one winner. Risk 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.

Given the design of the PR 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.

It is unfortunate that the project has not proved the original hypothesis though how indeed showed som success. And we can conclude, that Risk is certainly a challenging environment to perform plan recognition for.

5.7 Future Work

A platform has been setup for further improvements of the PR agent. Avenues of future work include exploring:

If the plan prediction accuracy of winners is better than loser is in fact may be due to because weighting of attack actions is towards successful attacks, of which winners perform more. This question can be explored.

The effects on accuracy when experimenting with A.I. play was done with six players. This would be interesting to see the effects on prediction accuracy differs depending on the numbers of players in the game.

Data has been collected for explanation accuracy measure but has not been analysed. Doing so could shed some light onto whether the accuracy of mission cards are different and why.

Using the collected data samples, test the PR agent with different models of the environment or different calibrations.

The model could be extended to include other mission cards.

5.7.1 Improvement of Action Probability Computation

Currently the system suffers from the problem of distinguishing noise from meaningful actions. Solving this problem could be tackled in two ways:

- Identifying actions are indirectly-consistent and modelling them to accurately contribute to explanations - further modelling to further differentiate noise from meaningful 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.

Considering the latter point. By 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.

Currently the weight for actions 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 propose using a dynamic weight system that captures certain aspects of the Risk environment as follows:

Explanation-Prob = Explanation-Prob * Weight * Fraction-Representing-State-of-Environment

More specifically:

Weight = 0.1 * 1- (No of Consistent actions) * (No of territories a player owns of that continent)

Therefore:

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

A player has 1 attack option out of 13 and owns 9 of the ten territories

$$0.1 * 1-(1/13) * 9/10 = 0.083$$

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

A player has 12 attack options out of 13 and owns 4 of the ten territories

$$0.1 * 1-(12/13) * 4/10 = 0.00307$$

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

5.7.2 Use of Machine Learning

According to PHATTs original design, we are not supposed to operate on a full set of explanations. By introducing the term for sub-goal probabilities again we can use machine learning on collected training data of games, to optimize the values for the priors of the sub-goal probabilities based on preferences of what choices are made by players in the training set.

5.7.3 Meta Optimisation

Using plan recognition software, given an intended plan of a programs behaviour, plan recognition algorithms can be used in detecting the programs success in performing that plan. More specifically using genetic algorithms we can perform meta optimisation of programs, tuning the program with the result of the plan recognition algorithm.

5.8 Criticism

5.8.1 Modelling Issues

The only action that has been modelled following PHATT's design is attack. The remaining actions are crude probability manipulations to by scaling explanation probabilities up or down based on design assumptions. No modelling done of indirectly-consistent actions.

In the computation of attack probabilities, treats any non-owned connected territory as attackable. This is not the case as some territories may not have two armies and so cannot attack the neighbour.

The model also makes no attempt to differentiate two identical child-root-goal e.g the plan prediction accuracy is equal between EU SA AU and EU AU SA in many cases.

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.

The 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.

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

5.8.2 Data Issues

A better picture of the PR agents plan prediction accuracy for human players, could have been seen by making the intervals of the game length smaller. Thus expanding constrained play which was a single data point in several graphs.

A comparatively small number of data samples were collected for free and constrained play. Therefore the data is not representative to the same degree as the A.I. data samples were. Additionally data samples were collected with a small group of participants, as the same people play the game several times, it becomes an influencing factor in the results of the experiments.

The sampling of explanation probabilities needs to be more frequent than only at end of each turn as several 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.

5.9 Summary

The PR agent was evaluated using four different experiments. Constrained play gave the highest accuracy, but free play is more representative of and actual game. Analysis of the PR agent's plan prediction accuracy over different game lengths and at various time intervals yielded several valuable insights. Generally speaking, the hypotheses were borne out by the data.

6. Conclusion

This project has presented a plan recognition agent for the board game Risk. The agent is based on the PHATT algorithm [8] and acts as an observer to the game with the goal of inferring the unknown mission cards based on player actions in real time.

After a detailed design specification of the PR agent, and a high level overview of its implementation, we went on to evaluate the PR agent. This was done by means of testing its accuracy and speed of convergence in four different experiments. The results of these experiments were then presented and analysed.

The PR agent has the highest accuracy on constrained play, but performed the worst on A.I. - 4 play. For the most representative test, free play between human players, it was generally a success compared to naive methods.

There is still room for improvement. For instance, by introducing factors that capture the state of the environment into the calculations, or choosing better prior probabilities with machine learning techniques, the prediction accuracy may be improved. These are avenues for further research. At present, the agent may be used to assess existing Risk A.I. players, develop better A.I. players, or act as an assistant to human players. The key point is this: that plan recognition can be applied successfully to multi-agent board games.

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

7.3 Explanation Accuracy Data

		% 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

Bibliography

[1] *Deep Blue Versus Kasparov: The Significance for Artificial Intelligence, Collected Papers from the 1997 AAAI Workshop*. AAAI Press, 1997.

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

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

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

[5] 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.

[6] 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.

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

[8] 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.

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

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

[11] John McCarthy. *What is Artificial Intelligence*. November 2007.

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

[13] 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.

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

[15] Matthew Molineaux, David W. Aha, and Gita Sukthankar. G.: Beating the defense: Using plan recognition to inform learning agents. In *In: Proceedings of Florida Artifical Intelligence Research Society, AAAI*, pages 337–343. Press, 2009.

[16] 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.

[17] Jonathan Schaeffer, Yngvi Björnsson, Neil Burch, Akihiro Kishimoto, Martin Müller, Robert Lake, Paul Lu, and Steve Sutphen. Solving checkers. In *IJCAI*, pages 292–297, 2005.

[18] 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.

[19] 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.

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