Faculty of Technology

User Intention Modelling and Intention Recognition in Games using World State Indicators

by

Matthias F. Brandstetter

(P09170054)

A Thesis submitted in Partial Fulfilment of the
Requirements for the Degree of

Master of Philosophy

July 2016

Abstract

The work presented in this thesis focuses on two main areas. First we develop a model of user intentions in games. That model defines various concepts related to player behaviour in (computer) games and interactive environments, such as *actions*, *goals*, *plans* and *intentions*. Additionally our model shows the relationship between those concepts and explains how they affect each other. The purpose of the model presented here is twofold. One the one hand it provides common definitions for research in the area of user behaviour modelling in games. On the other hand this model also forms the underlying basis for the remainder of our work presented here.

The second main area of focus is *intention recognition* in games. We propose a novel approach which is solely based on monitoring the changes in the game world state, instead of observing player actions. We evaluate current approaches to plan and intention recognition, their strengths and weaknesses. We further compare existing research on intention recognition to our approach and evaluate the performance of our prototype system *iRecognise* in the context of a case study using the board game RISK. A range of experiments that were carried out demonstrates that our proposed approach to intention recognition is valid and therefore verifies its intention recognition capabilities in the context of games.

Acknowledgements

Firstly, I want to express my sincere gratitude to my supervisor, Samad Ahmadi, who helped me both professionally and personally throughout the lifetime of this work. He was responsible for encouraging me to undertake this challenging piece of research and also for supporting me when those challenges became significant.

Moreover, I owe special thanks to Jenny Carter and Simon Coupland, both staff members of the University's Faculty of Technology. They have provided constant support with all my technical and personal questions and issues while preparing this thesis and also throughout my whole research.

Last but not least I would like to thank my whole family, who supported me through all these years. In particular I want to thank my uncle, Martin, who helped me a lot with the English language, as well as Kareen, my girlfriend, who was always supportive and understanding during the many hours I spent in working on my research and this thesis.

Contents

List of Figures

List of Tables

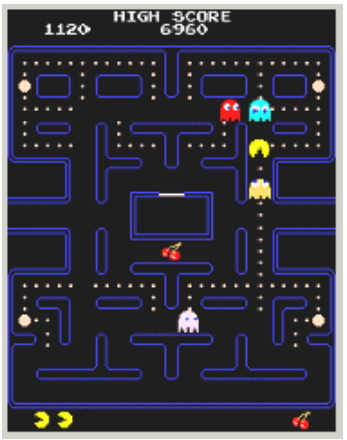
Chapter 1

Introduction

"Life is more fun if you play games."
– Roald Dahl

Computer games and thus the whole games industry is in a constant state of change (Egenfeldt-Nielsen et al., 2016). When games started to become mainstream about 30 to 35 years ago they were meant to be simple distractions, usually neither complex nor very hard to understand. Games at that time had simple mechanics, plain graphics (if at all), and basically no Artificial Intelligence (AI) built in. However, they were still fun to play and thus became more and more successful over the years. Examples of those early games – now classics – are *Pong*,[1] *Space Invaders*,[2] and *Pac Man*.[3]

(a) Space Invaders, 1978



(b) Pacman, 1980

Figure 1.1 – Sample Screens of classic Arcade Games.

Today, many of the modern computer games come with photo-realistic graphics, they often implement sophisticated mechanics and advanced artificial intelligence (Millington and Funge, 2009;

[1] Atari Inc., 1972
[2] Taito/Midway, 1978
[3] Namco/Midway, 1980

Yannakakis and Togelius, 2015). However, even modern game AI cannot usually outsmart a human player.[4] Thus people today tend to play more games online against real human opponents, instead of playing against artificial opponents, as has been the case since the gaming industry was young.

But in reality it is not always feasible or even possible to play a computer game with or against other human players. Maybe there is no online connection at the moment. Or there might no other player be available online at that moment. And of course one might simply want to practice offline and on ones own. Last but not least still some games just don't support online play at all, either because the game mechanics don't support that, or because the developers didn't implement such a feature because of limited resources during game development.

Whatever the reason is for playing a game offline, against the computer, it is the task of the artificial intelligence built into the game to simulate a real-world opponent as accurately as possible. This is the case for "simple" board game simulations, such as *Chess*, as well as for modern game genres, such as Real-Time Strategy Games (RTS). The player does not want to play against a dumb "mechanical" opponent, as that would become boring quite quickly. Instead, if the AI simulates a real human opponent, the player expects that artificial agent to play intelligently and as human-like as possible.

Today artificial opponents and Non-Playing Characters (NPCs) are much more intelligent than 30 years ago, when gaming became mainstream (Miyake, 2015). On one hand this simply comes from better hardware, which allows for more complex simulations than before. But this improvement also comes from advances in research on AI in the academic world. The games industry – like most industries – picks those parts from scientific research that help to develop better products (such as artificial intelligence for games).

Amongst these advances that can help to develop better game AI and therefore better games is the concept of player modelling and related techniques. In general the idea is that if the AI is able to model the player during game play as accurately as possible, it can more easily adapt to the human's playing style and thus lead to a more realistic and more challenging experience for the human player. More background information on player modelling (and related game AI concepts) can be found in a recent survey paper by Yannakakis and Togelius (2015). However, before one can model someone or something, one first needs to develop the concepts and constraints of that model.

1.1 Modelling User Intentions in Games

As one of our two main contributions in this thesis we will develop and describe such a *model of user intentions* in the context of (computer) games. For that model we are not so much interested in how *Human-Computer-Interaction* (HCI) works technically, but instead what the human player has in mind when he plays a game. We will evaluate and define concepts that are related to that question, such as actions, plans and goals. Then we will build a model that shows the relationship between those concepts. Later this model will be used as basis for our work on the main topic of this thesis: *goal and intention recognition*, which will be introduced in the next section.

[4] Although there have been some impressive success stories recently, as for example discussed in (Gibney, 2016)

But before we talk about intention recognition, let us first consider why we really need a model of user intentions. One answer to that question becomes apparent when you review the literature on various topics related to user intentions and behaviour in games, as well as plan and intention recognition in games and in general. One of the issues discovered during our own literature review for our work is the fact that different researchers come up with different definitions of what user intentions and related concepts are in the first place. Examples of these include *actions*: are these real events that happen in the game (or interactive environment), or are they abstract and custom definitions? Or consider *intentions*: what is an intention really? Is it just some artificial and vague concept borrowed from psychology, or is an intention something real that we can measure?

So one of the reasons why we want to have a model in place when we deal with user intentions (in games) is simply to avoid ambiguity. Having clear and exact definitions of all related concepts we are dealing with, such as actions, plans, goals and intentions, enables us to handle them precisely, without ambiguity. Moreover such a model can also help other researchers who work in a related field, as they can make use of that model for their own work. And of course, if two want to compare their work in related fields, they need to speak the same language. Our proposed model of user intentions can help with this task, as it provides definitions and relations of common concepts found in user intentions in games.

The second reason for why such a model is helpful is that of measurability. In our work we are dealing with a computer system that has the task of recognising the player's goals and intentions. That process involves various aspects of measuring data, for example to evaluate what is going on in the game world. A model helps in this task as well. It not only provides a set of concepts and definitions, which allows us not only to express what is going on in the game world in the form of numbers, but also shows the relationship between those concepts. For example, our proposed model defines how a plan can be represented by a sequence of actions and how an action affects the game world. The details of the model we propose are given later in this thesis. But let us first introduce the two aforementioned concepts, plan and intention recognition.

1.2 Plan and Intention Recognition

Even though the work presented in this thesis is focused on Intention Recognition (IR), not Plan Recognition (PR), both topics are closely related. Since it's therefore hard to describe one without also describing the other, this section will introduce both concepts.

Both terms, IR and PR, are composed of two parts: *intention* or *plan* on one hand, and *recognition* on the other. As outlined above the former two are related to user intentions. This thesis will describe both concepts in detail in the next chapter, but for the sake of clarity we can already summarise them in a few lines. First, the intention is a concept defined in the psychological and philosophical studies. It refers to the state of a (human) mind. When somebody has a particular intention, then he typically has something in mind that he wants to achieve. The intention therefore refers to the human's desire to achieve a particular goal. The concept of plans is closely related to intentions. A plan usually refers to a sequence of steps or actions someone comes up with in order to achieve a certain goal. In other words: the intention indicates *what* the human wants to achieve, while the plan defines *how* this could be achieved.

This becomes obvious when we look at an example: the well-known board game Chess. An experienced Chess player, i.e. one who not just "randomly" plays his moves, usually has the intentions of changing the game board to certain (intermediate) states. He might for example be interested in taking the opponent's Queen. This does however not tell how he wants to achieve that. But as already outlined, it is the plan that specifies such a sequence of actions. So in our example the Chess player might come up with the plan to take the opponent's Queen by a sequence of four certain moves. If the plan succeeds, the intention goal state is reached. Note in this context that one might have more than one intention at the same time. This in turn leads to the fact that one might also follow more than one plan at a time as well. As we will see later, this is one of the difficulties with automated intention and plan recognition in general.

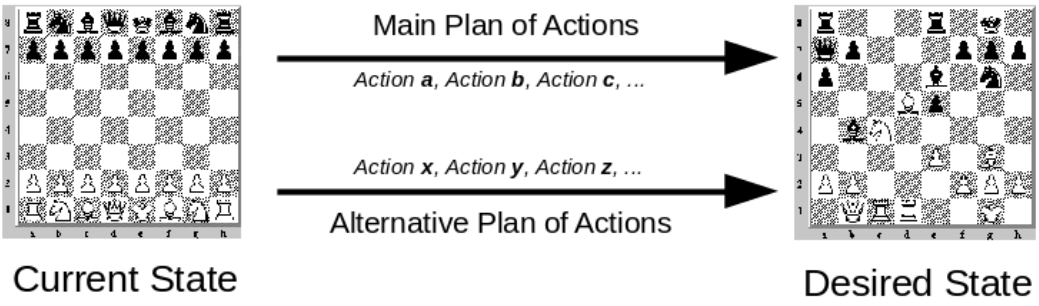


Figure 1.2 – Different plans can lead to the same goal.

Now let us see how this relates to the second part of both IR and PR: *recognition*. That can also be described in the context of the previous example. Usually a Chess game is played by two players. In our example player A has the intention to take the Queen of player B. He thinks of a plan to achieve that desired state, and so turn after turn player A follows the steps of his plan. If we now further assume that player B is experienced as well, he would be trying to anticipate what player A has in mind, i.e. *why* he plays exactly these moves he is playing currently. If player B succeeds in that anticipation, he can counter the attack of player A and maybe turn the game in his favour.

In general this example leads us to the definition of both IR and PR. The former describes the process of "reading the other's mind", i.e. trying to find out what the other is trying to achieve. This is very closely related to PR, because one's intention itself is not visible to the others. So in order to find out that intention, one has to observe what the other is doing. That in turn relates to the plan, which as described above defines the steps involved in achieving a certain goal. PR is therefore the act of identifying what plan one is currently following, and thus trying to anticipate what step(s) might come next.

When one considers either PR or IR, one usually refers to the automated process of these concepts. Research in the past has tried to find an answer to the question whether (and if so then to what extent) a computer system can try to recognise both the user's plans and intentions. This however has more than a scientific impact. Various companies can benefit from such a system. An online search engine for example may provide better search results if it is able to anticipate what the user has in mind when he conducts a search with just one or two words. We can see such technologies

in action every time we perform a search on *Google* for example: the search engine will not only try to provide better results, it also suggests related search terms while the user is still entering his terms.

In this thesis we are focusing on intention recognition in computer games. The area of games is particularly interesting for both IR and PR. That is because as we have seen in the previous example of the two Chess players it is indeed important for (experienced) players to try to know what the other player has in mind at the moment. If in a computer game the system is able to recognise the player's intentions and plans as well, like humans try to do, it can adapt to the playing style of the human player. This not only can enable the system to play in a more challenging way for the human player, it also leads to a more interesting experience for him in general, because the artificial opponent plays more like a real human being. And as described in the beginning of this chapter, it is in the general interest of the games industry to equip their games with better AI, which is able to play as human-like as possible.

For example, Microsoft Research recently published its research on a related topic by Lee et al. (2014). That research proposes a system that tries to compensate for network latency in cloud-based gaming environments. This is achieved, amongst other techniques, by predicting player actions based on past actions using a Markov-based prediction model.

The work presented in this thesis describes an intention recognition system called "iRecognise". While previous research on that topic has usually focused on both, IR and PR, by observing the human's actions, we instead propose an IR system design that allows to recognise the user intentions solely by monitoring the changes in the game world state over time. This novel approach is the second main contribution of our work presented here and will be described and evaluated in detail in the following chapters.

1.3 Original Contribution

The original contribution of the work presented in this thesis is twofold. First, we are presenting a model of user intentions in games. This model defines various concepts important for the description and work with player behaviour. Further our model describes and defines how these concepts relate to and influence each other.

Second, we are proposing a novel approach to intention recognition in games. That approach is based on the idea of representing a game world state in terms of numerical values only, *world state indicators* as we call them, and to use them as sole input for the intention recognition system. To our knowledge no such approach to intention recognition has been presented so far.

Additionally this thesis also presents three other contributions based on our work. First, as we use the board game *RISK* as test environment for our experiments, the game's complexity is being analysed in section 5.3. Next this thesis presents *iRecognise*, an implementation of our approach to IR in games written in Python. Last but not least we have implemented an iRecognise plugin for the Open Source implementation of the RISK game, Domination[5]. This plugin is used to observe

[5] `http://domination.sourceforge.net/` (accessed 10.07.2016)

game player actions and their effects on the game world. Both, the iRecognise software itself, as well as the plugin for Domination, are described in detail in chapters 5 and 6.

1.4 Thesis Structure

The remainder of this thesis is structured as follows. In the next chapter we will present and analyse previous research in the areas of plan and intention recognition, as well as user interaction and behaviour in general. Afterwards in chapter 3 we will present the first main contribution of our work, the model of user intentions in games. That chapter not only provides definitions of important related concepts, such as actions, goals, or intentions, it also defines the relation between those concepts in a formal manner. Chapter 4 then introduces the second main contribution of our work, the novel approach to intention recognition in games using world state indicators. Afterwards in chapter 5 we describe the case study used for our experiments. In that study we apply our approach to intention recognition to the board game RISK. In chapter 6 we present the experiments we have conducted in the context of this case study and evaluate their outcome, while in chapter 7 we summarise our work and present ideas for successive research.

Chapter 2

The History of Plan and Intention Recognition

This chapter provides an overview on past research related to the work presented in this thesis. The chapter's main focus lies on two areas: user behaviour and interaction in games, as well as intention and plan recognition. But before we evaluate the question of how other researchers have defined user interaction and behaviour, the next section gives a general overview on artificial intelligence in (computer) games.

2.1 Artificial Intelligence in Games

Long before the first computer games were published, researchers and engineers alike have already experimented with computers and how they could play (board) games on their own. Amongst the first who published their work in that area was Alan Turing (Copeland, 2004). Even though he did not build a game-playing machine on his own, he did for example worked on a Chess-playing algorithm. Chess as an example application also helped Turing to show what computers are able to do in general. But Turing was not the only one who used Chess as test bed for his ideas, it has been subject to a lot of work on AI in games (Levy et al., 2009).

As discussed in the introduction chapter of this thesis, good artificial intelligence is of particular importance for various kinds of computer games. There are many examples of how AI has been applied to successful commercial games throughout the last 30 years, as described by Millington and Funge (2009). Notable examples are path finding algorithms in *Warcraft*[1], machine learning in *Creatures*[2], or real-life simulation in *Full Spectrum Warrior*[3].

Also in the academic world games play an important role in the research on artificial intelligence. Examples for this have been mentioned in the introduction chapter of this thesis. In another

[1] Blizzard Entertainment, 1994
[2] Cyberlife Technology Ltd., 1997
[3] Pandemic Studios, 2004

example, Palazzo et al. (2013) introduces a BDI framework for computer games, based on the Belief-Desire-Intention model by Bratman, which we will see later in this chapter again. An interesting topic in both, the academic world and the industry, is *Procedural Content Generation* (PCG) for computer games (Togelius et al., 2013). Here the computer system generates content for computer games, such as maps or 3D models, which is usually created manually by a human.

2.2 User Interaction and Behaviour

Before we can evaluate the details of behaviour in computer games, we first need to consider traditional games in the real world. When we humans play a game, what do we have in mind, what are we thinking, and why do we play the game in the first place? Moreover, as we have seen in the introduction before, there is no single or consistent definition of concepts related to user intentions and interaction amongst researchers, so that in this section we are mainly interested in how these concepts have been defined in the past.

Amongst the first researchers who asked these questions and tried to answer them in a formal way were Von Neumann and Morgenstern (1953). While focused on the mathematical method in economic behaviour, in his work Von Neumann puts that into relation with behaviour in games when he states that "*the typical problems of economic behaviour become strictly identical with the mathematical notions of suitable games of strategy.*" Von Neumann further explains that even though some of the notions are fundamental for the discussion of games, in our everyday language their use is often highly ambiguous. For his work he therefore proposes some definitions of termini technici related to games.

Von Neumann and Morgenstern (1953) first distinguish between a *game* and a *play*. While the former is simply the totality of the rules that describe it and the name that is associated with it (like "Chess"), the latter refers to one specific instance of that game being played, from the beginning to the end. The next distinction that is explained is between *moves* and *choices*. Similar to the first definition a move denotes a possible option from a set of alternatives, defined by the rules of the game, while the choice is one particular instance of such an alternative chosen by a player during a play. In other words: "*the game consists of a sequence of moves, and the play of a sequence of choices.*" While Von Neumann does not explicitly define what a plan is, an important concept as we will see later on, he does put it into relation with *strategies*. These represent a player's selection of choices throughout a play. According to Von Neumann a strategy is a complete plan, one which includes all choices the player would make for every possible situation in a play.

Another theoretical framework of user intentions is presented in (Johnson et al., 1988), in which the authors present *Task Knowledge Structures* (TKS). These are related to *Task Analysis* (TA) and *Human-Computer-Interaction* (HCI) and can be applied to the generation of design solutions. In the context of our research we are in particular interested in the theoretical definitions of TKS, and how they put various concepts of human behaviour into relation. Johnson et al. (1988) present their definitions in a top-down direction, from high to low level:

- A **role** is defined as a set of tasks that an individual is responsible for performing, like his or her job. In the context of our work this relates to the "job" of a human playing a game.

- A **goal** refers to the world state that a particular task can produce. Obviously a person's role influences their choice of goals.
- A **plan** is a particular formulation and possible ordering of (sub-)tasks. These are being undertaken to achieve a particular goal.
- A **procedure** is a particular element of behaviour that forms part of a (sub-)task. Different groupings and orderings of procedures for the same (sub-)task are possible, so that a person can choose from different alternatives, based on their current goal and role. These alternatives therefore reflect different strategies.
- **Actions** are the lowest level of tasks, they are constituents of procedures.

An important concept that has not been discussed so far in detail is that of *intentions*, as described in (Spyrou and Darzentas, 1996). In that paper intentions are considered to be future-directed mental states, they are responsible for guiding the planning. In other words: intentions coordinate actions. For their model of intentions the authors follow the theory of *Rational Agent*, as presented in (Cohen and Levesque, 1990). That theory expresses intentions in logic, whose model theory is based on possible-worlds semantics. Agents choose the world they would like to be in, i.e. those worlds in which their goals are true. The theory of Rational Agent defines two such goals: the trivial or maintenance goals, which the agents believes are already true and which it wants to maintain, and the achievement goals, which the agent believes are currently false.

Further, the model of intentions presented in (Spyrou and Darzentas, 1996) defines plans as "*particular configurations of beliefs and intentions.*" Plans are thus always related to intentions. For the remainder of their model Spyrou and Darzentas follow those definitions we have already seen in (Johnson et al., 1988): tasks are performed in order to achieve one or more goals (i.e. to serve one or more intentions), while actions are the most elementary components, which assemble procedures. They correspond to the user's or agent's activities.

A very detailed and often cited work on a model of human practical reasoning has been presented by Bratman (1987). His model is described as a way of explaining future-directed intention: *Belief-Desire-Intention* (BDI). That model becomes relevant to our research when we see it in the context of intelligent or rational agents, which can also incorporate the BDI model. In that context Bratman's BDI model can be summarised as follows (Meneguzzi et al., 2007):

- **Beliefs** "*represent an agent's expectation regarding the current world state or the possibility that a given course of action will lead to a given world state.*" Note that what an agent believes may not necessarily be true and may change in the future.
- **Desires** "*represent a set of possibly inconsistent preferences an agent has regarding a set of world states.*" They represent objectives that an agent would like to accomplish.
- **Intentions** "*represent an agent's commitment regarding a given course of action, constraining the consideration of new objectives.*" These stand for what the agent has chosen to do. In the context of software agents this means the agent has begun performing one or more actions in order to achieve these objectives.

2.3 Plan Recognition

Knowing that humans and other agents plan their actions ahead, as described in the previous section, a tempting idea comes to mind: By sensing the actions of an agent, would it be possible to recognise its current plan(s)? Many researchers have tried to answer this and related questions in the past.

While in general many of the proposed approaches can be summarised as *plan recognition*, we can also divide them into different categories. According to Sadri (2010) there are three main types of plan recognition: *logic-based* (or symbolic), *probabilistic*, and *case-based*. We will see examples of all these in the following sub-sections.

Further, the area of plan recognition can also be split into *keyhole recognition* and *intended recognition* (Carberry, 2001). The former category deals with unobtrusive approaches, where the agent does not attempt to impact the recognition process (in fact, he might not even be aware of the recognition process at all). The latter category represents those situations in which the observed agent is aware of the recognition and might either try to help with the process or to obfuscate his actions, like in warfare. Note that for our own work, as presented in this thesis, we focus on keyhole recognition.

2.3.1 Symbolic Plan Recognition

Amongst the first explicit definitions of plan recognition is the one given in (Schmidt et al., 1978), which proposes three main concepts:

- Take as input the sequence of actions performed by an actor;
- Infer the goal pursued by the actor; and
- Organise the action sequence in terms of a plan structure.

While the approach presented in (Schmidt et al., 1978), focused on one actor making and eating something, is somewhat limited, it already contains a discussion of the two main challenges still remain to be solved today. First there is the question of how one can organise the recognition system's knowledge in a structured way suitable for the recognition process later. The paper proposes to split this knowledge into three models: the *World Model*, the *Person Model* and the *Plan Model*, where a model in general is defined as a (highly structured) collection of instances of the concepts and relations. The second important question to answer is, by utilising the aforementioned knowledge structure and sensing the actor's actions, how the system can recognise the actor's current plan and predict the next (few) actions. The authors of (Schmidt et al., 1978) address that by what they call *Expectation Structures*. These are set up in the plan model and "*represent the actions the observer expects the actor to carry out in a particular physical situation.*"

A more formal theory of plan recognition is presented in (Kautz, 1991). In that work Kautz explains that most research on plan recognition before him has focused on "*specific kinds of recognition in specific domains,*" while the work presented in his paper "*accounts for problems of ambiguity,*

abstraction, and complex temporal interactions that were ignored by previous research." His work focuses on keyhole plan recognition, where the observed agent is not aware of being observed (in contrary to intended plan recognition, in which the observed agent is aware of this fact and tries to structure his activities in order to make his intentions clear).

Kautz's work tries to address the aforementioned issues with plan recognition by a number of explicit closed world assumptions, and conclusions that are absolutely justified on the basis of the observations. He refers to both actions and plans as *events*, while the recogniser's knowledge is represented by a set of first-order statements called *event hierarchies*. These define the abstraction, specialisation, and functional relations between various types of events. *End Events* are special events that are not components of any other events, and plan recognition is therefore defined by Kautz as "*the problem of describing the end events that generate a set of observed events.*" He mentions as main characteristics of his approach that it does not assume a single plan that is underway, nor that the sequence of observed actions is complete, nor that all steps of a plan are linearly ordered. On the other hand, neither goals nor beliefs are explicitly represented in Kautz's approach.

While Kautz also compares his work with probabilistic approaches to plan recognition, he also notes that their strength over his work lies in their capabilities to allow for certain a priori plans that are more likely than others. Furthermore, his approach is not able to recognise new plans by chaining together the preconditions and effects of other plans.

2.3.2 Probabilistic Plan Recognition

The authors of (Charniak and Goldman, 1993) see plan recognition largely as "*a problem of inference under conditions of uncertainty.*" The key aspect of this approach is that the plan recogniser must be able to decide to what degree the evidence of recognised actions and user intentions supports a plan hypothesis. The approach presented is based on the retrieval of potential candidate explanations, which are then assembled into a *Plan Recognition Bayesian Network*.

According to the authors, referring to the work of Kautz mentioned above, there are three main flaws in the symbolic or logic-based plan recognition approach:

1. **Minimal set covering**: If the observed actions lead to two competing plans, both possible because of the knowledge stored in the plan library, then the system is not able to choose one, even if one of the plans is much more likely than the other.

2. **Top-level plans**: These are minimised, while other plans are not. As an example the action of walking is mentioned, which often is in service of a higher plan, i.e. to get somewhere. However, walking can also be a top-level plan on its own, if for example the agent is "just in the mood" of walking around. Kautz' work cannot distinguish between those plans, because all actions are being minimised.

3. **Set minimisation for abduction**: The authors state that in many cases two (or more) plans at the same time are more likely than one particular plan. They provide the example of

medical diagnosis, in which it is more likely that a patient has two common ailments rather than a single uncommon one.[4]

Still within the area of probabilistic plan recognition Charniak and Goldman (1993) present their work as a system that makes use of two kinds of knowledge representations. While the first is predicate-calculus-like to represent general knowledge about plans and actions, the second one is the Bayesian network, which contains the probability information. Further, the proposed system makes no distinction between plans and actions, i.e. plans are just complex actions with sub-actions. According to the authors, the key to their model is their proposed set of rules to translate from a plan recognition problem into Bayesian networks. They conclude that since their proposed approach can deal with the shortcomings of symbolic plan recognition systems, as described in the previous section, it is the superior method of plan recognition.

The symbolic plan recognition approach has also been criticised in (Yin et al., 2004), because of two main reasons. First, "*classical plan recognition algorithms often make the ideal assumption that the initial states of the world are completely known and the effects of an action are always deterministic.*" For obvious reasons this is not always the case. The second point of criticism "*is that most time plan recognizers can only derive a set of equally plausible plan hypothesis.*" The work of Kautz and related approaches select a plan with least commitment amongst all alternatives, which is clearly not applicable in every problem domain. Further, the authors state that the plan libraries used in symbol-based plan recognition systems need to be hand-crafted a priori with known actions, goals and plans, which naturally restricts the possible domains of applications for such systems.

To address the aforementioned issues with symbolic plan recognition Yin et al. (2004) propose a probabilistic approach that does not rely on a priori plan libraries. For their work they list the following factors that influence the problem of plan recognition:

- A set of operators that can be instantiated to actions.
- A finite, dynamic universe of typed objects.
- A set of propositions called the initial conditions.
- A set of goal schemata specifying possible goals.
- A set of observed actions and a set of actions that can be inferred with probabilistic effects.
- An explicit notion of discrete time step.

Further, they define a *state* as a complete description of the world at a single point in time, and actions as trigger to transitions from one state to another. Each action consists of a set of preconditions, which describe the states the action is applicable in, and a set of probabilistic effects they have. The authors conclude that even if their approach produces good results, more research work is necessary in order to make the system scale well with larger and complex domains.

[4]However, for this last point of criticism the authors unfortunately do not provide an example related to plan recognition.

Following the trend of probabilistic plan recognition systems that can be used also for larger domains Albrecht et al. (1998) propose a Bayesian plan recognition system for adventure games. The domain, Multi User Dungeons (MUD), is of interactive adventure games, which can be quite large. The example game used by the authors is *Shattered World*, a MUD that consists of over 4,700 (virtual) locations and 20 different goals. For the presented research over 7,200 actions have been observed. And as the authors further explain in this domain "*the plan recognition problem is further exacerbated by the presence of spelling mistakes, typographical errors, snippets of conversations between players, newly defined commands and abbreviations of commands.*"

In order to deal with such a large search space, the proposed approach does not rely on hand-crafted plan libraries, but instead incorporates *Dynamic Bayesian Networks*, for which several different models are presented. The authors have identified the following parts as relevant for their research:

- **Network Nodes**: The nodes in the network represent domain variables. *Actions* represent the possible actions a user may perform in the game, *Locations* represent the possible locations of the player when performing an action, and *Quests* that represent the quests a player may undertake when playing the game.
- **Network Structure**: The authors present and evaluate four different network models for their plan recognition system. These differ in the way the network nodes are connected to each other, i.e. how information flows through the network and how probability values are updated.
- **Probabilities**: Each node in the network contains a *conditional probability distribution* value, which represents the probability of that node based on the combined probability of its parent nodes. These values are generated from training data that has been extracted from real game sessions.
- **Belief Updating**: After the network has been created and all probabilities in the nodes have been calculated, new data from a user is added. The network then updates its belief on the user's next action and next location, as well as their current quest.

The authors demonstrate that their proposed approach works well within the sample domain (MUD), so that they see their work fit for the application to other domains as well.

A more recent approach to probabilistic plan recognition is based on plan tree grammars and presented in (Geib and Goldman, 2009). The authors claim that their approach, *PHATT*, is the first single system that provides a solution to these issues and constraints related to previous plan recognition systems:

- The observed agent is only pursuing a single plan at a time.
- The observed agent's plans are totally ordered.
- Failing to observe an action means it will never be observed or the observer will see an arbitrary subset of the actual actions.
- The actions within a plan have no explicit temporal relations.

- The plan representation is purely propositional. That is, actions do not have parameters.

Like other approaches presented in this section, PHATT takes a Bayesian approach to plan recognition. The key idea is that plans are highly dynamic, and the actions the agent performs depend on what previous actions have been performed. Agents perform those actions that help them achieving their current goal, but they only perform an action if it has been enabled by a prior action of the plan. These actions that have been enabled by previously performed actions are collected into what the authors call the *pending set*. Those actions that have no prerequisites form the initial pending set. By observing the agent's actions and re-generating the pending set, PHATT is able to provide explanations. Based on the initial probabilities augmented to the plan library the system can then calculate the conditional probability of each explanation, and therefore in turn the agent's plans and goals.

2.3.3 Case-Based Plan Recognition

A different approach to plan recognition than those presented in this chapter so far is using *Case-Based Reasoning* (CBR), as for example described in (Kerkez and Cox, 2003). The main difference from previously proposed solutions is that the case-based approach not only observes the agent's actions, but also the state the world was in when an action was performed. According to the authors, this approach leads to a better performance than taking only the actions into account.

Referring to (Barès et al., 1994), which is known as the first proposed approach to case-based plan recognition, Kerkez and Cox (2003) present the manner the system represents plans as the main improvement in their work. Plans (i.e. the cases) are stored as sequences of action-state pairs, so that in contrast to previous plan representations the new system's plans also contain intermediate state information. That way the system is able to match actions to plans in arbitrary points in the stream of observed actions. In order to compensate for the more complex state representation, the authors propose to simplify that by using a smaller, abstract state-space. All world states in the system presented in that paper are represented by sets of logical predicates (ground literals) that are true (closed world assumption). Goals are similar to states, with the exception that they do not necessarily contain the description of the full world state, but only those parts relevant to the goal. We will use some of these ideas also in our own approach, as presented later in this thesis.

Further, the approach presented in (Kerkez and Cox, 2003) can also deal with incomplete plan libraries. This is important in particular for large domains in which many different plans are possible, so that they cannot a priori easily be included into a plan library. Moreover the proposed system is also able to deal with unknown, new actions, which have not been seen before. It is in particular these latter two advantages that makes the proposed approach successful and distinct.

2.4 Intention Recognition

In (Sadri, 2011) intention recognition, or goal recognition, is described as "*the task of recognizing the intentions of an agent by analyzing some or all of their actions and/or analyzing the changes in the state (environment) resulting from their actions.*" Plan recognition, as described in the previous

section, extends that concept by recognising the agent's current plan and its steps. The paper lists the following challenges in the context of intention recognition:

- Multiple competing hypotheses are possible regarding the agent's intention.
- Actions may be only partially observable.
- Multiple concurrent intentions and interleaving plans to achieve them.
- Alternate plans may be executed in parallel to achieve the same intention.
- Actions that are executed in error and confusion.

Further, the paper lists three main components of any intention recognition system:

- A set of intentions from which the system chooses;
- Some sort of knowledge on actions and plans and how they contribute to goals; and
- A sequence of observed actions executed by an agent.

Even though many of the aforementioned challenges have been solved or at least addressed so far in past research, according to Sadri (2011) still some other challenges remain. These include the removal of (a priori) plan libraries from the recognition systems, the recognition of actions performed by multiple agents at the same time, as well as actions that are only partially observable.

A more general view on intention recognition is given in (Xu et al., 2009). In particular, the authors of that paper give a theoretical framework of what they call *intention computing*. An important thing to note in this context is coming from the research on intentions in philosophy and psychology. In both areas two main approaches of intention identification of human behaviour are known: *direct-perception-framework* (solely based on observed actions), and *inferential framework* (also based on context factors, such as prior knowledge). As the authors further explain, in the digital world intention identification consists of three main concepts:

- **Agents**: The subject of intention, who generates and "owns" the intention. An agent can be either a human or an artificial construct, like a robot. In any case however it has the following two features: (1) it is able to perceive the environment via sensors and (2) it acts upon the environment via actuators.
- **Intentions**: Considered as an agent's state of "mind", both action- and goal-oriented. An intention includes a goal and an action path from the initial (current) state to the goal state. As the authors note, time and timing is also relevant in the context of intentions.
- **Intention Computing**: The process of computing the probability of holding a specific intention by an agent at one time. The probability of owning intention is the product of the probability of owning a goal and choosing an action path to that goal.

As Xu et al. (2009) further explain, one cannot easily recognise any agent's intention without considering a range of assumptions. In particular, they name five important assumptions:

- **Rational Agent**: Agents are usually rational, so they will likely follow those action paths (plans) and try to achieve those goals which they can benefit from the most. In general we can say that the aim of a rational agent is to optimise their benefit. This is closely related to Bratman's BDI model we have seen above, as it is easier to recognise the agent's intention if we know their beliefs and desires.

- **Inertial Thinking**: People tend to apply what they know from their past. This helps in intention recognition, as for example if there are multiple action paths to the same goal, and we know that the observed agent has used one of these plans in the past to pursue that goal, then it is likely that they will follow the same action path in the future.

- **Behavioral Structure**: Sequential associations among the actions composing the agent's behaviour help us to understand their intention better. For example, if we know that one action is part of multiple plans to different goals in our library, then by observing only that action we cannot easily infer intention. However, if we further observe a subsequent action that is only part of one action path, this in turn helps us understanding the agent's overall behaviour better.

- **Semantic Relativity**: The meaning of intentions can be represented by some linguistic concepts, such as letters, words, or images. Therefore, any linguistic description relating with the intention can be used for semantic analysis to get the agent's real intention.

- **Contextual Confine**: The context of the agent is important to understand their intentions. In particular the following three questions can help: Where, Who, When?

Intention recognition in the context of mobile systems is further discussed in (Kiefer, 2011). The author of that book argues that the difference between PR and IR is solely defined by the output (plans vs. intentions), but not by the knowledge used internally: "*An intention recognition approach may use plan libraries to structure intentions.*" However, as Kiefer further explains, PR is the harder problem, as multiple plan structures may lead to the same intention. In the context of mobile systems intention recognition seems to be sufficient, as providing the right information to the user is more important than explaining why the intention occurs.

A recent paper on intention recognition in games and simulation environments by Yue et al. (2014) proposes the use of *Hidden Markov Models* (HMM), a special case of Bayesian networks. The author's approach, which they call *Logical Hidden Semi Markov Models*, is able to apply the task of IR to environments with multiple agents, which in general is a difficult problem to solve, as shown in various past research on this topic. The authors present their proposed model and describe a custom, simple game they created to evaluate their approach. The proposed approach seems to be promising, however, no details are given on how well the HMM-based approach works for real-world applications.

Furthermore intention recognition is also applied in the area of (mobile) robotics. In general, using IR, the robot is enabled to anticipate human behaviour by sensing their movement and interaction with the robot. Recent examples are described in Tan and Kawamura (2015), Huang et al. (2015), and Xu et al. (2015).

2.5 Discussion

This chapter has evaluated a selection of research on various concepts related to the work presented in this thesis, from modelling user behaviour and interaction, over past approaches to plan recognition, to different aspects of intention recognition. By reading the literature outlined above it becomes clear that much research has been undertaken so far on the topics related to our own research. User behaviour is not only being modelled in the context of games, but also in a much more general way, for example when it comes to user modelling in the area of *User-Computer-Interaction*. Further, as the literature presented in this chapter shows, plan recognition systems and the theory behind them is not something new we came up with, but a research area many others have explored so far from various perspectives. Last but not least our main topic, intention recognition, has been subject to past research as well.

However, by carefully reading and evaluating the literature presented here we came to the conclusion that two topics in particular have so far not been examined in detail. One of these is a model of user interaction in games, which includes all related and important concepts. Much research has been undertaken on specific concepts, such as actions or plans for example, but we were missing one model of user interaction, focused on games, which not only describes all related concepts, but also their interactions and dependencies. We will address that in the next chapter on our proposed model of user interaction in games.

The second area which from our perspective lacks detailed research so far is the evaluation of changes in the world state over the course of a game play in order to recognise player intentions. So far we have seen approaches mainly based on actions and their effects on the game world. However, as we will see in chapter 4 below, in our approach of intention recognition in games, instead of focusing on player actions, we solely observe how the game world changes over time. In turn, as we will see, a computer system can make use of that information to infer player intentions without the need for any kind of plan library.

Chapter 3

A Model of User Intentions in Games

This chapter describes our proposed model of user intentions in games. As outlined in the introduction such a model is an essential part of an intention recognition system. We will first evaluate related concepts, such as actions, plans, goals and intentions, and how they have been defined in previous research. Secondly, the chapter presents a model of user intentions, which demonstrates the relationship between these concepts and specifies how they influence each other.

It should be noted that the model concepts and their definitions are not new. In fact the definitions given below are based on past research on the respective concept.[1] Here we have put all these concepts into relation with each other, and present all of them as part of a unified model. This does not only help us in our own research on intention recognition, as discussed in this thesis, but also may help other researchers looking for such a model and build on in their own research.

3.1 Agents

Usually (computer) games are played by human beings; in this thesis we refer to them as *players*. It is however not necessarily a human that plays a game. An intelligent computer program can also play games, for example IBM's *Deep Blue* which became world Chess master in the last century (Campbell et al., 2002). In addition, even trained animals could play some kind of games (Humphrey, 2012). Last but not least it could also be a robot that might play a game (Barakova and Lourens, 2010).

To simplify such an enumeration of possible game players we could refer to them all as just *players*. Computer science in general however does not solely focus on game players, but on users interacting with a computer system in general. In that context the term *player* would be misleading (or even just wrong). We are therefore interested in something that can interact with a (computer) system,

[1] As referred to in the following sections.

that is, perform various actions on that system and perceive its output, even though it might not be a human being. It is therefore logical to summarise everyone and everything that could do so into one single term.

Various different scientific disciplines have (informally) agreed on such a term: *agents*. A very general definition by Russell and Norvig (2009) states that an agent is simply anything that can "*perceive its environment through sensors and act upon that environment through actuators.*" Based on this general description of an agent for the research presented in this thesis we will utilise a similar definition.

> **Definition 1**: An agent performs actions in an interactive environment, such as a computer game. The agent can perceive its virtual environment through sensors and act upon it through actuators.

In this context we will focus on agents as human players. Note however that in general another computer system or program could act as a system user or game player. An example, which will be extended in each section throughout this chapter, is presented below.

> **Example**: The example application for all concepts defined in this chapter is the classic arcade game *Pac Man*.[2] In that game there are two types of agents: Pac Man itself, controlled by the human player, and the four ghosts, controlled by the logic hard-coded into the game.

Note that, as shown in this example, usually in the context of computer games one distinguishes between *Player Characters* (PC, those controlled by the (human) player) and *Non-Player Characters* (NPC, those controlled by the artificial intelligence implemented in the game[3]).

3.2 Actions

In interactive environments *actions* represent the basic commands the user can issue to the computer system in order to control it. From clicking on a button to firing a virtual gun in a game, the set of all actions defined for a given interactive environment represent all possibilities the user has at hand for system control. From the perspective of an artificial agent, the set of all these actions defines the global search space when constructing a valid plan (as defined in the next section).

However, as described in (Luger, 2009), it is necessary to distinguish between atomic actions and abstract actions. The former represent the real ("physical") actions that can be issued to the environment, they cannot be split into smaller sub-actions any more. Unfortunately this may lead to a problem when we want to recognise the plan of the environment's user: the impact of such an atomic action to the game environment may be so small that it does not change the world state (significantly).

[2] Released 1980 by Namco today the game Pac Man is considered one of the most successful examples from the golden era of arcade games. *The player guides [Ms.] Pac-Man through a maze filled with pills and power pills. At any point, the player can choose to move up, down, right or left, but can never move through walls. A level is cleared when all pills have been eaten. Unfortunately, four ghosts are haunting the maze and will kill the protagonist upon contact. However, after eating a power pill, the tables are turned for a short time, during which Ms. Pac-Man will kill ghosts upon contact.* (Cardona et al., 2013)

[3] Even though it can be argued whether such NPCs always seem to be *intelligent*. But in our opinion an NPC even, for example, standing somewhere and waving towards the player character needs some kind of "intelligence".

Example: In the game Pac Man there are exactly four atomic actions, and they are simple to define: *Up*, *Down*, *Left*, and *Right*. Each of them corresponds to user input and controls the movement direction of the Pac Man character throughout the maze. However, observing one instance of such an action might not reveal much information. If neither a ghost or pill is anywhere close to the current location of Pac Man, the observer can hardly recognise what the player is aiming for by looking at one single action.

In order to address this issue we define the concept of abstract actions, as proposed by Luger (2009). These actions can be seen as a sequence of atomic actions. Such an abstract action does not correlate to a plan, as it does not include a certain goal to achieve. Instead, an abstract action summarises two or more atomic actions that together represent the action performed that led to the given change of world state.

Example: A combination (sequence) of multiple atomic actions, i.e. an abstract action, in Pac Man could result in the player character moving to a certain location, e.g. a junction in the maze close to Pac Man. For example we could thus define *move-to-left-junction* or *move-to-right-junction* as two abstract actions in the game.

This leads us to the second definition in our model:

Definition 2: In an interactive environment an action represents an elemental operation that can be performed by the agent. All possible actions together form the global search space of actions for the given environment. One has to distinguish between atomic actions, those that cannot be split into smaller sub-actions, and abstract actions, those comprised by a sequence of one or more atomic actions that lead to changes in the current world state.

Note that an action definition can also include a set of preconditions. These must be fulfilled before the action can be performed. For the sake of simplicity however in this thesis we do not deal with these preconditions, they are considered to be fulfilled in any case.

3.3 Plans and Strategies

A simplified definition of a plan, as given in the discussion of the BDI model in the previous chapter, is that it represents a sequence of one or more actions. Traditionally, as outlined by Luger (2009), planning relies on various search techniques. From an artificial agent's perspective planning is to search "*through a space of possible actions until the sequence necessary to accomplish a task is discovered*" (Luger, 2009). In various problem situations planning therefore can be seen as a search problem. However, from a general perspective, thus also considering human agents, a plan can be defined as a strategy for acting: "*Planning involves choosing a plan by considering alternative plans and reasoning about their consequences*" (Chen, 1999). By performing actions as specified by a plan we want to achieve some goal, as defined in the next section.

Definition 3: In an interactive environment a plan is a sequence of one or more actions performed by the user in order to achieve a specific goal.

Example: Let us assume the player wants to consume a certain power-pill that is close to Pac Man (the goal). They could thus come up with the plan to first move to the next junction left,

then upwards to the following junction, and then left again where the power-pill is located. The player's plan therefore includes three abstract actions in a sequence.

In the area of computer games another definition of user interaction may be added: *strategies*. The word "strategy" often has a military connotation, closely related to military tactics and combat manoeuvres. Strategies are also related to business, as for example in a marketing strategy. As military and business theory are sometimes said to be based on the same roots, Mazzucato (2002) defines this concept as follows: "*The essence of strategy is choosing to perform activities differently than rivals do.*"

For our work we are however more interested in a formal definition of a strategy in the context of computer games and game theory. Straffin (1993) defines strategies as courses of action which a player (i.e. our agent) may choose to follow, where the "*strategies chosen by each player determine the outcome of the game.*" Strategies are therefore closely related to plans, as described above, but reside one step above them on our abstraction ladder. For our purpose we define a strategy as a set or collection of plans that share the same goal world state (as defined below), and one or more common properties. Beside a common goal these properties could for example also be specific actions used, or the context of the game world state in which these actions have been performed.

Definition 4: In an interactive environment a strategy is a collection of plans that share the same goal and one or more common properties. The types of these properties in different strategies can be of different nature. Strategies therefore cluster related plans.

Example: In our example Pac Man might follow two different routes to reach the power-pill. Next to the sequence of actions listed before an alternative might be to first move upwards, followed by two movements to the left. In both cases the outcome of the plans is the same, i.e. they share the same goal, even though the action sequences those plans are built of are not the same.

3.4 Intentions and Goals

3.4.1 Philosophical Discussion

Let us first evaluate what the word "intention" means. In her book *Intentions*, one of the classic works of 20th century philosophy, Anscombe (1957) provides an in-depth discussion of what intentions are (or could be). In this work Anscombe notes that intention has three distinct uses in the English language:

1. As an adverb: *A is Xing intentionally.*
2. As a noun: *A is Xing with the intention of doing Y.*
3. As a verb: *A intends to do Y.*

The work of Anscombe also defines the concept of an intentional action. According to that definition a human being performs an intentional action if the question to "why" (this action is being

performed) can be answered by her giving a reason for that action. In a more technical sense we can thus say that human beings perform (at least some of their) actions because they have the intention of performing those actions.

From a philosophical and psychological point of view there is further discussion on the topic of intentions ongoing, so that Anscombe's (and thus also our technical) definition of intentions is of course also subject to criticism. Bratman (1987) for example argues that Anscombe's "*belief/desire model of intention fails to adequately model the organizational role intentions play in practical deliberation.*" (Driver, 2011)

In this thesis however we will utilise Anscombe's definition that humans in many cases perform various actions because of their intentions to accomplish a specific goal or fulfil their current desire.

3.4.2 User Intentions in Interactive Environments

For this work focused on user interaction and intention recognition in computer games we are interested in particular how intentions relate to actions performed in an interactive computer environment. We will continue to use the concept that intentions are the "driving force" behind each such action, as discussed in the beginning of this chapter. Following Bratman's BDI model (Bratman, 1987) we define an intention as the agent's commitment to change the current world state to a specific, other world state (i.e. a goal). The intention is therefore a combination of both, a plan and its outcome, i.e. the goal.

> **Definition 5**: In an interactive environment, such as a computer game, the user's intention is the commitment to perform a sequence of one or more actions, a plan, in order to achieve a specific goal.
>
> **Example**: In the example above the intention would be the plan the player has chosen for active pursuit, as well as the goal to consume the next power-pill close to Pac Man.

3.4.3 Intention Context

As defined earlier, the user's intention is the commitment to perform some action to achieve a specific goal. This definition however omits an important factor of every intention: its context. In practice the intention context plays an important role, as a simple example demonstrates.

Let us consider someone (the agent) thinking of opening the entrance door of his or her house. His intention is to perform some action in order to achieve the goal of having the door opened. We can further easily think of different scenarios of why our agent has this intention. One of them could for example be that the doorbell rang and the intention to open the door is the agent's desire to let the visitor into the house. On the other hand our agent could simply have the desire to leave the house through the entrance door.

In both of these scenarios the goal is the same, i.e. to have an opened entrance door. The action to be performed are the same, and the intention to perform this action is the same as well. However, the reason of why the agent wants to achieve that goal differs in both scenarios. In general this fact of different reasons holds also for interactive computer environments, which brings us to the next definition.

Definition 6: In an interactive environment the context of the user intention defines the reason why some action is being performed to achieve a certain goal.

Example: In Pac Man one does not want to consume the power-pill for its own sake. The reason to do so is either to make the ghosts edible, which allows the player to hunt those ghosts for a short amount of time, or to gain additional points (or both). In our example three of the four ghosts are currently close to Pac Man, so it is a good time to consume a power-pill. In this example the intention context thus is to consume the pill in order to hunt those three enemy ghosts.

3.4.4 Goals

For a proper understanding of (user) intentions we need to define another related concept: *goals* (or *objectives*). When an agent intentionally performs some action, they usually have a specific goal in mind. In other words: they have the *desire* to reach a certain goal or objective. The agent could have more than one desire, i.e. different objectives. However, in many cases only one such objective is "active" at the current time, in particular if the agent cannot perform multiple actions simultaneously. This leads us to the definition of goals.

Definition 7: In an interactive environment a goal represents the user's desire that he or she has currently adopted for active pursuit. It is the user's intention to achieve this goal.[4]

Example: As already stated above, the goal is the outcome of a plan. In our example the goal is thus to reach and consume the power-pill close to Pac Man.

Note that in the context of our work a goal more formally refers to a certain goal world state the player aims for, as described next.

3.5 World States

Another concept that we need to define for our proposed model and the remainder of this thesis is the state of the interactive environment or the *world state*. That term refers to the state of the world (i.e. the environment, computer program, game etc.) at a specific point in time. This relates to our previous definitions as the user's intention is to change the current state of the environment to a certain other state, where each such state includes an exact and complete definition of the environment in terms of program variables, attributes, etc.

Definition 8: In an interactive environment the world state refers to the state of that environment at a specific time. It includes the values of all program variables, attributes, and the state of related concepts that are necessary to define that specific state.

Example: In Pac Man a world state describes the exact location of the player character, as well as the locations of all four ghosts, the number of pills and power pills left and their

[4]Note that in this thesis we use *goals* and *objectives* interchangeably.

position in the maze, the current level number, the time spent so far in that level, and the player score.

One question that is relevant in the context of world states is that of how those states can be defined or formally described. Various approaches may be possible, from simple key-value pairs (i.e. feature vectors), to complex, hierarchical approaches, and many more, as for example often discussed in the context of *Case-Based Reasoning* (Aha et al., 2005; Bergmann et al., 2005; Fagan and Cunningham, 2003).

3.5.1 Game World State Representation

As with many different types of computer systems we also want intention recognition systems to work in different problem domains. That is, we do not want to build a new system from scratch for every new problem. In our case, as we are focusing on games, the IR system must be applicable to different (types of) games. The world state representation thus must be compatible to all these different game genres. For the representation of the game world in our system of IR we have built on the idea presented by Kerkez and Cox (2003) of using *ground literals* (instantiated logical predicates). One positive aspect of ground literals is that they are domain independent, so that they fit into our system as well.

Of course there are also other ways to represent the world state. Amongst these alternatives are simple numerical values (key-value pairs, i.e. *feature vectors*), textual strings, XML- or JSON-encoded data, and others, each with their own advantages and disadvantages (Bergmann et al., 2005). But the main advantage of using ground literals instead of these alternatives is that logical predicates can be used to apply rules of first-order logic. For example, using ground literals one might deduce to a new world state from a given set of world states, or simplify complex world states. Additionally one could also describe logical relations amongst various given world states. Another advantage is that many of these first-order logical rules and transformations can be applied automatically, i.e. by a computer program.[5]

To illustrate how ground literals can be used to represent the state of a gaming environment we have examined the *blocksworld* domain, which is described by Kerkez and Cox (2003) as this:

> *"The simple blocksworld domain assumes that a single agent (illustrated by a [robot] hand, but not explicitly part of any problem) can re-configure various blocks using the following actions: pickup (OBJECT), putdown (OBJECT), stack (OBJECT, OBJECT), and unstack (OBJECT, OBJECT). Blocksworld problems contain various objects that are always blocks. The problem specifies an initial and a final desired arrangement of the blocks."*

The blocksworld example shown in figure 3.1 can be described using this set of ground literals:

```
(clear, B); (on, B, A); (on, A, C); (on-table, C); (arm-empty)
```

[5]Note that for simplicity reasons we do not apply any of those rules or transformations in this thesis.

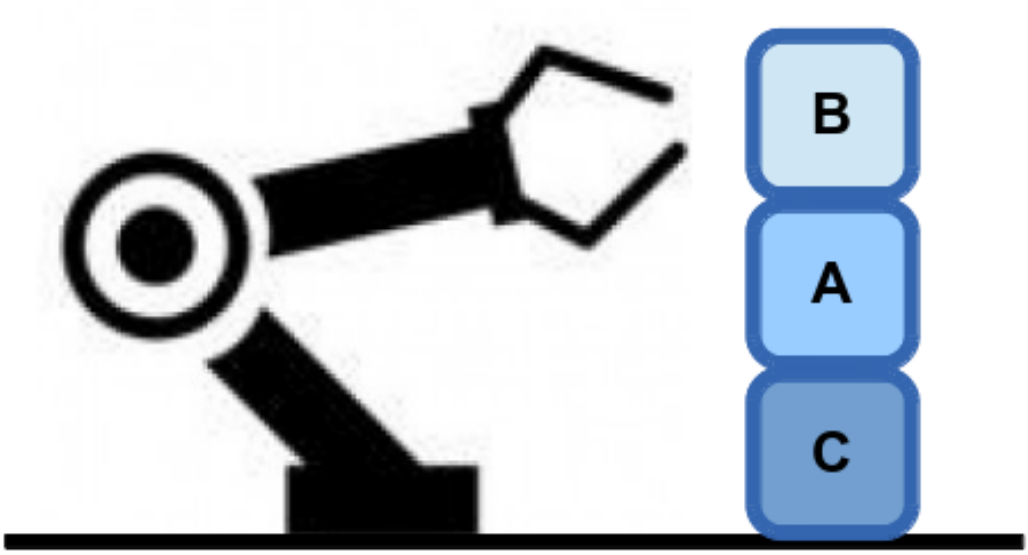


Figure 3.1 – Blocksworld Domain Example

These ground literals represent different aspects of a given state. The format of their notation is simply the name of one state aspect (e.g. "clear"), and zero or more arguments (e.g. "B"). The example above would mean that

- no block is on top of block B,
- block B is on top of block A,
- block A is on top of block C,
- block C lies on the table, and
- the robotic arm does not hold a block.

The concept of ground literals is not restricted to the blocksworld example. As a concept they are, as outlined above, domain-independent and context-free. However, to be used in a particular (game) environment a set of domain-dependent ground literals must be defined a priori. That typically takes place while the IR system is being designed, and in turn is domain-dependent.

Having a generic set of ground literals for game environments, which can be used in different games (or game genres), might be helpful in saving time when designing a new IR system for games. We believe that such a generic set of ground literals would be an interesting area of research. It is, however, not in scope of this thesis. How such a set of ground literals can be designed and used is described in the section on our practical use case in chapter 5.

3.5.2 World State Indicators

In our proposed system we make use of *World State Indicators* (WSI) to abstract the world state information contained in the ground literals described in the last section. The main idea is that we

are not directly interested in the world states themselves, but in the *changes* of different aspects of the world states over time. We propose to represent both, the current world state as well as these changes over time, using world state indicators.

> **Definition 9**: A World State Indicator, or WSI, represents one specific aspect of a given (game) world or environment (for example "main character health points"). A combination of multiple WSI can represent the gaming environment as a whole. Each WSI is encoded as simple floating point value.[6]

For a better understanding think of how a human game player in Chess (the observer) would try to recognise what his opponent could have in mind at the moment (i.e. the opponent's current intention). The observer would think of a few possible intentions his opponent might have. Then by continuing to observe what the opponent is doing the observer would see how parts of the world state change due to the actions performed by the opponent. For example, the distance between the two Queens on the Chess board might change. Then by looking at these changes in the world state over time the observer can narrow down which intentions are more likely than others. Our approach of intention recognition follows the same idea by using the aforementioned world state indicators.

A significant characteristic of WSI is that they can be used to represent many different aspects of game worlds. It is also desirable in this case to start with only a small selection of such indicators and then gradually improve the solution by adding more and altering existing indicators. Further, those WSI fit well in our approach, as we can use simple numerical values to represent the game world (like key-value pairs), and in turn monitor their changes over time. Note that as described above we use ground literals to represent the game world state at commencement. In chapter 4 we illustrate how the system transforms these ground literals into numerical WSI values.

A simple example of how world state indicators can abstract the world state of a game and how various aspects of that game world change over time can be seen in the game *Connect Four*. This game is described on Wikipedia[7] as follows:

> *"Connect Four is a two-player connection game in which the players first choose a color and then take turns dropping colored discs from the top into a seven-column, six-row vertically suspended grid. The pieces fall straight down, occupying the next available space within the column. The object of the game is to connect four of one's own discs of the same color next to each other vertically, horizontally, or diagonally before your opponent."*

This game has a relative small search space (in comparison to many other games), yet it is still complex enough to allow for a few different ways of playing and trying to win the game. An evaluation of Connect Four and its complexity can be found in (Edelkamp and Kissmann, 2008). Note that we will discuss a more complex game with a much larger search space in chapter 5 below.

As we are dealing with intention recognition, we can assume that there is one observed player (OP), and one opponent or enemy player (EP). The world state of this game, regardless of how it

[6] Note that WSI values may, but do not need to be normalised to a value range of $[0.0, 1.0]$.
[7] `http://en.wikipedia.org/wiki/Connect_Four` (accessed 10.07.2016)

is being represented technically, contains information about each of the cells on the game "board". The first step to represent this world state in terms of WSI, as numerical values, is therefore quite apparent:

- Number of OP's horizontal/vertical/diagonal connections of two pieces
- Number of EP's horizontal/vertical/diagonal connections of two pieces
- Number of OP's horizontal/vertical/diagonal connections of three pieces
- Number of EP's horizontal/vertical/diagonal connections of three pieces
- Number of OP's horizontal/vertical/diagonal connections of four pieces (either 0 or 1 = win)
- Number of EP's horizontal/vertical/diagonal connections of two pieces (either 0 or 1 = win)

For an overall understanding of what is going on in the game world of Connect Four that selection of WSI might be sufficient. However, in order to recognise the user's goals and intentions we may have to consider further how experienced players plan their strategies. Contrary to more complex board games, like Chess or Go, in Connect Four there are not many different ways of winning the game. However, there are still a few playing strategies known, which are discussed online.[8]

In general it is advantageous for a player to focus on the middle column of the game board. Experienced players will therefore always try to occupy as many cells in that middle column as possible. Another important factor in winning a game of Connect Four is to play in a row such that no enemy pieces are either left or right of the own pieces. We can thus add four more objectives to our list of WSI:[9]

- Number of OP's pieces in the middle column
- Number of EP's pieces in the middle column
- Number of OP's pieces in a row with no enemy pieces on either side
- Number of EP's pieces in a row with no enemy pieces on either side

Having these indicator values in place an intention recognition system can now detect whether a player is for example concentrating his play on the middle column or not. The next chapter will explain in detail how WSI are used in our proposed approach to recognise player intentions. But first we will combine all model concepts seen in this chapter so far in order to define the formal characteristics of our model.

3.6 Summary

An example will illustrate how all these definitions fit together: Alice and Bob (the agents) are in the middle of their Tuesday-evening Chess game (the context). By looking on the current Chess

[8] For example at `http://www.wikihow.com/Win-at-Connect-4` (accessed 10.07.2016)

[9] Naturally advanced players might find even more important aspects of the game which might be good candidates for additional indicator values.

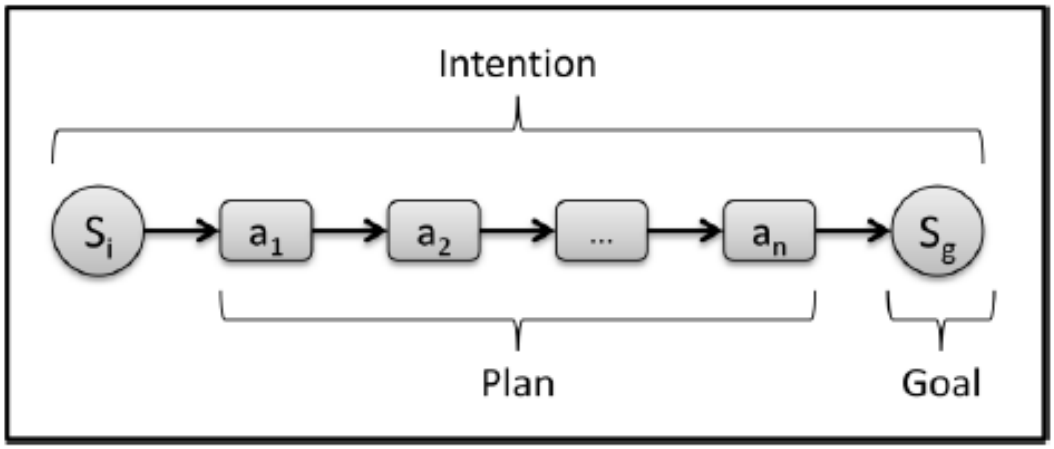


Figure 3.2 – Overview on Actions, Plans, Goals, and Intentions.

board configuration (the world state) Alice seems to be close to winning the game. Knowing that fact Alice has the desire to beat Bob. She decides to win the game by a checkmate (the intended world state) that should be possible to achieve from the current Chess board configuration. Alice's intention is to achieve that goal by flanking Bob's King with her remaining Bishop and a Knight. Her plan is to attack Bob's King with her Bishop, towards her Knight, and then to flank the King with this Knight. After Alice has moved her Bishop (action-1), she plans to move her Knight in the next two turns (action-2, action-3) towards her opponent's King. Shortly after, Bob is staring at the Chess board not knowing how he could have lost the game against Alice for the first time ever.

Based on these definitions we are now able to construct a full model of user intentions, as we use it in our work presented in this thesis.

3.7 A Model of User Intentions

In this section we describe the characteristics of our proposed model of user intentions in games. First the definitions discussed so far in this chapter are presented in a formal way, afterwards these definitions are put into relation. That latter section shows how these concepts of user intentions influence each other.

3.7.1 Definitions

- $T = (t_0, t_1, ..., t_n)$
  is the ordered list of $n + 1$ time steps over the course of one observed game (play).
- $W = \{w \mid w \text{ is a world state}\}$
  is the unordered, finite, and countable set of world states, i.e. all possible world states in the game domain.
- $W_p = (w_{p_0}, w_{p_1}, ..., w_{p_m})$
  is the ordered sequence of $m + 1$ world states in one specific game play p. w_{p_0} refers to the

initial world state, i.e. the state of the game when it starts, before any player action has been performed, and w_{p_i} refers to the i-th world state at time step $t_i \in T$.

- $g = (name, arg_1, arg_2, ..., arg_n)$
  refers to one ground literal g, defined by a $n+1$-tuple containing the ground literals name and a list of n arguments. *name* may refer to an activity, which applies to the arguments, or a state of arguments, etc. For example: $(move, actor, location)$

- $\forall w_i \in W : w_i = \{g_{i_1}, g_{i_2}, ..., g_{i_m}\}$
  is the i-th world state of all world states in W, described as a set of logical predicates (ground literals). Note that $m \geq 1$, i.e. each world state includes at least one ground literal. As described in section 3.5 we use these predicates to describe (parts of) the state of the world at a given time step. Note that we do not necessarily need to describe every single detail of the game world. For the work presented here we describe only those parts of the game world, which are relevant for our ultimate task of intention recognition. We selected *relevant* predicates (that is, parts of the game world state) by analysing sample games and based on what features of the game world human players use in recognising their opponent's intentions.

- $V = \{v_1, v_2, ..., v_n\}$
  refers to the set of n world state indicators, as described earlier. See equation 3.1 in section 3.7.2 below for a definition of how indicator values relate to world states.

- $A = \{a_1, a_2, ..., a_n\}$
  is the unordered, finite, and countable set of n actions. Note that this notation is applicable for both, atomic and abstract actions. The list of actions is domain-dependent and thus has to be defined a priori by the system designer.

- $a_i \in A$ and $a_i = (name_i, e_{i_1}, e_{i_2}, ..., e_{i_m})$
  refers to one action a_i from the set of all actions A, defined by the action's name and a list of m action effects. See section 3.7.2 for a definition of how an action affects the world state.

- $p_i = (w_{i_0}, a_{i_1}, a_{i_2}, ..., a_{i_m})$
  refers to a plan p_i, which is defined by the current world state w_{i_0} and a sequence of m actions.

- $o_i \sim w_j \in WS = \{g_{i_1}, g_{i_2}, ..., g_{i_m}\}$
  indicates that a goal (objective) o_i is defined as a world state w_j and defined by a set of m ground literals. Note however that a goal may include only a subset of ground literals in one world state. In that case the state (or values) of the remaining ground literals are not restricted for this goal, the goal is then defined as a set of world states (instead of only one world state). Mathematically, for every $i \in \mathbb{N}$, there is a $1 \leq j \leq |W|$ and a world state w_j where o_i represents w_j or one of its subsets. $|W|$ is the number of world states.

- $i_j = (player_j, o_j, p_j)$
  refers to a player's intention defined by the (name of the) player itself, his or her desired goal world state o, i.e. the world state the player wants to change the current world state into, and the plan p to achieve this goal world state. That plan in turn contains the current world state, as defined earlier.

3.7.2 Model Characteristics

In this section we want to incorporate the definitions of user intention concepts presented above into a model of user intentions. As we will see in chapter 4 below we want to calculate the current probabilities of player intentions. For that we need to represent a given world state in form of numerical values (we call these *World State Indicators*, WSI). Our model of user intentions thus needs a function f^v that maps a given world state w to a set of n real-numerical *indicator values*:

$$f^v(w \in W) \rightarrow \{v_1, v_2, ..., v_n \mid v_i \in \mathbb{R}\} \tag{3.1}$$

where $n \geq 1$. We refer to this resulting set as V_w.

Having sets of numerical values we can now quite easily compare two such sets. This approach can be useful if it is required to compare the current world state to a given goal world state, i.e. to get the difference between those two world states. This difference can, for example, be calculated as distance (subtraction) between the indicator values that define both world states.

Since an action can have an impact on the (numerical representation of the) world state, we can measure this effect of action a_i too, with a function f^e:

$$f^e(a_i) \rightarrow \{e_{i_1}, e_{i_2}, ..., e_{i_m}\} \tag{3.2}$$

where e_{i_j} denotes the j-th effect of action a_i from a set of m effects.

One such effect e_i as part of action a_j is defined by how it changes one numerical world state indicator value v of the world state w_t in the current time step t, in comparison to the world state w_{t-1} in the previous time step $t-1$:

$$e_i \simeq f^e_{v_i}(a_j) \rightarrow f^v(w_t) - f^v(w_{t-1}) \tag{3.3}$$

In order to determine what has changed between w_t and w_{t-1} in terms of one specific word state indicator we simply calculate the distance between that indicator value in w_t and w_{t-1}. Note that calculating this distance is only one way of determining the difference between two given world states. For example one might also use a time series instead.

A plan is a sequence of actions. Each action has an effect on the world state, i.e. it alters (at least) one aspect of the world state, so the plan as a whole has an effect as well. The total effect of plan p_j on the world state is therefore the sum of the effects of all k actions contained in the plan sequence:

$$f^e(p_j) \rightarrow \sum_{\substack{i=1 \\ a_i \in p_j}}^{k} f^e(a_i) \tag{3.4}$$

The plan itself is defined by an initial (the current) world state and a sequence of k actions, which alter the current world state w_c one after another:

$$p_j = (w_c, a_1, a_2, ..., a_k) \tag{3.5}$$

Actions being part of a plan represent functions applied to a given world state at (current) time step t, changing it in comparison to the previous time step $t-1$:

$$a_i(w_t) = f^e(a_i(w_{t-1})) \tag{3.6}$$

where f^e of action a_i is defined in equation (3.2).

The goal of a plan is considered the world state after the last action of a plan has been performed. Thus a goal leads to a plan of n actions, where the current world state w_0 is modified by actions a_1 to a_n. We can formalise such a goal g by recursively listing the effects of all n actions in the plan p in reverse order:[10]

$$g \simeq w_n = a_n(w_{n-1}) = a_n(a_{n-1}(...(a_1(w_0)))) \mid a_i \in p \tag{3.7}$$

The total set of all goals available to a player corresponds to the total set of all possible world states in the game.

In addition to a plan p_j leading to a certain goal, i.e. world state w_g, there might be other plans leading to the same world state as well. In general to achieve a certain goal, multiple plans might be available to the player. However, based on the current world state and the set of actions available to the player it might also be that currently no (valid) plan at all can lead to that goal world state.

In order to evaluate whether or not the search space of available actions can lead to a valid plan that turns the current world state to a goal world state, one can apply a brute force algorithm over the whole search space. However, depending on the domain (the game and its actions in our case), such a brute force approach might not be applicable, due to the complexity of the search. To determine whether or not there is a plan in the search space to achieve a goal world state is a significant problem on its own. As we are dealing with plan and intention recognition rather than planning itself, addressing this problem is not in scope of this thesis. The interested reader can find more information on this in relevant literature, for example in (Erol et al., 1992).

[10] Note that the game to be modelled and its rules need to be deterministic for this to work.

Chapter 4

Using World State Indicators to Recognise Player Intentions in Games

The chapters so far in this thesis have discussed previous approaches to intention recognition and formalised a model of user intentions. That model helps us to understand various concepts of human behaviour involved when one plays a computer game. This chapter is based on both – our model as well as the theory of plan, goal and intention recognition and presents our proposed approach to intention recognition in games.

As the following sections will explain, our approach to IR is novel with regard to how the IR system handles and evaluates information from the observed game. In practice this means that the system does not track any actions the player performs in the running game, but how these actions influence the state of the world over time. In addition to the model of user intentions presented previously it is this novel approach of intention recognition that forms the main contribution of the work presented in this thesis.

4.1 Intention Recognition via World State Indicators

As we have seen in chapter 2 there are several different approaches to solve the plan and intention recognition problems. Most of them are based on observation of the user's actions and a pre-defined library of actions and their relations to potential plans and goals. As discussed so far in this thesis and also outlined in (Sadri, 2011) however, observing user actions for plan or goal recognition may lead to some difficulties. For example, not all actions might be "visible" to the recognition system. Also the user might be aware of this observation and trying to obfuscate his actions. For these reasons and those discussed so far in this thesis a plan or intention recognition system may therefore not always fully rely on user actions observed in a computer system.

Based on these considerations we have focused our approach of intention and goal recognition not

on any actions, but solely on the changes in the world state over time. Our design is based on two main ideas, which are described in the next section.

4.1.1 Observing Actions for Intention Recognition

Contrary to typical approaches of IR, in which sequences of observed actions are being analysed (Sadri, 2011), in our IR system we do not consider any actions. There is a simple reason for this: when performing actions in a (computer) game, assuming the player has at least some experience in that game, these actions are performed in order to alter the current state of the game world. Actions are performed to achieve a goal.[1]

Now let us consider the problems with *a priori* plan libraries, as discussed in chapter 2, in particular in the context of symbolic plan and intention recognition approaches. Those plan libraries often need to be constructed or at least optimised by hand. Also in many cases multiple different actions can lead to the same outcome, so that the plan libraries have to account for that as well. Thus, instead of focusing on actions performed in the game, we solely focus on their effects on the game world.[2]

4.1.2 Changes in Indicator Values

This section continues with the ideas on World State Indicators as described in section 3.5.2.[3] We have the WSI for *Connect Four* in place, so the next task is to consider how to make use of them in order to recognise intentions. As outlined above in our approach to IR we are not only interested in the values of the indicators at a given time step, but also in their changes over time. That is because a sole snapshot of the game world state of any single time step alone does not say much about what a player might have in mind at that time. Instead, we also need the history of past WSI values. In other words: in order to predict how the indicator values might shift in the near future, we need to look at how they have changed up to this point.

In theory the approach of observing the changes of WSI over time is as follows. At every time step, e.g. one player's turn in the game, the IR system receives a list of ground literals describing the current world state. The definition of these ground literals is of course domain dependent, in our case based on the observed game and its mechanics. In our example of Connect Four a complete world state could simply be described as a list of ground literals that describe each grid cell in the game board:

`(cell, <column>, <row>, <value>)`

where `cell` is the static name of the ground literal, `<column>` and `<row>` refer to that cell's column and row in the board grid, respectively, and `<value>` can be one of "empty", "OP", "EP". So in

[1] Even if that goal is to fool the other player(s).

[2] This approach may not work well for novice players who do not know what they are doing. Usually such inexperienced players do not follow any plans. For our solution and the remainder of this thesis we thus assume that players are not novice and have at least some experience in playing the game.

[3] A World State Indicator (WSI) represents one specific aspect of a given (game) world or environment. A combination of multiple WSI can represent the gaming environment as a whole. Each WSI is encoded as simple floating point value.

order to describe a full world state at one time step the system would receive a list of instantiated ground literals, i.e. without any variables. So for example one such "snapshot" in Connect Four could look like this:[4]

```
(cell, 0, 0, empty)
(cell, 0, 1, empty)
(cell, 0, 2, OP)
...
(cell, 5, 4, OP)
(cell, 5, 5, EP)
(cell, 5, 6, empty)
...
```

Receiving such a list of ground literals at every time step the IR system can now calculate the indicator values. This is again domain dependent and needs to be defined for every game manually. In our example of Connect Four this transformation from ground literals to indicator values is simple. The system just needs to count how many pieces of the OP respectively the EP are currently in the game, and how they are placed on the game board. This leads to a list of indicator values, which the system then saves into a database at each time step.

Having these values and their history in place the IR system can now process them. How it can "calculate" the goals and intentions based on the history of the objectives is described in the next section.

4.1.3 The Quest for Good World State Indicators

So far in this chapter we have described our approach to making use of world state indicators for the overall task of recognising the player's intentions. However, during that discussion we have not answered one central question: how do we identify *good* indicators in the first place?

Unfortunately, based on related literature we have reviewed and our own experiments, there is no single "golden rule", which alone would suffice to find the important or relevant indicator values to describe any given game. Every game has its own mechanics and rules, so research in the past has come up with custom definitions of what aspects of a game is important to fully define a world state in that specific game.

A typical example, often used as a test bed for various AI concepts, is Chess. In the real world, different (experienced) players have different strategies to follow when playing Chess. That is true in particular for world class players, who have developed their own playing style over the years. What one such player would consider as important aspect of the Chess board in order to evaluate a given board configuration might not be the same definition as that for another player. Or in other words: it is in the nature of game playing in general that the definition of what is important when looking at the game board or environment is based on the player's particular experience and playing style.

[4] Where $0, 0$ refers to the top-left corner, which is just a custom definition.

However, games that are widely known, in particular "traditional" board and card games, have often been evaluated in much detail by both expert players and researchers alike. That means we can make use of these evaluations when we wish to formulate our own definitions and evaluations. In particular what is required is a list of indicator values that fully describe the important aspects of any given world state representation. What is *important* in turn is usually defined by experienced or expert players, as discussed. That means we can simply ask one or more such player(s) to help us coming up with this list of indicator values. We believe that, while this approach is feasible, it would involve considerable time to gather and formalise these expert opinions.

But there is also a second option. For games that have been used for research in computer science and artificial intelligence in particular in many cases evaluation functions have been defined. These are used for example to guide an artificial player (agent) during game play, in order to make intelligent moves. Other uses of such evaluation (or fitness) functions can be found in AI techniques such as *reinforced learning* or *evolutionary computing*. These evaluation functions typically simply map a given board configuration, e.g. one "snapshot" of a Chess board at turn x, to an evaluation or fitness value. That value is just a numerical estimation or heuristic of how "good" that particular board configuration is. This is important especially if we want to compare two (or more) board configurations.

Technically there are no hard and fast rules for how such an evaluation function will calculate the value of a board configuration. By looking at related research however, as for example (Shannon, 1950) or more recently (Buro, 1999), it is often built from a set of (weighted) distinct features of the game. For our own research this is exactly what is required: features that describe a given game world state. We therefore propose that in order to find a suitable list of indicator values usable for the evaluation of a game world state we "extract" them from proven expert opinions in the form of evaluation functions. A practical demonstration of the effectiveness of this approach is given in the sample case study in a later section of this thesis.

Note that technically there is no restriction to having these evaluation functions defined by human experts. For example, this is shown in (David-Tabibi et al., 2011), where genetic algorithms are used to semi-automatically improve the performance of Chess evaluation functions. Such an approach might be an interesting option in this case as well, and it is our aim to focus on that aspect in our later research.

In general the task of selecting good indicator values is closely related to the topic of *feature selection*. Various research areas make use of such feature selection, including machine learning, text processing and statistics. These application areas all have in common the fact that a range of hundreds, thousands or even more potential features or attributes are available within the domain. To find and evaluate the most suitable selection of all those features different approaches have been proposed in the past. For example in (Guyon and Elisseeff, 2003) the authors describe multiple potential benefits of variable and feature selection: *"facilitating data visualization and data understanding, reducing the measurement and storage requirements, reducing training and utilization times, defying the curse of dimensionality to improve prediction performance."* Various approaches are presented in that paper, including techniques from the broad areas of variable ranking, variable subset selection, feature construction, as well as space dimensionality reduction. But regardless of which approach is being used to select suitable features for a given problem, an essential part of this feature selection process is the validation of those selected features, as also

described in (Guyon and Elisseeff, 2003). Note that due to the complexity and wide research on the whole topic of feature selection we will not go into detail here. The interested reader can refer to the aforementioned article and related papers on the subject. The approach we took for our work presented in this section is described in chapter 5, below.

4.2 Our Approach to Intention Recognition

We propose that by observing those changes in the numerical values of world state indicators one can calculate the probability of the observed player's current goal and therefore intention. This is the main idea in our approach, which represents the original contribution of our work.

Based on the discussion in this chapter so far, we can summarise our approach of intention recognition in computer games as follows: One or multiple player(s) are being observed. We are not interested in the player's actions, but instead in the effects these actions have on the game world. Therefore we ignore which actions in particular a player performs, but focus on the changes in the game world state over time. The game world is defined by world state indicators in the form of simple numerical values. Our proposed solution supports any arbitrary number of such indicator values. The selection of proper indicators depends on the game in question, and forms one of the main challenges of whether or not the system will be able to successfully recognise any intentions. Further, the intentions to be recognised need to be defined a priori by a domain expert, i.e. an experienced player of the game.

Note that there is one issue with our approach that needs to be addressed even with our handcrafted selection of world state indicators. Since we are ignoring the actions of the players, the system does not know whether a change in the world state is based on the observed player's actions, or because of any other event. We therefore need to tell the system not only the current value of each of the world state indicators, and the changes in these indicators over time, but also whether or not it was an observed player's action which initiated any change in the indicator value. We will see in the chapter on the description of the implementation of our solution how we can address this requirement in practice.

Currently, the selection of the WSI, the definition of possible intentions, as well as the relation between objectives and intentions has to be carried out manually by a domain expert. This works well for simpler games, or if only a few intentions need to be recognised. It is apparent however that this manual approach leads to a kind of rule-based system. Such rule-based approaches are not necessarily a bad thing on their own, but in general we want to make use of automatic solutions, at least for domains with a larger search space. We therefore see our manual, rule-based approach as prototype in order to evaluate our approach of WSI for IR in general. A future improvement for our further research in this area will then be focused on the automatic identification and evaluation of both, the world state indicators for a given game and the possible intentions to be recognised, as well as their relation to the indicators.

4.2.1 The Indicator-Goal Matrix

The next step after the definition and description of the world state, the world state indicators, as well as the intentions to recognise, is to define the relationship between the WSI and the intentions.

The basic concept is that changes in the game world state initiated by the observed player in total and over time indicate which player intentions are likely at that moment. The user plays specific moves and performs actions not just for their own sake, but to alter the world state in his favour. In the long run the player wants to win the game, so he tries to achieve intermediate goals that help him to reach that final goal of winning the game. The player therefore tries to alter the game state in such a way that it is closer to the desired world state which represents his current goal.

One exception to this rule could be if the player wants to fool the other players. This might in theory also fool the IR system. In practice however that does not have to be an issue. After a (supervised) training phase the IR system could be able to "measure" exactly those kind of changes in the world state that are known to be based on "false play" by the player. That way it could recognise that intention of fooling the other players on its own. One the other hand it is of course likely that the player can successfully fool an IR system as well, as smart as it might be. But in the end that "false play" on purpose is a very basic and important element of game playing in general, so in practice we would not consider that issue necessarily a bad thing.

Based on the consideration that the player tries to alter the current world state in his favour, the main challenge is now to evaluate and formalise how the changes in WSI relate to intentions. One obvious "rule of thumb" is to assume that since both, WSI and intentions, are defined a priori by game experts, their relation can be defined in that way as well. In fact, this approach is the one we follow for our prototype. This approach does work well as is demonstrated in the experiments chapter of this thesis. The main purpose of the proposed prototype is to demonstrate its ability to recognise intentions based on changes in the world state, which is also what this thesis focuses on. We therefore assume that a hand-crafted set of rules is sufficient for our prototype IR system.

To define these rules a matrix is created that lists all intentions in the rows, and all WSI in the columns. If an indicator is relevant to an intention the intersection of their row and column lists the rule that combines them. Such a rule describes how this WSI must change over time (i.e. in recent game turns or time steps) to influence the intention. It could be anything that describes such a change, for example a simple rule like "up" (i.e. that indicator values goes up over time), over a static value like "3+" (i.e. that indicator value must be at least 3), to any more complex rule. We will see how such an indicator-goal matrix can look like in practice when we describe our use case example in the next chapter. However, in general the matrix is built in a scheme shown in table 4.1.

	Indicator 1	**Indicator 2**	**Indicator 3**	**Indicator 4**	**Indicator 5**
Intention A	up		1+	last round +1	
Intention B		down	1+		
Intention C		down		= last round	up and <10

Table 4.1 – Sample Indicator-Goal Matrix

Note that those intersections that are empty in the matrix represent indicators that are not relevant for the intentions in these rows.

4.2.2 Alternative: Artificial Neural Networks

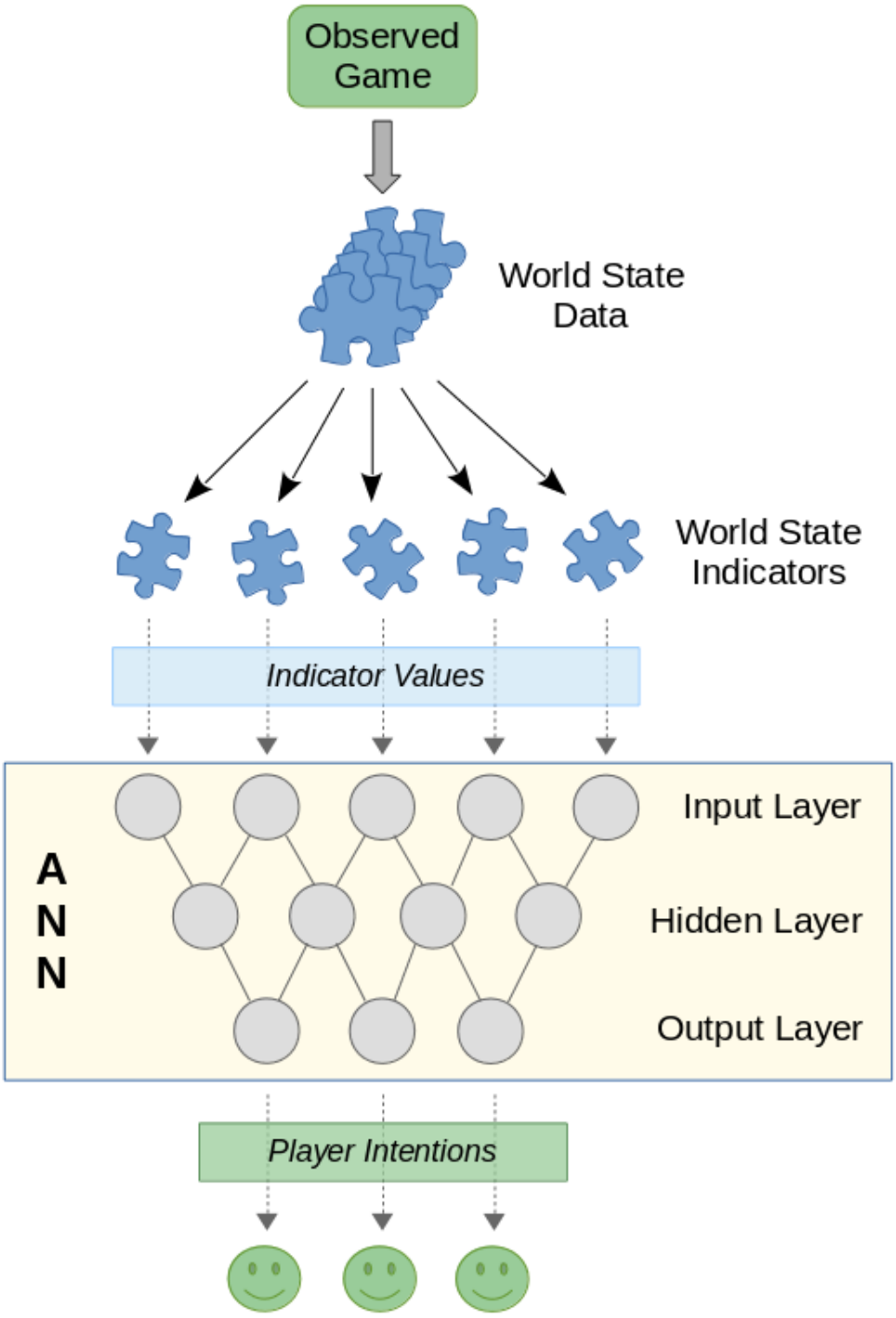


Figure 4.1 – Using an ANN for Indicator-Goal Mapping.

It is our intention to automate this manual rule crafting process in our further research. We have considered a number of options to achieve that automation. One option would be to use an *Artificial Neural Network* (ANN) for this task, as illustrated in figure 4.1. The WSI values would lead to the input neurons of such an ANN, while the output neurons represent the player

intentions. A training setup would allow human players to not only play the game, but also select their current intentions while they are playing and performing actions. Meanwhile, the training system would collect both the world state data and the selected player intentions, and store them in a database. Later the ANN can be trained in a supervised approach to build the connections from the input neurons (WSI) to the output neurons (intentions). This would lead to a "black box system", as it is not possible (or at least very hard) to determine exactly how the input values influence the output of the ANN. On the other hand we assume that approach would assist us in the elimination of the need for hand crafting the indicator-goal rules. Even though it is not directly related to our research, the work presented in (Bevilacqua et al., 2006) shows an example of how multiple indicators are used by an ANN classification system, which has been optimised using a multi-objective genetic algorithm.

4.2.3 Alternative: Decision Trees

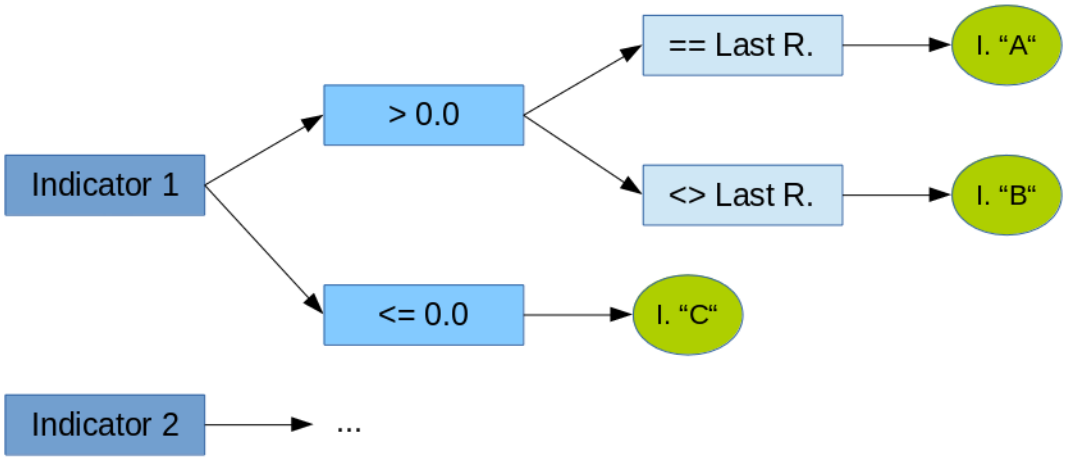


Figure 4.2 – Using DTs to replace Intention-Goal Matrix.

Another representation of the indicator-goal matrix described above is that of *Decision Trees* (DT). In general DTs are tree-like graphs, which support decision making processes. They can be applied to various problem domains, in our case they could replace the indicator-goal matrix, as illustrated in figure 4.2. Each indicator (blue rectangles) would require one DT, which in turn maps to an intention (green circles) based on the rules set in this tree. Since an intention usually refers to more than one indicator value, this representation would require the combination of multiple DTs in parallel. Note that, typically, a DT is manually crafted by a domain expert, so that we believe the previously described alternative, artificial neural networks, would lead to a better solution than the use of DTs.

4.2.4 Intention Recognition System Design

The following chapter describes in detail how the proposed IR system has been built for the practical use case application; this section now describes how that IR system works in theory.

Observed Game

All relevant data is collected from the observed game. Here the players interact with the game world and perform their actions. The IR system monitors one (or potentially also more) player(s). The IR system lives outside the game, but it connects to it via a plug-in. At every time step or game turn that plug-in gathers all relevant world state data and sends it to the IR system.

World State Data

The world state data collected in the observed game is being sent to the IR system. That data is described using instantiated logical predicates, i.e. ground literals. These ground literals have been defined a priori based on the observed games and its mechanics. Each piece of world state data contains a flag that signals whether or not it has been altered by the observed player during the last time step or game turn. This data set contains all information gathered by the IR/game plug-in, no data is being filtered out at this step.

World State Indicators

In the next step the world state indicators are "calculated" out from the complete world state data. These indicators have been defined a priori by a domain expert. They are measured in plain numerical values. Each such indicator value represents one specific aspect of the current game world state. When the indicator values are being read from the world state data at each time step, they are saved into the IR system database. That way the systems builds a history of all the values over time. This history enables the IR system to recognise which values go up, which go down, and which stay at the same value.

Indicator-Goal Rules

The WSI values are the "fuel" of the indicator-goal rules, which also have been defined a priori by an experienced player of the observed game, using an indicator-goal matrix. These rules define the mapping from the indicator values to the player intentions. Usually human intentions in a game relate to various aspects of the game world, therefore it is likely that each intention is mapped to the world state by a range of rules. Each such rule defines how the corresponding indicator value has to be or change if it indicates the rule's intention. In other words: each intention is defined by one or more indicator rules, whereas the "sum" of all rules must "fire" in order for an intention to become more likely. A rule can become as complex as necessary to describe the world state to intention mapping.

Intention Probabilities

The indicator-goal rules define the mapping from world state indicators to player intentions. Based on a) how close each indicator value matches the according rule, and b) how many rules match for an intention in total the probability of that intention is calculated by the IR system. These

probabilities can range from 0 to 1, inclusive, where a value of 0.0 indicates that intention to be not likely at all, where a value of 1.0 indicates a very likely intention. Multiple intention probabilities can be at the same value. The current values of the WSI and thus their history constantly changes over time – some more, some less. By updating these values at each time step or game turn in the IR system, it is able to constantly update the intention probability values as well.

4.3 Summary

Figure 4.3 shows a summary of how we relate our model of user intentions discussed in the last chapter with the proposed approach of intention recognition based on world state indicators.

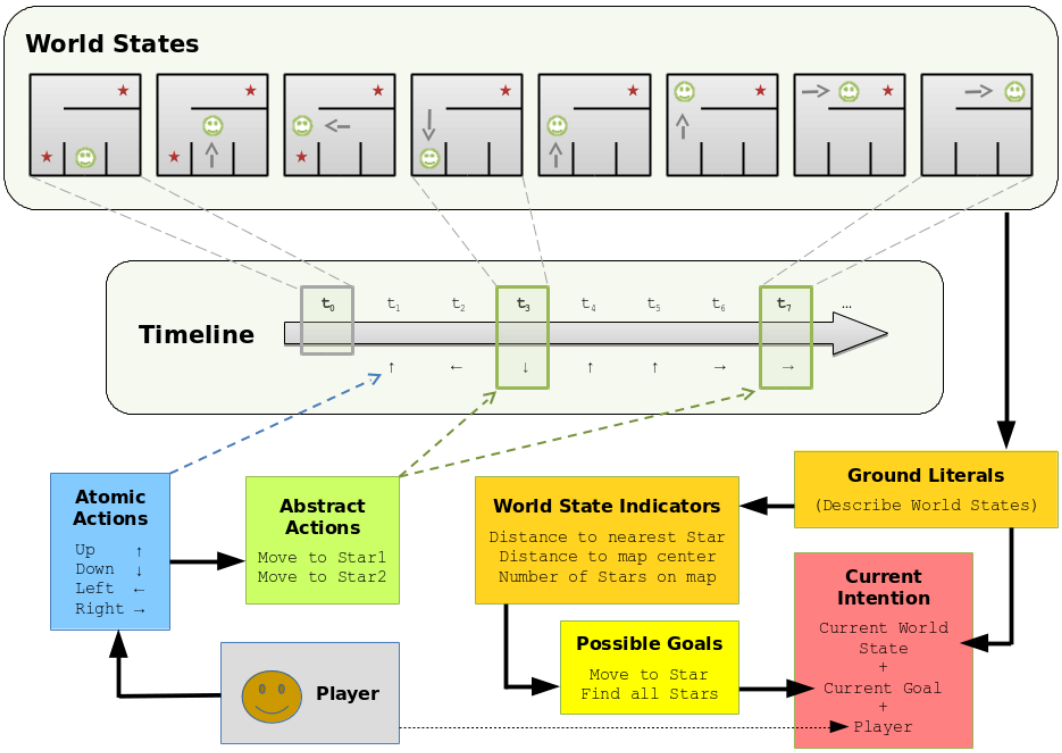


Figure 4.3 – Intention Recognition System Design

This graphical summary can be described as follows. The system monitors the changes in the world state over the course of an observed game. At every time step of the game the observed player may or may not perform an action in the game world. The set of actions is pre-defined for the game, as defined in our model of user intentions. Some or all of these actions have an effect on the world state. As we monitor the world state, we also monitor the changes in the world state over time using the selected WSI. Based on the a priori rules of indicator-goal relations the system is able to calculate the current probability of each goal. This in turn leads to the intention that is most likely at the moment, and therefore the player's current intention.

Chapter 5

RISK: A Case Study

One of the games we use in our research on IR is called *RISK*, a classic turn-based board game for two to six players, first released in 1957, now produced by Hasbro. The original version is played on a board depicting a political map of the Earth, divided into forty-two territories (countries), which are grouped into six continents. The primary objective of the game is world domination, or as the game's original handbook states "*to occupy every territory on the board and in so doing, eliminate all other players.*" Players control armies with which they attempt to capture territories from other players, with results determined by dice rolls.

5.1 RISK Game Elements

The classic board game version of RISK comprises the following parts:

- The **game board**, which represents an abstract map of our planet Earth. As mentioned before, this map includes 42 territories (or countries), divided into six continents. These territories represent political and geographical areas of our real world.
- A set of **territory cards**, each of them with the name of one of the 42 territories on the map. Every card shows one of the three army symbols for Infantry, Artillery and Cavalry. Additionally there are two extra joker or wild cards. These can represent one of the three army symbols, chosen by the players, as discussed in the next section.
- A set of **mission cards**. Depending on the edition of the game, and the year it was produced, different such mission cards are shipped with the game. These missions define what the players are supposed to do, e.g. to eliminate one particular other player. These missions are an optional element in RISK, without them each player has the same mission: to eliminate all other players and conquer the whole map.
- The **armies** (or units) of the players. Each army is represented by a unit piece to be placed directly on the game board. For convenience, depending on the edition of the game, there may also be larger pieces that represent more than one army, e.g. one medium sized piece for five units, and one large piece for ten units.
- A set of **five dice**. Three red dice for attacking, and two blue (or white) dice for defending.

Figure 5.1 – Original RISK Game Board.

5.2 RISK Game Rules

5.2.1 Overview

A game play of RISK can be summarised by the flow chart shown in figure 5.2. The blue steps are part of the setup stage in the game, while the green steps refer to the normal game play stage. Gray elements in the chart represent start and end nodes. These stages are described in detail in the following sub sections.

5.2.2 Setup Stage

The setup stage is the first stage in a play of RISK. The game starts with an empty board, i.e. all territories on the map are unoccupied. In the setup stage each player places his units on to the map.

When the game starts each player chooses a color and receives a set of n units in that color, where n is the same for every player in the game. Now players place their units in turn on to free territories on the map. There are more units in the game than territories, so that at one point no free territories are left, while players still have units to distribute. From that point on players still

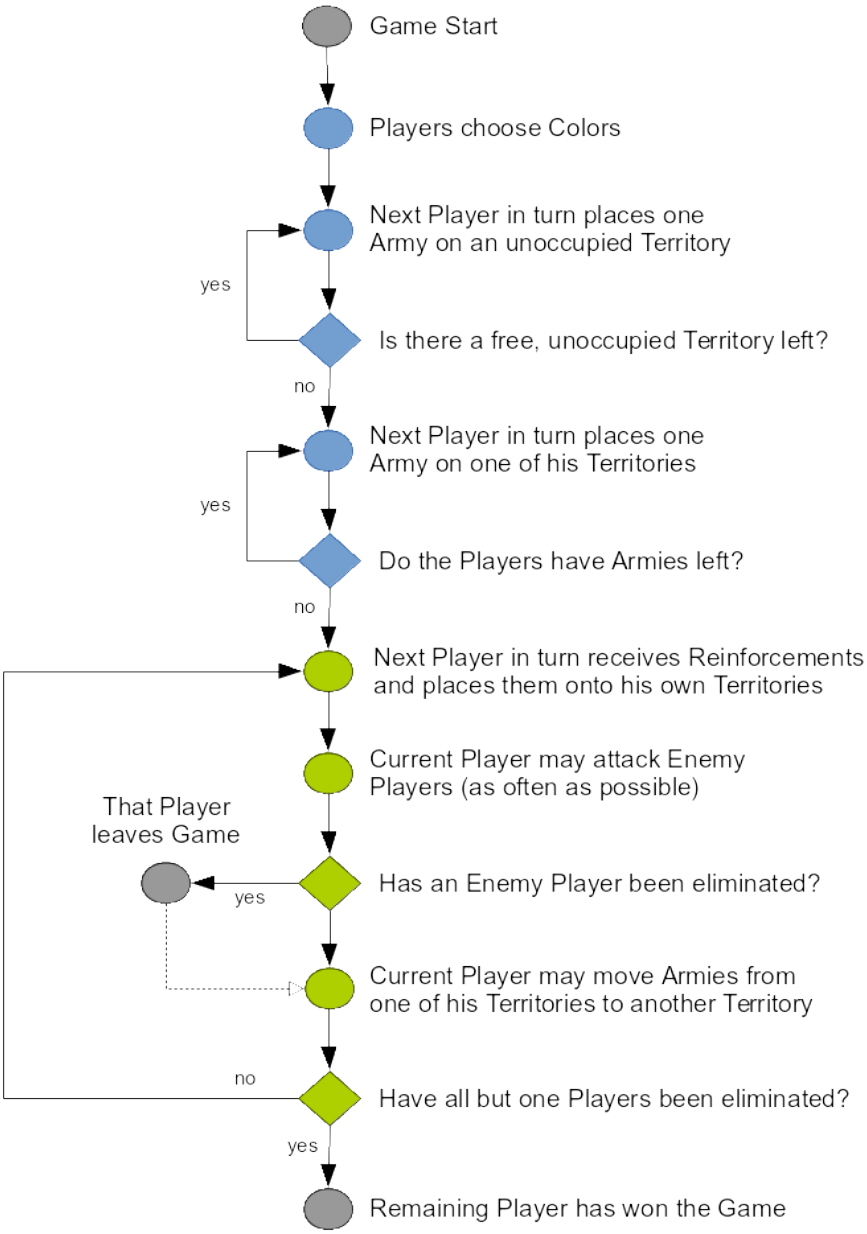


Figure 5.2 – RISK Rules and Game Flow.

place their units in turn, but only on to territories they have already occupied. The setup stage ends as soon as all players have distributed all their units.

An alternative way of playing the setup stage in RISK is to shuffle all territory cards and distribute them randomly among the players. Now each player places his units on to all those territories found on the territory map. After all units have been placed, the territory cards are shuffled again and placed face-down next to the game board.

In both variations of the setup stage one additional rule applies: at the end of the setup stage no territory may be unoccupied. This is true for the remainder of the game as well. At any point during game play each territory must at least contain one unit, while at no point during game play may more than one player occupy the same territory.

5.2.3 Game Stage

After the setup stage the game stage begins. This main stage lasts until the game is over, i.e. all but one of the players have been eliminated. During the game stage each player that has not been eliminated from the game yet plays in turn. Each such turn is split into four phases.

Phase 1: Trading Cards

The players start with no territory cards in their hands. Every time a player successfully conquers one or more territories during his attack phase (see below), he receives one territory card from the stack. There are three types of territory cards in the game: Infantry, Artillery and Cavalry. Each card belongs to exactly one type, while two extra Joker or wild cards may be treated by the player as any one of these three types.

When a player is in possession of three cards of the same type, or three cards of all different types, he may trade in that card combination during the trading phase. That player may also choose to wait for another round, but as soon as he has five cards, he must trade three of his cards in his next trading phase.

When a player trades in a territory card combination, he receives additional reinforcement units. An important element in the RISK game is the fact that each time a card combination is being traded by a player, he receives more extra units as from the last trade, regardless of whether is was the same or another player. The first trade is worth four units, the next trade six units, the third: eight units, the fourth: ten units, the fifth: twelve units, the sixth: 15 units, and for every additional trade thereafter five more armies than the previous set turned in. The reinforcement units gained by a player that way then have to be placed on to his own territories, as described in the next sub section – phase 2.

If the trading player owns one or more territories depicted on the cards traded in, he may choose one of these territories and place two extra units on to it.

Phase 2: Placing new Units

At each round during phase 2 a player receives new units to be placed on to the territories he owns. The number of reinforcement units a player receives depends on three factors:

1. **Card Trading**: As described in the previous subsection, a player may trade in cards he owns for additional units.
2. **Territories Occupied**: For each three territories a player owns, he receives one extra unit, with a minimum of three units for five or less occupied territories.

3. **Continents Occupied**: If a player controls a full continent, i.e. all territories in it, he receives additional units each round. Depending on the size of the continent and he number of countries it may be attacked from by other players, the number of extra units varies amongst the continents. The number of extra units gained for each continent is listed in table 5.1.

Continent	**Reinforcements**
South America	2
Australia	2
Africa	3
Europe	5
North America	5
Asia	7

Table 5.1 – RISK Reinforcements

Phase 3: Attacking

During phase 3 a player may attack another player's territory, if he chooses to do so. In such a case one territory attacks one other territory controlled by an enemy player. The attacking player may only do so if the attacking territory holds two or more units, and if the defending territory is adjacent, i.e. a border territory, or connected to the attacking territory by a sea line (as marked on the game board).

Each attack comprises a sequence of one or more rounds. The attack lasts until a) the defending territory is defeated, i.e. no more of the enemy player's units are left, or b) in the attacking territory only one unit is left, or c) the attacking player chooses to end the attack before either a) or b) happens. The outcome of the attack is determined by the attacking and defending players both rolling dice. The number of dice being rolled depends on how many units the attacker chooses for this round, and how many units the defender has in the defending territory, respectively.

If the attacker defeats all units in the defending territory, he then owns that territory. In that case the attacking player must move all units involved during the last attack round into the new territory. He may choose to move more units as well, with the rule that at least one unit must remain in the attacking territory.

Note that if the defending player loses his last territory that way, he is eliminated from the game.

Phase 4: Fortifying

After the attack phase a player may choose to move units from one of his territories to another of his territories. In that case similar rules as during an attack apply:

- The player may only move units between adjacent territories, or territories connected via sea lines.
- After the move the territory that has sent units to another territory must contain at least one unit.

This movement of units from one territory into another is only allowed once during a player's turn. After phase 4 for the current player is done, the next player in turn starts over in phase 1 again.

5.3 RISK Game Complexity

In the case of (board) games used as test environments for artificial intelligence, the game's complexity is of particular interest. That is because, if the game allows for only a few different possible game states, then we do not need much "intelligence" to play it. A simple game like *tic-tac-toe* ("OXO") for example can easily be played with only very few hard-coded rules. To master a complex game like Chess on the other hand, using simple rules will not lead to an artificial agent being able to beat expert human players. Because of that for a researcher dealing with artificial intelligence for (board) games the game complexity is an important factor for evaluating an AI algorithm's performance.

The complexity of a traditional board game is often measured via the so-called *State-Space Complexity* (SSC; Ghory, 2004). For the game RISK this complexity has been calculated by Wolf (2005). However, that evaluation does not provide much detail on how the author reached this complexity analysis. For the work presented in this thesis this section will therefore describe how the SSC for the game RISK can be calculated from scratch.

5.3.1 State-Space Complexity

One feature of many traditional board games is the total number of all possible states the game world can take. For example in tic-tac-toe there are exactly 765 possible combinations of X's and O's placed on the game board.[1]

For RISK Wolf (2005) proves that the game's SSC value is (countably) infinite. The reason for this is simple: there is no theoretical upper-limit of units in the game. Each turn each player receives at least three new armies, while attacks are optional. In other words: if no player attacks, more and more units would be placed on to the board, and the game would last forever. However, in practice this case cannot come true. In order to calculate the complexity for RISK we thus need to assume an upper limit for the total number of armies on the game board. For this analysis we set this upper limit to 200 units.[2]

In order to calculate the total number of possible world states when playing with (up to) 200 armies, we first need to calculate how these armies can be distributed over the territories. We use

[1] `http://www.mathrec.org/old/2002jan/solutions.html` (accessed 12.06.2016)
[2] Which seems to be a reasonable assumption based on our own experience with the game RISK.

the game's default 42 as number of territories. To count how these armies can be distributed over territories, we need a *combinatorial identity*.

Described by Feller (1950) this is a typical counting problem.[3] There are n indistinguishable objects (armies) to be distributed over k distinguishable bins (territories), such that all bins contain at least one object. Feller proves that for any pair of positive integers n and k, the number of k-tuples of positive integers whose sum is n is given by the binomial coefficient $\binom{n-1}{k-1}$.

In our example of RISK, with $A = 200$ armies and $T = 42$ territories, where each territory must at least contain one army, we end up with

$$\binom{A-1}{T-1} \tag{5.1}$$

ways to distribute the armies among the territories.

We also need to calculate the distribution of territories amongst the players. We will assume $P = 6$, as this is the maximum number of players allowed in the game. Every territory is different, so we treat them as different objects. Now we take each territory, one by one, and assign one of the players from P to it. Doing this for each territory we end up with P^T possible territory distributions.

The last step is to put these two distributions together. Note that for the army distribution we have defined an upper limit of 200. But since this is a *limit*, in our calculations we need to include 200 armies, as well as 199 armies, as well as 198, and so on, down to 42 armies (since we have 42 non-empty territories). We thus need to sum these army distributions and multiply that sum with the territory distribution among the players:

$$SSC(T, M, P) = \sum_{A=T}^{M} ArmyDistribution \times TerritoryDistribution \tag{5.2}$$

$$= \sum_{A=T}^{M} \binom{A-1}{T-1} \times P^T \tag{5.3}$$

So in our example where $T = 42$, $M = 200$ and $P = 6$ we end up with the following equation:

$$SSC(T, M, P) = \sum_{A=42}^{200} \binom{A-1}{41} \times 6^{42} \tag{5.4}$$

To calculate the Army Distribution part of this equation we can make use of the so-called *Hockey-Stick Identity*:[4]

$$\sum_{A=42}^{200} \binom{A-1}{41} = \sum_{n=41}^{199} \binom{n}{41} = \binom{199+1}{41+1} = \binom{200}{42} \tag{5.5}$$

[3] This is sometimes referred to as *stars and bars* problem.

[4] `http://www.artofproblemsolving.com/wiki/index.php/Combinatorial_identity` (accessed 10.07.2016)

We therefore end up with the following equation for the SSC of RISK:

$$SSC(T = 42, M = 200, P = 6) = \binom{200}{42} \times 6^{42} \approx 1.46 \times 10^{76} \tag{5.6}$$

We can thus approximate the State-Space Complexity of RISK with an upper bound of 200 units and 6 players to 1.46×10^{76}. For a comparison: according to Shannon (1950) Chess has an SSC of roughly 10^{43} (which is known as the *Shannon Number*).

5.4 Research on RISK

RISK has been subject to various research in the past. One work that focused on *Dynamic Planning and Execution* (DP&E) is presented in (Vaccaro and Guest, 2004). According to that paper, because of the large State-Space Complexity in RISK, traditional tree search algorithms may not be applicable. The paper proposes an optimised three-step algorithm to tackle this problem:

1. Bayesian Guided Evolutionary Computation (BGEC): A Bayesian Network autonomously prunes traditional evolutionary computation outcomes, by chaining individual state transition probabilities.
2. Temporally Aware Planning Termination (TAPT): Balances the computational resources needed to develop a plan with the needs of the user and the quality of the plan. This is achieved by generating one evolutionary computation generation per RISK move.
3. Success Moderated Execution (SME): Considers only the successful outcomes of the Bayesian calculations for each independent campaign. A campaign is by definition a connected series of operations designed to achieve a particular result.

According to the authors of that paper the proposed approach can be used for a range of various application areas, including battle planning and wargaming, autonomous vehicle navigation, or investment portfolio management.

In another research paper (Vaccaro and Guest, 2006) the same authors formalise a new way to measure the progress in DP&E tasks. Here an evolutionary tournament strategy is presented that learns new parameters included in the fitness function.

In (Wolf, 2005) an intelligent autonomous RISK playing agent is described. The author's approach is twofold: First, a basic version of the autonomous agent is designed based on hand-crafted rules and game features. That agent evaluates the current and all possible next world states using a evaluation function which evaluates a given world state by utilising the game state features in a linear fashion. Based on this approach an enhanced agent is presented, which automatically sets the weights for each feature in the evaluation function based on *Temporal Difference Learning* (TDL). The enhanced version of the agent also makes use of goals and plans. We will have a closer look at the game state features proposed in (Wolf, 2005) in the next section of this thesis when we describe our approach of IR in the RISK game.

A mathematical analysis of various aspects of the RISK game is presented in (Blatt, 2002). For this analysis the author uses Markov chains and probability theory. Two key points in RISK are being evaluated: the probability of capturing a territory in the case of an attack, and the number of lost armies if one of your territories is attacked by an enemy player. The article provides calculated numbers for these two key aspects of the RISK game. The authors conclude that the key to answer these questions is the use of Markov chains to create a state-space model.

Another work is closely related to our research, focusing on plan recognition in RISK (Khan, 2013). The author describes how to apply PHATT based plan recognition on the game RISK, which has been presented by Geib and Goldman (2009) and discussed above in chapter 2. The work by Khan however has not been published, so that we are not able to discuss it here in detail.

5.5 The Application of our Approach in IR to RISK

This section will explain how we apply our approach to intention recognition to the RISK board game environment. It will define the ground literals used to describe the world state in RISK, how that world state of RISK can be evaluated and how this evaluation leads to world state indicators. Later in this section we will show and explain the indicator-goal matrix used for RISK, as well as the system architecture and design of our intention recognition prototype system "iRecognise".

5.5.1 RISK World State Ground Literals

The world of the RISK board game can quite easily be described in words. There are a number between 2 and 6 players. The game board consists of 6 continents, which in turn consist of 42 territories in total. Each territory belongs to exactly one player. Each territory holds at least one army/unit of the player it belongs to. In addition, each player can hold up to 5 country or territory cards. This is sufficient to describe the static view on the world state, that is, the world state at a given time.

For the context of intention recognition, however, we also need to know what is happening in the game world. Therefore a few more parts have to be added to the description of the world state. First, we want to know whether armies have been gained by a player this game round and how many armies, as well as how many armies have been lost. A player gains armies each turn automatically based on how many territories and continents he controls. Even more armies can be gained by trading in territory cards, as described above in the outline of the RISK game rules. On the other hand a player loses armies either while attacking other territories, or while defending against such attacks. Last but not least we also need to know whether a player has attacked another territory. Since we know already which territory belongs to which player, and whether it belonged to that player in the last round as well, we can easily determine whether such an attack was successful or not.

In our IR system we observe one player in the game in particular. We therefore need to put the changes in the world state into relation to that player. For example, if our player has lost armies in the last round, that could either be the result of his attack against another territory, so that he triggered the event, or because another player has attacked him. Thus each ground literal in our

design includes a boolean flag called "User-Modified" (*um-flag*). If it is true, then the observed player has initiated the change in this part of the world state. If it is false, another player has triggered the change.

This leads us to the following ground literals that we use in our case study on RISK:

- `card-1, um-flag (country<name|"null">, type<name|"null">)`
  `("null" if no card is in this slot)`
- `card-2, um-flag (country<name|"null">, type<name|"null">)`
- `card-3, um-flag (country<name|"null">, type<name|"null">)`
- `card-4, um-flag (country<name|"null">, type<name|"null">)`
- `card-5, um-flag (country<name|"null">, type<name|"null">)`
- `country-<name>, um-flag (player<name>, name<country-name>, armies<number>)`
- `...  (repeat once for each country)`
- `armies-gained, um-flag (armies<number>) (um-flag is always true)`
- `armies-lost, um-flag (armies<number>, country<country>)`
- `...  (repeat as often as armies have been lost)`
- `attacked-country, um-flag (attacker<country>, defender<country>,`
  `other-player<name>) (um-flag is always true)`
- `...  (repeat for each attacked country)`

5.5.2 Game State Evaluation

The next step in our approach is to evaluate this world state, i.e. identify crucial aspects that are relevant to intention recognition. As described above, this step can be supported by using game state evaluation functions found in the literature on this subject.

For our case study we have reviewed papers and articles on various aspects of RISK, some of which also describe their approach to RISK world state evaluation. Two papers, (Vaccaro and Guest, 2006) and (Wolf, 2005), are of particular interest to the work presented here. The evaluation of a RISK world state described in these two papers can be summarised with this list of important game state aspects:

- **Armies**: Number of player's armies in relation to total number of all armies in play
- **Best Enemy**: Power of the best enemy player (avg. relative army/territory strength)
- **Continent Safety**: Threat from enemy players against continents fully owned
- **Continent Threat**: Thread from current player against enemy continents

- **Distance to Frontier**: Army distribution throughout current player's territories
- **Enemy Estimated Reinforcements**: Total number of units in enemy reinforcements
- **Enemy Occupied Continents**: Number of continents occupied by enemy players
- **Hinterland**: Territories behind borders in relation to all owned territories
- **Maximum Threat**: Probability of current player being able to occupy an enemy territory
- **More than one Army**: Territories of current player with more than one army
- **Occupied Territories**: Territories owned by current player in relation to all territories
- **Own Estimated Reinforcements**: Expected total reinforcements received next turn
- **Own Occupied Continents**: Number of continents fully occupied by current player
- **Risk Cards**: Number of cards in the current player's hand
- **Continent Army Domination**: Own armies in target continent vs. total number
- **Continent Domination**: Relative power of current player in target continent $\times$ rating of continent

As shown in the next sub section, we will use only a selection of this list plus some features defined by ourselves for the IR system's world state indicators.

5.5.3 The Indicator-Goal Matrix for RISK

For the intention recognition system proposed in this thesis, based on the considerations outlined in this chapter so far, the following relevant world state indicators have been identified (in relation to the observed player, OP):

- **IV-01**: Total number of units owned by OP
- **IV-02**: Number of units the OP owns in continent X (for each continent)
- **IV-03**: Number of units the OP gained this round
- **IV-04**: Number of units the OP lost this round
- **IV-05**: Number of units in a continent border country (for each continent)
- **IV-06**: Total number of territories controlled by OP
- **IV-07**: Number of territories controlled in continent X (for each continent)
- **IV-08**: Number of territories the OP attacked this round
- **IV-09**: Number of territories the OP has won this round
- **IV-10**: Number of territories the OP has lost this round

- **IV-11**: Percentage of continent controlled by OP (for each continent)
- **IV-12**: Number of continents fully owned by OP
- **IV-13**: Maximum number of units in one single territory
- **IV-14**: Number of different enemies the OP has attacked this round
- **IV-15**: Number of new enemies attacked this round (i.e. not attacked last round)
- **IV-16**: Number of cards the OP has in his hand
- **IV-17**: Number of tradeable card pairs the OP has in his hand

Based on these WS indicators the following intentions in the context of RISK have been identified. These intentions have been selected based on experiments with experienced human players, as well as on related literature, for example (Vaccaro and Guest, 2004) and (Wolf, 2005):

- **I-01, Conquer one new Territory**: Every time a player successfully attacks and thus conquers an enemy-controlled territory, he can move his armies into that territory and gain control over it. Further, that player may also pick a new territory card, which can later be used to trade for more reinforcement units.
- **I-02, Occupy a whole Continent**: Controlling a full continent, i.e. controlling each territory in that continent, leads to more reinforcement units for that player. It can therefore be an intention to conquer each territory in a continent, for example if a player owns most of these territories already.
- **I-03, Conquer as many territories as possible**: Sometimes a player wants to conquer and control as many territories on the whole RISK game board as possible, for example to simply weaken the other players in general.
- **I-04, Fortress Border to Enemy Continent**: In RISK it is likely that a player who fully controls a continent wants to protect the border territories in that continent from attacks. Now both players, the one controlling the whole continent, as well as the other player controlling a neighbour territory, may want to stack more units on the border territories in order to protect them.
- **I-05, Maximise Number of Units in one Territory**: Some players follow a strategy of stacking as many units on one territory as possible, in order to start a broad attack on neighbour territories later on.
- **I-06, Wait for Card Trade**: Every trade of territory cards provides more reinforcement units than the previous trade(s). A player might therefore want to wait with his card trading activities, so that another player trades in his cards before him.
- **I-07, Occupy one Territory in Enemy Continent**: As described, owning a full continent provides more reinforcement units each turn to that player. Often it is therefore the intention of the other players to conquer at least one territory of that continent, so that the first player does not receive any extra reinforcement units for the continent.

- **I-08, Eliminate one Enemy Player**: There are two reasons for the intention to eliminate a player from the game. Either because the game is being played with mission cards and the mission card of one player says so, or simply because that player is already weak and close to being defeated anyway.

Having both, the list of world state indicators, as well as the selection of potential player intentions, we can thus define the following indicator-goal matrix that we use in our case study on RISK (tables 5.2 to 5.4). Note that "LR" refers to the value of that world state indicator in the last round.

Goal	IV-01	IV-02	IV-03	IV-04	IV-05	IV-06
I-01						LR - IV10 + 1
I-02		(LR - IV04) up				
I-03						(LR - IV10) up
I-04					up	LR - IV10
I-05	up					equal / down
I-06						
I-07					down	LR - IV10 + 1
I-08						

Table 5.2 – Indicator-Goal Matrix used in RISK, part 1.

Goal	IV-07	IV-08	IV-09	IV-10	IV-11	IV-12
I-01		1+	1			
I-02	up	1+ AND $\geq$ IV09			up AND $> 50\%$	equal / up
I-03		3+ AND $\geq$ IV09				
I-04						
I-05		0	0			
I-06						
I-07		1+	1+		up AND LR = 0	
I-08		*special**)				

Table 5.3 – Indicator-Goal Matrix used in RISK, part 2.

*) *Value for IV-08 in I-08 = max. attacks against one enemy / total number of attacks*

5.5.4 System Architecture

Figure 5.3 illustrates the overall iRecognise system architecture.

Goal	IV-13	IV-14	IV-15	IV-16	IV-17
I-01		1		up	equal / up
I-02					
I-03	down	1+			
I-04	up				
I-05	up	0			
I-06				up OR 5 (=max)	1
I-07		1			
I-08		1	0		

Table 5.4 – Indicator-Goal Matrix used in RISK, part 3.

5.5.5 System Components

Monitored Game (MG)

In order to let our IR system recognise the user's intentions we need to attach it to a game that is then actively monitored. In theory our model should allow to monitor a wide range of different types/genres of games. For simplicity reasons however we start by monitoring a relative simple, turn-based strategy game that allows the player to plan several steps in advance.

Abstract Game Interface (AGI)

The AGI is responsible for retrieving world state information on the MG at each time step (e.g. a turn in the game). The main purpose of the AGI is to abstract the game mechanics from the IR system's perspective. Thus, the IR system itself should not bother which game is currently being monitored.

- In order to connect the AGI to the MG one needs to write an AGI plug-in for that game. Obviously this plug-in is game- and therefore domain-dependent, and thus cannot be specified in advance. The MG itself needs to support such a plug-in, as mentioned above (e.g. due to an API). To abstract the game mechanics from the main system's perspective each AGI plug-in must implement a specific class interface. This allows the IR system to receive world state information from each MG in the same manner (i.e. data structures).
- The other end of the AGI is connected to the IR system itself, it does not need any game-related plug-ins. As described below, the IR system listens for new game state information from the AGI, so that the AGI is therefore responsible to actively forward game state information at each time step.

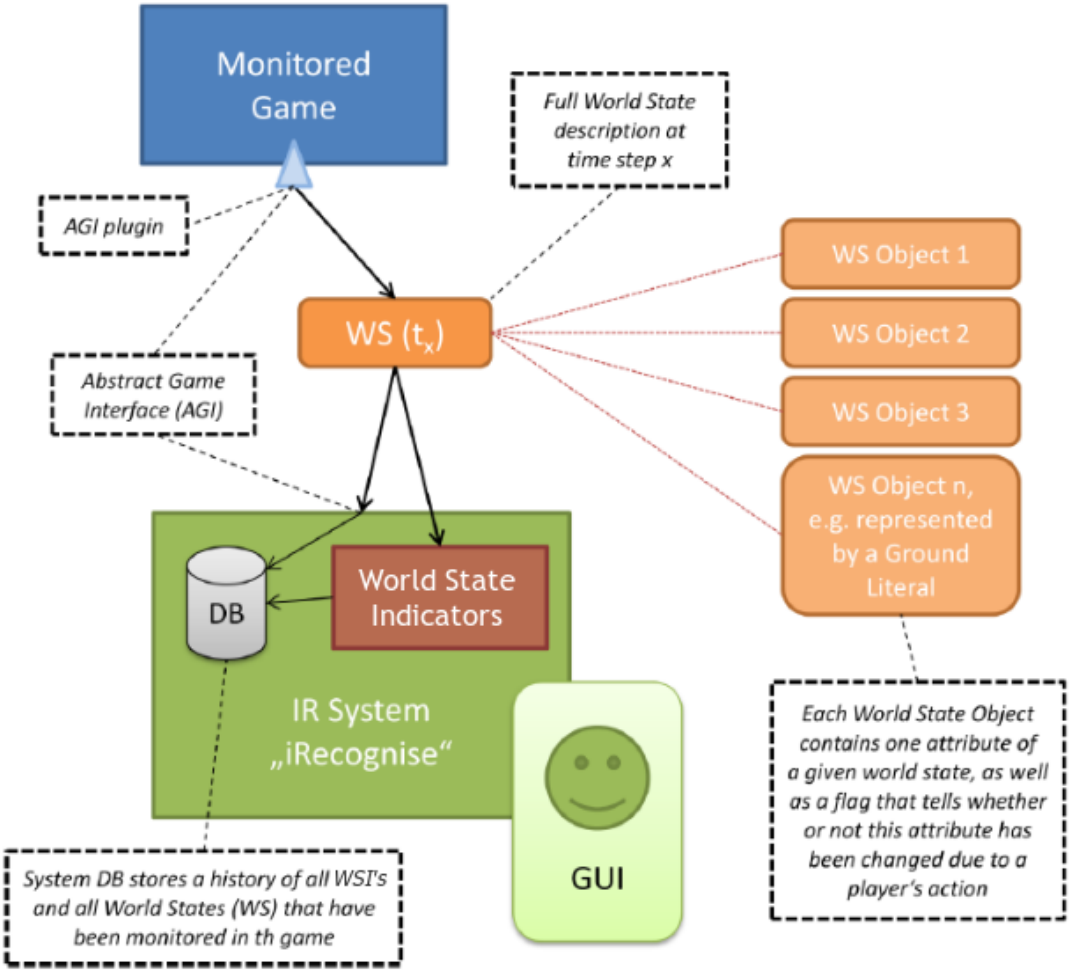


Figure 5.3 – iRecognise System Design

World State (WS)

At each time step the AGI sends the current game world state information to the IR system. An example of a time step could be one turn/round in the MG. The information that sums up the world state is game-dependent, thus has to be defined a priori by the system designer. However, the format in which this information is expressed in is formal and abstract, so that the IR system can process it properly. The WS at each time step comprises of several world state objects, as described in the next section.

World State Objects (WSO)

The WS at each time step comprises of (theoretically) one to several items of information, i.e. attributes, describing the current state of the game world. In our system model each such attribute is expressed as a World State Object (WSO). For our first prototype a WSO is built of an instantiated ground literal. Note, however, that as we want to be able to use also other representations, these attributes are abstracted into WSO. Technically the WSO provides a transformation operation to convert from the inner representation (e.g. the ground literal) to an indicator value, as described

below. Moreover, each WSO also contains a flag ("um-flag") that tells whether or not this specific attribute has been changed since last time step due to a player's action.

World State Indicators (WSI)

One of the central parts in our IR system design are WSI's. These are several real-numerical values describing certain attributes of the game. As with the WS attributes these WSI are game-dependent and therefore to be defined a priori by the system designer. Each WSI is represented as a real-numeric value that changes over the course of a game session. The current values of each WSI are set from the WS that is received from the AGI. As mentioned above, each WSO contains a transform operation to retrieve the value of a certain game state attribute, which in turn leads to a certain WSI value.

System Database (SDB)

For the task of intention recognition we are not only interested in the current value of all WSI's, but also their progression over time, i.e. the course of the running game session. We therefore also need to save all historic values of each WSI into the SDB. In addition, the SDB also saves each WS object per time step. The database itself is implemented as an SQL-compatible system, in order to maintain flexibility when it comes to data management and processing. For this task we are considering two alternatives:

- A traditional MySQL or SQLite database, that is free of charge (Open Source) and is fast, stable, and mature. These have been widely used for a number of years and therefore are a de-facto standard when considering relational database system (RDBS). Moreover, the SQL database system remains stable and reliable even with large data sets, as may be required for our system setup.
- An in-memory SQL database does not store the data permanently on to the disk, but instead directly into the system memory (RAM). This has the major advantage that access to the DB's data is very fast. However, system memory is often much more limited than disk capacity, so that an in-memory database system might not be an option in our setup.

Note that as described below for our experiments we have decided to rely on an SQLite database.[5]

Graphical User Interface (GUI)

The GUI that is provided as part of our IR system allows the user to interact with the system. It provides an easy way to initialise and configure each relevant part of the setup:

- The game to be monitored (MG)

[5]Because SQLite is natively supported in Python, which we use as main programming language in our experimental setup.

- The interface between the main system and the MG (AGI and game plug-in)
- The format of the game world state (WS and WSO)
- The definition of all intentions to monitor in the game (WSI)
- The layout of all SDB tables based on the WSO and WSI

The next chapter will describe how we used this setup to conduct our experiments with iRecognise and RISK, as well as their results.

Chapter 6

Experiments and Evaluations

After the theoretical discussion of our approach to intention recognition in games, and the description of the case study (the board game RISK), this chapter now describes how we have evaluated the performance of the proposed approach. For this evaluation we have conducted a range of practical experiments, which will be described and their results analysed next.

For the experiments we have followed two main approaches: to test the system in an automated manner using an artificial RISK player bot, as well as to evaluate the performance of the system based on game sessions with real human players. But before we describe the experiments and their results, the next section first outlines the experimentation setup and environment.

6.1 Experimentation Environment

To evaluate the performance of our approach to IR and the iRecognise prototype system, all experiments have been performed within the same environment. This setup is split into three main parts: the RISK game itself, the iRecognise IR system, and the IR system database.

6.1.1 RISK Game Software: Domination

The software used for our experiments is called *Domination* and is available for free from the Sourceforge hosting platform.[1] This software is a direct clone of the original RISK board game, but has been renamed due to licensing issues. It is written in Java and available as Open Source software under the GPL license.

Using the Domination game software has two advantages in the context of our work. First, since it is Open Source, Domination can be used and modified free of charge without licensing costs. Second, it includes an API for artificial player bots, which is used for the automated experiments, as described below.

[1] `http://domination.sourceforge.net/` (accessed 10.07.2016)

For our experiments no game logic in Domination has been modified. The only changes made to the official repository sources are to enable iRecognise to connect to the running game, as described in the previous chapter (refer to the *Abstract Game Interface*, AGI). Technically, these changes let the Domination software collect the world state data and send it to the iRecognise system each turn in the game. The connection between Domination and iRecognise is achieved via a standard TCP socket. A Domination sample screen is shown in figure 6.1.

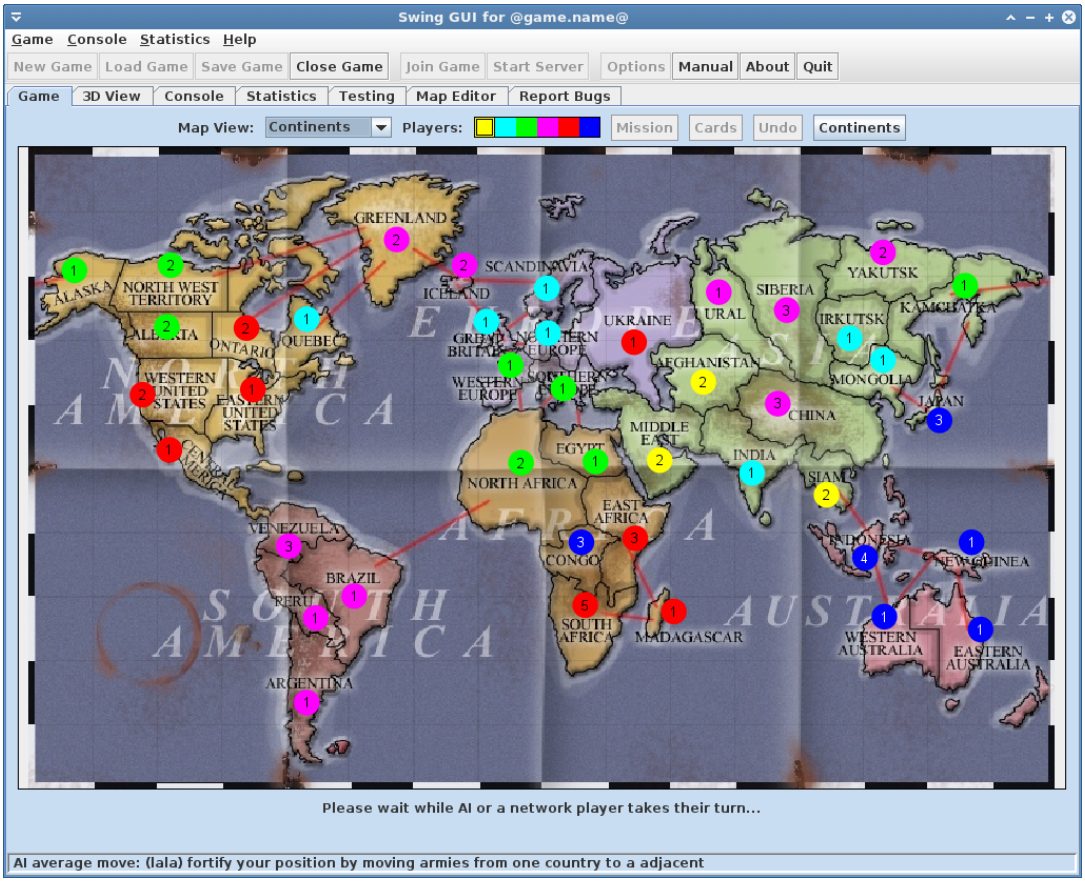


Figure 6.1 – Domination Sample Screen, Game View.

6.1.2 Intention Recognition System: iRecognise

The iRecognise prototype implementation has been developed in Python, using the Qt libraries for GUI presentation. It is designed as outlined in the previous chapter. Before it launches the Domination game, the iRecognise system opens a TCP server port to which the AGI within Domination later connects to in order to transmit world state data, i.e. ground literals.

At every turn in the game the AGI in Domination sends the current world state data to iRecognise. This data is received via the TCP socket and stored into the system database, as described in the next section. Additionally, the received world state data is processed by the world state indicators $\rightarrow$ player intention rule set, which has been hard-coded into iRecognise. Later, these hard-coded rules could be replaced by a dynamic sub-system, e.g. an artificial neural network, as outlined before. By parsing the world state data and processing them according this rule set the probabilities of each intention is constantly updated and shown to the user in the GUI.

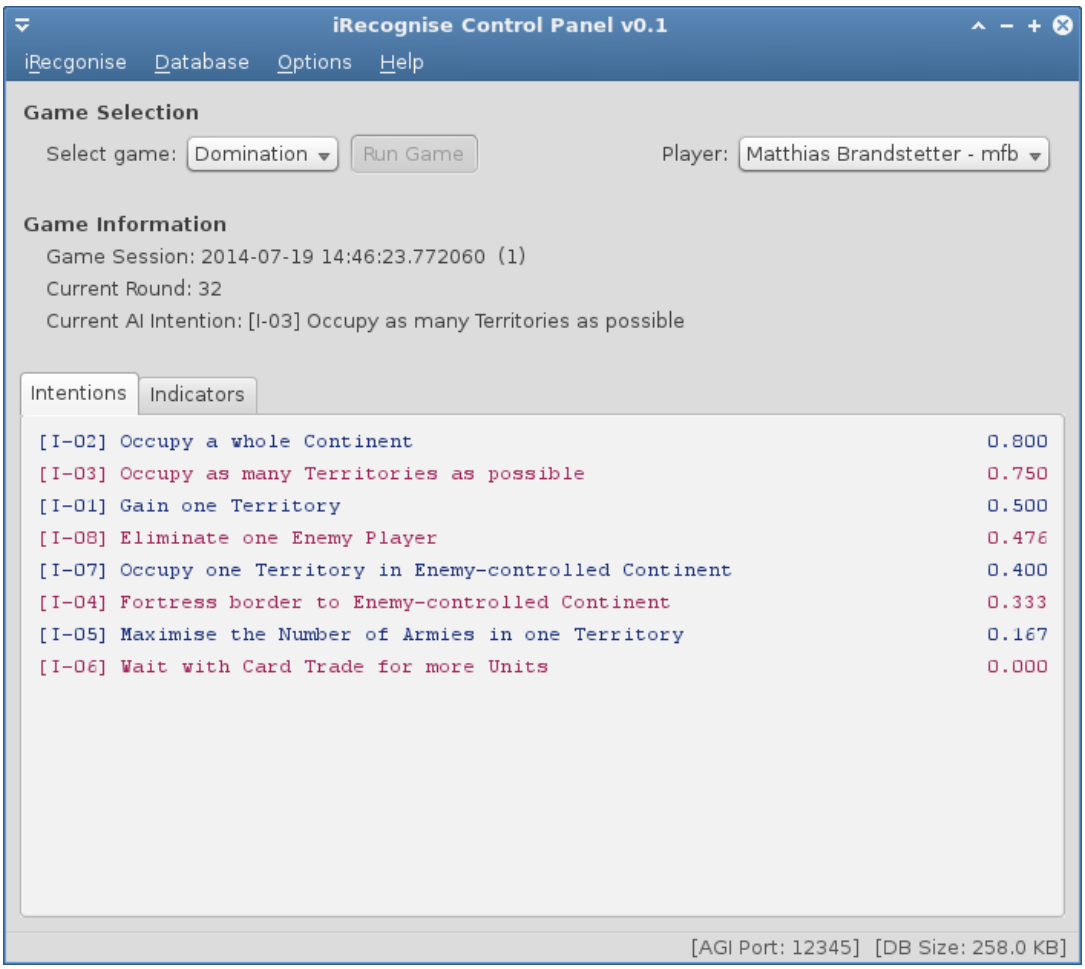


Figure 6.2 – iRecognise Sample Screen (Intentions)

6.1.3 System Database: Sqlite

World state data received by the iRecognise system is not only used to update the intention probabilities, but also saved into the system database. This has two advantages: first the IR system can make use of these historic values, even if they are not present in the system memory any more. And second, this allows us to analyse this data later on, as with a log file.

The iRecognise prototype system uses an SQLite database to store the historic world state data. This has the advantage that no additional database system has to be running in parallel, as SQLite stores all data in a single file. Moreover, as iRecognise has been developed in Python and SQLite support is built into Python by default, no additional drivers are needed. For easier access to and management of the database tables iRecognise uses the *Peewee*[2] database library (Object-Relational Mapper, ORM).

Refer to tables 7.1 to 7.4 (Appendix B) for the exact layout of the iRecognise database tables used to store the (world state) data.

[2] `http://peewee.readthedocs.org/en/latest/` (accessed 10.07.2016)

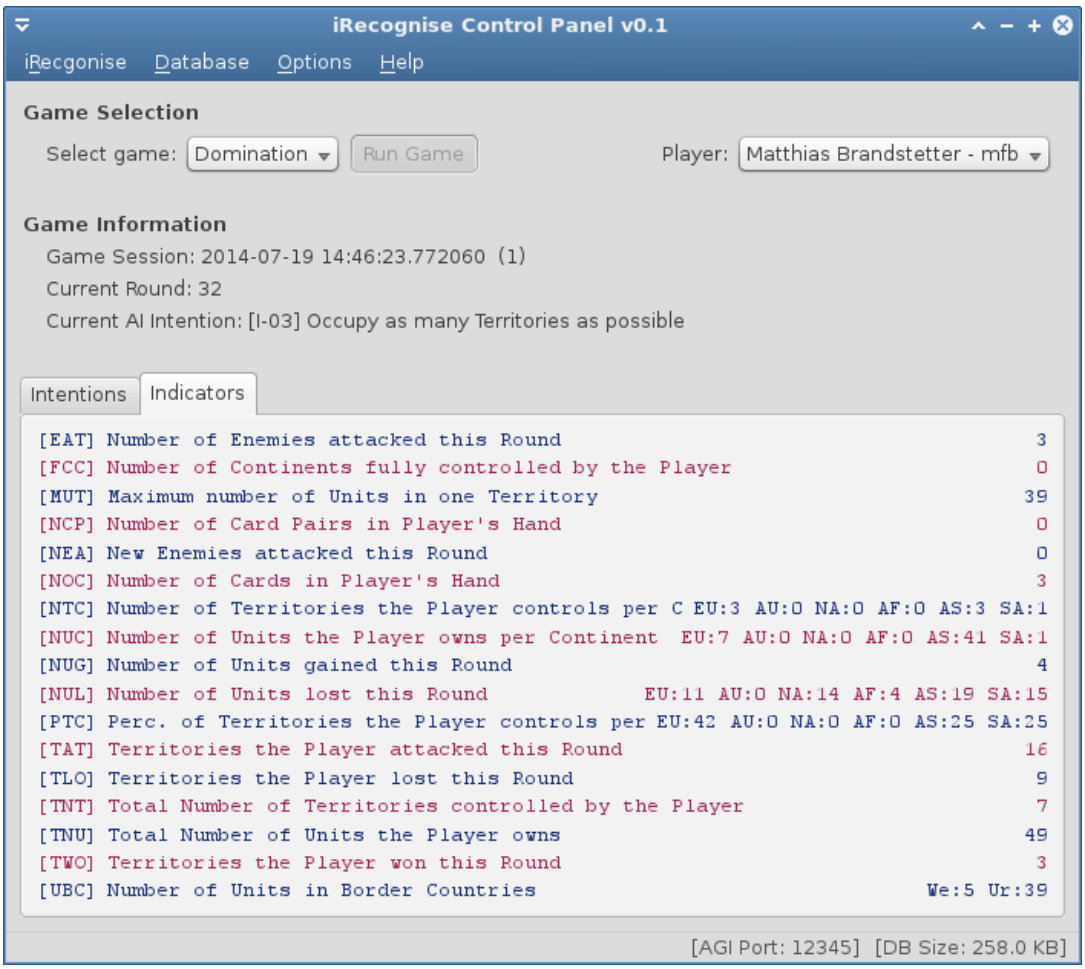


Figure 6.3 – iRecognise Sample Screen (Indicators)

6.2 Automated RISK Player Bot

This section describes the experiments that have been conducted using an automated RISK player bot. The bot has been developed from scratch for the specific purpose of iRecognise performance evaluation. It is written in Java and makes use of the Domination API for AI bots.

The RISK bot implements playing logic for all player intentions used in our experiments (listed above in section 5.5.3). It can either be set to follow one intention for a whole game session, or randomly change between these intentions within one session. In the latter case, the current intention stays selected for a small, random number of game turns, until a new intention is selected. If the goal of the newly selected intention is currently not achievable, another intention is chosen.

The main advantage of using an automated RISK bot is that experiments can be run as often as desired in a relatively short amount of time, which is not possible with real-world human players. Moreover, this type of bot allows us to easily re-run certain experiments, if needed. The downside of an automated bot on the other hand is that it only plays according pre-defined playing rules, as implemented a priori into the bot. This limits the capabilities of the bot, in particular in comparison to human players. To overcome this limitation one could improve the performance of the automated bot by implementing more advanced AI techniques, for example a sophisticated

planner such as a Hierarchical Task Network Planner. For the work presented in this thesis these bot improvements were not included in the scope.

The following sub sections explain in detail the experiments which have been undertaken using the automated RISK bot.

6.2.1 Experiment 1: One Intention per Game Session

In the first experiment the RISK bot plays according the logic of one particular intention per game session. The intention to use is set in advance when a sample game starts. For each intention ten games have been played. Table 6.1 lists the average results of these sample games for each of the defined player intentions. Note that for technical reasons intention I-06 has not been evaluated in this experiment.[3]

Intention	WS Count	Exact Hit	Top2	Top3
I-01	9	0,36	0,54	0,74
I-02	9	0,51	0,72	0,76
I-03	9	0,13	0,39	0,41
I-04	11	0,45	0,63	0,91
I-05	12	0,73	1,00	1,00
I-07	9	0,03	0,37	0,64
I-08	8	0,28	0,56	0,77

Table 6.1 – Experiment 1: Average successful recognition per intention.

Note that in this and the following tables the columns are:

- **Intention** – The (pre-defined) intention that has been tested
- **WS Count** – World States considered before Game Over
- **Exact Hit** – How often this intention has been correctly identified by iRecognise
- **Top2** – How often iRecognise has identified this intention as one of the two most probable intentions (i.e. the intention has been ranked either the most or second-most likely intention)
- **Top3** – How often iRecognise has identified this intention as one of the three most probable intentions (as before, but including even the third-most likely intention)

6.2.2 Experiment 2: Random Intentions per Game Session

The second experiment makes the task of intention recognition a bit more difficult, as the bot's intention randomly changes within each one game session. The logic of the current intention is

[3]Because our RISK bot does never trade-in any cards.

played for a few game rounds, then another intention is randomly chosen. Not every intention is applicable in any given situation of a RISK game, e.g. because a goal cannot be achieved. That can happen for example if the intention would be to protect the borders of a fully controlled continent, but the bot does not fully control any continent at all. In such a case another, suitable intention is selected. For this experiment ten sample games were played. As in the first experiment intention I-06 has not been evaluated in this experiment. Results can be found in table 6.2.

Session	**WS Count**	**Exact Hit**	**Top2**	**Top3**
Game 1	9	0,22	0,67	0,78
Game 2	8	0,25	1,00	1,00
Game 3	11	0,27	0,64	0,73
Game 4	9	0,11	0,56	0,67
Game 5	15	0,20	0,47	0,53
Game 7	8	0,25	0,38	0,38
Game 8	10	0,70	0,90	0,90
Game 9	8	0,62	0,75	1,00
Game 10	10	0,30	0,60	0,80
Average	10	0,31	0,64	0,77

Table 6.2 – Experiment 2: Successful recognition of randomly chosen intentions.

6.3 Human RISK Players

This section outlines the experiments that have been undertaken with real human players and our IR system prototype for the RISK game.

6.3.1 Experiment 3: Performance Evaluation per Player

Like with the automated AI bot used in experiments 1 and 2 we had some human players play sample games of RISK, while being observed by the iRecognise system. Results of this experiment can be found in tables 6.3 to 6.5.

6.3.2 Experiment 4: Performance Evaluation per Intention

This experiment evaluates the performance of the prototype IR system in recognising each of the intentions pre-defined for RISK, as listed above. Note that each of the sample players follow these intentions in different ways and at a different intensity, so that there is an unequal distribution of experimentation data amongst these intentions. Results of this experiment are shown in tables 6.6 to 6.8.

Session	WS Count	Exact Hit	Top2	Top3
Game 1	19	0,42	0,68	0,79
Game 2	14	0,29	0,64	0,71
Game 3	28	0,32	0,61	0,71
Game 4	7	0,43	0,71	0,86
Game 5	13	0,46	0,77	0,85
Average	16	0,38	0,68	0,78

Table 6.3 – Experiment 3a: Intention Recognition results in RISK with player 1.

Session	WS Count	Exact Hit	Top2	Top3
Game 1	8	0,00	0,25	0,50
Game 2	6	0,17	0,33	0,50
Game 3	10	0,60	0,70	0,90
Game 4	7	0,14	0,29	0,29
Game 5	14	0,21	0,50	0,79
Average	9	0,22	0,41	0,59

Table 6.4 – Experiment 3b: Intention recognition results in RISK with player 2.

Session	WS Count	Exact Hit	Top2	Top3
Game 1	15	0,27	0,60	0,93
Game 2	12	0,08	0,42	0,67
Game 3	9	0,22	0,44	0,78
Game 4	19	0,32	0,63	0,84
Game 5	11	0,36	0,55	0,64
Average	13	0,25	0,53	0,77

Table 6.5 – Experiment 3c: Intention recognition results in RISK with player 3.

6.4 Experiment Result Evaluation

We will now evaluate the experiments and their results as shown in the previous section. This evaluation critically analyses the performance of the iRecognise prototype (and therefore the proposed intention recognition approach), and provides suggestions on how performance could be improved in successive iterations.

Player	I-01	I-02	I-03	I-04	I-05	I-06	I-07	I-08
Player 1	0,67	0,33	0,00	0,27	0,73	0,00	0,00	0,14
Player 2	-	0,18	0,00	-	0,69	0,00	-	0,00
Player 3	0,50	0,07	0,21	-	0,56	0,00	-	0,31
Average	0,59	0,20	0,07	0,27	0,66	0,00	0,00	0,15

Table 6.6 – Experiment 4a: Exact recognition of intentions per player.

Player	I-01	I-02	I-03	I-04	I-05	I-06	I-07	I-08
Player 1	0,89	0,67	0,00	0,73	0,73	0,44	0,60	0,43
Player 2	-	0,45	0,25	-	0,92	0,00	-	0,15
Player 3	0,88	0,27	0,57	-	1,00	0,00	-	0,62
Average	0,89	0,46	0,27	0,73	0,88	0,15	0,60	0,40

Table 6.7 – Experiment 4b: Top-2 recognition of intentions per player.

6.4.1 General Performance Evaluation

We are not aware of any peer-reviewed, published research on intention recognition in the context of RISK, so that the performance of our approach cannot be compared to any existing approaches. Based on the results of the experiments shown above it is apparent that iRecognise is not able to exactly recognise the user's (or bot's) intention all the time. This fact on its own is not surprising though, as (to our knowledge) no existing approach to intention recognition is able to recognise intentions in 100% of all cases within such a complex domain as RISK or a comparable (board) game. In particular in the context of real human players this is a logical conclusion, as even the most experienced human players are not able to correctly guess every other player's intentions all the time.

In our experiments we have therefore also checked whether iRecognise is able to correctly recognise the player intention within the top 2 or top 3 of the most probable intentions. If we also consider it as a correct match when iRecognise recognises the player intention within the top 3 most likely intentions, the correct hit rate dramatically increases. In the context of real human players this is also logical, as, from our personal experience we noticed that during game play we consider the two or three most probable intentions of our opponent players, not only the single most probable one.

In general the results of the experiments listed above prove that our novel approach of IR using world state indicators works. However, we wish to state that, even though the results of our experiments are based on a prototype version, by evaluating the experiment results, it becomes apparent that there is room for improvement. We will discuss possible improvements in this context in the following sections.

There is also a large variation in the recognition performance of different intentions. For example,

Player	I-01	I-02	I-03	I-04	I-05	I-06	I-07	I-08
Player 1	0,89	0,83	0,00	0,82	0,91	0,56	0,80	0,43
Player 2	-	0,73	0,50	-	0,92	0,00	-	0,54
Player 3	1,00	0,67	0,86	-	1,00	0,40	-	0,77
Average	0,95	0,74	0,45	0,82	0,94	0,32	0,80	0,58

Table 6.8 – Experiment 4c: Top-3 recognition of intentions per player.

in all experiments the correct recognition rate is considerably better for intentions I-01 or I-05 than for intentions I-03 or I-06. This has a number of causes: On the one hand it may be necessary to improve the rules in the indicator-goal matrix for those intentions that have not been fully identified. Since in the current approach this rules definition is carried out manually by a domain expert, improving on it would be a time-consuming task. To address this issue we therefore propose an automated generation of these rules, for example via an artificial neural network, as outlined in section 4.2.2.

On the other hand intentions involve different world state indicators, and they are generally of a different complexity. It is therefore likely that their recognition rate depends on their complexity. The more aspects of the world state have to be considered for an intention, the harder it will be to correctly recognise that intention in general.

Regardless of the rules complexity, an improvement in our approach could be to employ a human expert to train the system to recognise correct guesses, while discarding incorrect ones, as it is done in reinforced learning techniques. That approach could also take the history of player's intentions into account. For example, if the system becomes aware that a player invariably follows intention B after intention A, then this is also likely for future game plays of that player. That improvement could be implemented with a Bayesian network for example, as outlined in section 2.2.

The selection of intentions for the RISK game used in our experiments is hand-picked based on the personal experience of the authors, as well as on the evaluation of the RISK game found in literature and other research. However, different players will have different intentions in mind when playing this game. We would therefore recommend that future research on our approach should allow for custom intentions per player.

6.4.2 Automated RISK Player Bot

The automated RISK bot proved to be successful not only in experiments, but also during development, as it assisted us in the evaluation of the performance of the system without the bottleneck of waiting for human players. It is apparent though, from our experiments, that an automated bot cannot fully replace human players for several reasons.

It is interesting that the performance of iRecognise is roughly equal in both experiments 1 and 2. This is unsurprising, as our approach currently does not take many historic values into account, except the last round in some of the rules, as shown in the indicator-goal matrix. In other words:

iRecognise currently does not care what the player did so far in this game, it simply calculates the probability of each intention "from scratch" in every game round. As described in section 4.2 on the theoretical discussion of our proposed approach, we believe that the overall performance of the system can be improved by also taking the history of the observed player into account. That would in turn presumably lead to better results in experiment 2 in comparison to experiment 1.

In general, however, in some cases it is difficult to use only one intention throughout a full game play. This is because, in these cases, i.e. certain world states, it is not possible to aim for a specific intention at all. For example, trying to occupy one territory in an enemy-controlled continent (I-07) would not be possible if no enemy player fully controls a continent in the first place. In such a case it does not make sense for the automated bot to follow I-07. Such cases are checked by the bot in experiment 2, as it makes use only of those intentions that are valid in the context of a given world state. Moreover, it was observed that, using only one single intention throughout a full game play often leads to a quick defeat of the player. It seems that variations in the playing style, thus changing between goals over the course of a game session, lead to better results in general.

The intentions playing logic incorporated into the automated RISK bot is hard-coded and quite simple. It strictly follows each of the pre-defined intentions, without variations. The performance of the automated bot in game play is therefore far from an experienced human player (it is however comparable to a beginner human player). The performance of the bot could be improved by designing and implementing a more sophisticated planning algorithm. One such improvement could be the use of a *Hierarchical Task Network Planner*, as presented in (Hoang et al., 2005).

6.4.3 Human RISK Players

When considering the results of experiment 3 it becomes apparent that intention recognition with iRecognise works best for player 1, which was the player who defined the original indicator-goal matrix for this experiment.[4] This leads to the conclusion that the rules defined in the matrix are based on the subjective perspective of the human expert. In turn the IR system that implements these rules makes use of that subjective perspective as well. For a real-world application of iRecognise it would thus be necessary to generalise the approach. One way of doing so would be the use of an artificial neural network to replace the hand-crafted rule set, as proposed above. A neural network trained with game session data from different players would function more effectively in an IR system used to recognise intentions of different players. Another approach might be to have a number of human experts to create the rules for the indicator-goal matrix.

A drawback of our current approach is that the human player can only select one single intention during the experiment at a time. This approach is insufficient for a real-world application, because, for example in RISK, a player might have multiple intentions in mind at any given point. The iRecognise system can handle this situation in theory, as it simply calculates the probability for each intention, without selecting one single intention as the "winner". In further research it would be beneficial for the experiments to allow for the selection of multiple intentions per turn within the experimentation framework.

[4]Note that player 1 is the author of this thesis.

During our experiments it became clear that playing several iterations of RISK games is a time-consuming task for human players. For the experiments presented above the number of sample games has therefore been limited, as well as the number of players playing these sample games. An improvement in this methodology would be to create an online version of the RISK experimentation framework. If that were to be achieved, the link to that online version could be sent to various people and thereby gather more experimentation data.

6.5 Model Evaluation

In this section we apply the model of user intentions proposed in chapter 3 of this thesis to the RISK game used in this case study.

At each game turn iRecognise receives a list of ground literals, which together form a world state w. The list of valid ground literals in RISK is defined in section 5.5.1 above. For each world state iRecognise applies the function f^v in order to transform the ground literals into a set of real-numerical indicator values, V_w, as defined in equation (3.1). For the RISK game in our case study, the list of all valid indicator values to which this function can map is given in section 5.5.3.

The rules of the RISK game define a list of possible actions that can be used by the players to alter the current world state (like *attack* or *trade-cards*, as defined in the description of the RISK game, above). Each such action has one or more effects on the world state. In our model of user intentions these effects of a given action are calculated by the function f^e, as shown in equation (3.2). One such effect is defined as shown in equation (3.3). In our RISK case study, for example, *trade-cards* decreases the number of cards in the player's hand, while increasing the number of reinforcement units for that player.

The collective effect of all actions in a player's plan is again calculated via the function f^e, as shown in equation (3.4). In RISK one plan could be to first trade in a pair of cards, then to stack all new reinforcements armies on to one single territory, in order to attack the enemy player in a neighbouring territory. The formal definition of such a plan is given in equation (3.5). Note that currently in iRecognise player plans are not taken into account.

Each action in RISK, either being part of a plan or just considered in isolation, has at least one effect on a given world state. This effect is measured by comparing the current world state at time step t with the world state in the last time step $t-1$, as defined in equation (3.6). In the previous example the number of player cards in the world state at time step t after performing the *trade-cards* action decreases by 3 in comparison to the world step at time step $t-1$.

Finally, we treat intentions as goals, as outlined in the description of the indicator-goal matrix above. In the proposed model of user intentions, such a goal is a world state within the game environment desired by a player. To change the current world state to that desired goal world state the player performs one or more actions, i.e. a plan, as defined in equation (3.7). In the example of our RISK case study the desired goal state may be to increase the number of armies, i.e. to gain additional reinforcement units. The desired goal world state is thus the current world state modified by the increased number of armies. To achieve that goal the player can, for example, perform the action *trade-cards*.

Chapter 7

Conclusion and Future Work

The work presented in this thesis focuses on two main areas: the development of a model of user intentions in games, and the recognition of player intentions in games. For the latter we have evaluated existing approaches and presented our own novel IR system design.

We started with our work on the model of user intentions after discussions on various aspects of games and interactive environments in general, and how users interact with them. In these discussions we realised that it is not obvious what those different words mean to different people: actions, plans, goals, strategies, intentions, etc. We thus felt the need for a common dictionary for these words and their definitions. This dictionary eventually evolved into the model as presented in chapter 3. We hope that the proposed model also helps other researchers in their work with plan and intention recognition in games, as well as (modelling) player behaviour in general.

However, the model we have developed for our work not only provides a set of common definitions. It also made us consider how all these concepts relate to each other. These relations are one of the key aspects of the proposed model, as they are fundamental to the second main part of the work presented in this thesis: intention recognition in games.

Our work on intention recognition is based on different existing ideas in that area. For our research we have evaluated various concepts of both plan and intention recognition. Some are brilliant approaches, however, none of them is a) primarily targeting gaming environments, and b) devoid of some shortcomings. Considering these different concepts, we asked ourselves what parts of them we could alter to deal with the various issues arising. After initial experimentations we eventually agreed to focus on a novel approach to intention recognition: to solely consider changes in the game world state over time.

On one hand this approach allows us to simplify the overall design of our IR system, as we can ignore actions performed by players altogether and omit the use of any plan library. On the other hand we can focus on what is the reason for performing these actions in the first place: to alter the current world state. And in our opinion this is exactly what the function of an intention recognition system is.

The results of the experiments shown in chapter 6 of this thesis prove that our proposed approach to IR works. There is however room for improvements. If we consider only exact matches in our

prototype system iRecognise, then the system performance is not acceptable. We consider this to be unsurprising, as we know from our personal experience that it is impossible to read the opponent's mind. It is clearly not possible, even for experienced players, to play a game against a human opponent and always know what intention, goal, or strategy the opponent has in mind all the time. Even for experienced human players it is more likely to guess that the opponent's intention is one out of two or three potential intentions. Thus, if we broaden the definition of "successful recognition" to the top three most probable intentions, our prototype system works quite well. In approximately three quarters of all cases the system correctly recognises the player intention. We believe that for a real-world application of our approach improving the rules in the indicator-goal matrix would improve on that performance even further.

In further future research we can recommend several ways to improve on our proposed approach to intention recognition in games. Foremost, we consider the manual definition of rules in the indicator-goal matrix the main issue in the current design. That manual approach is not only time-consuming for human experts, it is also error prone and would take a considerable amount of time to optimise. Moreover, as our experiments have shown, that approach means that the IR system works better for the human expert who has defined these rules than for other players. To address this issue in future research we would propose that the use of artificial neural networks is implemented, to replace these rules, as discussed in section 4.2.2. The concept is to let the network learn the mapping from indicator values as system input to player intentions as system output. This approach would allow us to use sample game data from different players, in order to improve on the generalisation ability of the IR system performance.

Other improvements to the current approach as presented in this thesis (as well as interesting fields of successive research) are:

- The use of more and different indicator values,
- the application of the prototype system to other games than RISK, and
- an online experimentation platform to gather more sample data from more players.

The approach proposed in this thesis might also be extended to recognise not only intentions, but plans (i.e. the next actions) as well. We also want to continue research on the automatic identification and selection of valid and suitable indicator values for a given game, as discussed in section 4.1. Finally we would like to apply our IR approach not only to games, but to other areas of interactive computation as well.

The general outcome of our research on intention recognition is that our initial concept of omitting player actions is proven to be a suitable approach. As outlined in the introduction of this thesis the use of plan and intention recognition systems can greatly improve the overall quality of games in general. We believe that our research as presented in this thesis can eventually become an important improvement for professional industry games.

Bibliography

Aha, David W., Molineaux, Matthew, and Ponsen, Marc. *Learning to Win: Case-Based Plan Selection in a Real-Time Strategy Game*, pages 5–20. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005.

Albrecht, David W, Zukerman, Ingrid, and Nicholson, An E. Bayesian models for keyhole plan recognition in an adventure game. *User modeling and user-adapted interaction*, 8(1-2):5–47, 1998.

Anscombe, G.E.M. *Intention*. Basil Blackwell, Oxford, 1957.

Barakova, Emilia I. and Lourens, Tino. Expressing and interpreting emotional movements in social games with robots. *Personal and Ubiquitous Computing*, 14(5):457–467, 2010.

Barès, Michael, namero, Dolores Ca Delannoy, Jean-Francois, and Kodratoff, Yves. XPlans: Case-based reasoning for plan recognition. *Applied Artificial Intelligence*, 4(8):617–643, 1994.

Bergmann, Ralph, Kolodner, Janet, and Plaza, Enric. Representation in case-based reasoning. *The Knowledge Engineering Review*, 20:209–213, 9 2005.

Bevilacqua, V., Mastronardi, G., Menolascina, F., Pannarale, P., and Pedone, A. A novel multi-objective genetic algorithm approach to artificial neural network topology optimisation: The breast cancer classification problem. In *Neural Networks, 2006. IJCNN '06. International Joint Conference on*, pages 1958–1965, 2006.

Blatt, Sharon. RISKy Business: An In-Depth Look at the Game RISK. *Undergraduate Math Journal*, 3(2), 2002.

Bratman, M. *Intention, plans, and practical reason*. Harvard University Press, 1987.

Buro, Michael. From simple features to sophisticated evaluation functions. In *Proceedings of the First International Conference on Computers and Games*, pages 126–145. Springer-Verlag, 1999.

Campbell, Murray, Hoane, A.Joseph, and hsiung Hsu, Feng. Deep blue. *Artificial Intelligence*, 134 (1):57–83, 2002.

Carberry, Sandra. Techniques for plan recognition. *User Modeling and User-Adapted Interaction*, 11(1-2):31–48, Mar. 2001.

Cardona, A. B., Togelius, J., and Nelson, M. J. Competitive coevolution in ms. pac-man. In *2013 IEEE Congress on Evolutionary Computation*, pages 1403–1410, June 2013.

Charniak, Eugene and Goldman, Robert P. A bayesian model of plan recognition. *Artificial Intelligence*, 64(1):53–79, Nov. 1993.

Chen, Z. *Computational Intelligence for Decision Support*. CRC Press, Boca Raton, FL, USA, 1999.

Cohen, Philip R. and Levesque, Hector J. Intention is choice with commitment. *Artificial Intelligence*, 42(2-3):213–261, Mar. 1990. ISSN 0004-3702.

Copeland, B.J. *The Essential Turing*. Clarendon Press, 2004. ISBN 9780191606861.

David-Tabibi, Omid, Koppel, Moshe, and Netanyahu, NathanS. Expert-driven genetic algorithms for simulating evaluation functions. *Genetic Programming and Evolvable Machines*, 12(1):5–22, 2011.

Driver, Julia. Gertrude Elizabeth Margaret Anscombe. `http://plato.stanford.edu/entries/anscombe/`, 2011. Online; Accessed: 2014-08-03.

Edelkamp, Stefan and Kissmann, Peter. Symbolic classification of general two-player games. In Dengel, AndreasR., Berns, Karsten, Breuel, ThomasM., Bomarius, Frank, and Roth-Berghofer, ThomasR., editors, *KI 2008: Advances in Artificial Intelligence*, volume 5243 of *Lecture Notes in Computer Science*, pages 185–192. Springer Berlin Heidelberg, 2008.

Egenfeldt-Nielsen, S., Smith, J.H., and Tosca, S.P. *Understanding Video Games: The Essential Introduction*. Routledge, 2016. ISBN 9781317533139.

Erol, Kutluhan, Nau, Dana S., and Subrahmanian, V. S. When is planning decidable? In *Proceedings of the First International Conference on Artificial Intelligence Planning Systems*, pages 222–227, San Francisco, CA, USA, 1992. Morgan Kaufmann Publishers Inc.

Fagan, Michael and Cunningham, Pádraig. *Case-Based Plan Recognition in Computer Games*, pages 161–170. Springer Berlin Heidelberg, Berlin, Heidelberg, 2003.

Feller, W. *An introduction to probability theory and its applications*. Number v. 1 in Wiley mathematical statistics series. Wiley, 1950.

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

Ghory, Imran. Reinforcement learning in board games. Technical report, Department of Computer Science, University of Bristol, 05 2004.

Gibney, Elizabeth. Google AI algorithm masters ancient game of go. *Nature*, 529(7587):445–446, jan 2016.

Guyon, Isabelle and Elisseeff, André. An introduction to variable and feature selection. *The Journal of Machine Learning Research*, 3:1157–1182, Mar. 2003. ISSN 1532-4435.

Hoang, Hai, Lee-Urban, Stephen, and Munoz-Avila, Hector. Hierarchical plan representations for encoding strategic game ai. In *Proc. Artificial Intelligence and Interactive Digital Entertainment Conference*. AAAI Press, 2005.

Huang, J., Huo, W., Xu, W., Mohammed, S., and Amirat, Y. Control of upper-limb power-assist exoskeleton using a human-robot interface based on motion intention recognition. *IEEE Transactions on Automation Science and Engineering*, 12(4):1257–1270, Oct 2015.

Humphrey, Nicholas. This chimp will kick your ass at memory games - but how the hell does he do it? *Trends in Cognitive Sciences*, 16(7):353–355, 2012.

Johnson, Peter, Hilary, Peter Johnson, and Shouls, Alan. Task-related knowledge structures: Analysis, modelling and application. In *People and Computers IV*, pages 35–62. University Press, 1988.

Kautz, Henry A. Reasoning about plans. chapter A Formal Theory of Plan Recognition and Its Implementation, pages 69–124. Morgan Kaufmann Publishers Inc., 1991.

Kerkez, Boris and Cox, Michael T. Incremental case-based plan recognition with local predictions. pages 413–463, 2003.

Khan, Jibram A.Z. Plan recognition in risk. April 2013.

Kiefer, P. *Mobile Intention Recognition*. Springer Science+Business Media, LLC, 2011.

Lee, Kyungmin, Chu, David, Cuervo, Eduardo, Kopf, Johannes, Grizan, Sergey, Wolman, Alec, and Flinn, Jason. Outatime: Using speculation to enable low-latency continuous interaction for cloud gaming. Technical Report MSR-TR-2014-115, August 2014.

Levy, D.N.L., Shannon, C.E., and Turing, A.M. *Computer Chess Compendium*. Ishi Press, 2009. ISBN 9784871878043.

Luger, George F. *Artificial Intelligence: Structures and Strategies for Complex Problem Solving*. Addison-Wesley, sixth edition, 2009.

Mazzucato, Mariana. *Strategy for Business*. SAGE Publications Ltd, 2002.

Meneguzzi, Felipe, Zorzo, Avelino Francisco, da Costa Móra, Michael, and Luck, Michael. Incorporating planning into BDI agents. *Scalable Computing: Practice and Experience*, 8:15–28, 2007.

Millington, I. and Funge, J. *Artificial Intelligence for Games*. CRC Press, 2009. ISBN 9780080885032.

Miyake, Youichiro. *Current Status of Applying Artificial Intelligence in Digital Games*, pages 1–32. Springer Singapore, Singapore, 2015.

Palazzo, L., Dolcini, G., Claudi, A., Biancucci, G., Sernani, P., Ippoliti, L., Salladini, L., and Dragoni, A. F. Spyke3d: A new computer games oriented bdi agent framework. In *Computer Games: AI, Animation, Mobile, Interactive Multimedia, Educational Serious Games (CGAMES), 2013 18th International Conference on*, pages 49–53, July 2013.

Russell, Stuart Jonathan and Norvig, Peter. *Artificial intelligence: a modern approach*. Prentice Hall, 3rd edition, 2009.

Sadri, Fariba. Intention recognition with event calculus graphs. In *Proceedings of the 2010 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology - Volume 03*, pages 386–391, Washington, DC, USA, 2010. IEEE Computer Society.

Sadri, Fariba. Logic-based approaches to intention recognition. In Chong, N.Y. and Mastrogiovanni, F., editors, *Handbook of Research on Ambient Intelligence and Smart Environments: Trends and Perspective*, IGI Global Research Collection. Information Science Reference, 2011.

Schmidt, C. F., Sridharan, N. S., and Goodson, J. L. The plan recognition problem: An intersection of psychology and artificial intelligence. *Artificial Intelligence*, 11(1-2):45–83, Aug. 1978.

Shannon, Claude E. Programming a computer for playing chess. *Philosophical Magazine*, 41: 256–275, 1950.

Spyrou, T. and Darzentas, J. Intention modelling: Approximating computer user intentions for detection and prediction of intrusions. In *Information System Security*, pages 319–335. Chapman & Hall, 1996.

Straffin, Philip D. *Game Theory and Strategy (New Mathematical Library, No. 36)*. The Mathematical Association of America, 1993.

Tan, H. and Kawamura, K. Generation of acceptable actions using imitation learning, intention recognition, and cognitive control. In *Robot and Human Interactive Communication (RO-MAN), 2015 24th IEEE International Symposium on*, pages 389–393, Aug 2015.

Togelius, Julian, Champandard, Alex J, Lanzi, Pier Luca, Mateas, Michael, Paiva, Ana, Preuss, Mike, and Stanley, Kenneth O. Procedural content generation: Goals, challenges and actionable steps. *Dagstuhl Follow-Ups*, 6, 2013.

Vaccaro, James and Guest, Clark. Evolutionary bayesian network dynamic planner for game risk. In *Applications of Evolutionary Computing*, volume 3005 of *Lecture Notes in Computer Science*, pages 549–560. Springer Berlin Heidelberg, 2004.

Vaccaro, James and Guest, Clark. Learning multiple search, utility, and goal parameters for the game risk. In *IEEE Congress on Evolutionary Computation*, pages 1208–1215, 2006.

Von Neumann, J. and Morgenstern, O. *Theory of games and economic behavior*. Princeton University Press, 1953.

Wolf, Michael. *An Intelligent Artificial Player for the Game of Risk*. PhD thesis, TU Darmstadt, Knowledge Engineering Group, Mar. 2005. Unpublished Doctoral Dissertation.

Xu, W., Huang, J., and Yan, Q. Multi-sensor based human motion intention recognition algorithm for walking-aid robot. In *2015 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 2041–2046, Dec 2015.

Xu, Yanxiang, Luo, Tiejian, Zhu, Tingshao, and He, Haibo. The principles of intention computing. In *Web Society, 2009. SWS '09. 1st IEEE Symposium on*, pages 242–249, Aug 2009.

Yannakakis, G. N. and Togelius, J. A panorama of artificial and computational intelligence in games. *IEEE Transactions on Computational Intelligence and AI in Games*, 7(4):317–335, Dec 2015.

Yin, Ming-Hao, Ou, Hua-Jie, Gu, Wen-Xiang, Lu, Yin-Hua, and Liu, Ri-Xian. Fast probabilistic plan recognition without plan library. In *Machine Learning and Cybernetics, 2004. Proceedings of 2004 International Conference on*, volume 1, pages 157–162, Aug 2004.

Yue, Shi-guang, Zha, Ya-bing, Yin, Quan-jun, and Qin, Long. Multi-agent intention recognition using logical hidden semi-markov models. In *Simulation and Modeling Methodologies, Technologies and Applications (SIMULTECH), 2014 International Conference on*, pages 701–708, Aug 2014.

Appendices

Appendix A: List of Abbreviations

AI — Artificial Intelligence

API — Application Programming Interface

BDI — Belief Desire Intention

CBR — Case-Based Reasoning

GUI — Graphical User Interface

HCI — Human-Computer-Interaction

HMM — Hidden Markov Models

IR — Intention Recognition

JSON — JavaScript Object Notation

MO — Multiple Objectives

MUD — Multi User Dungeon

NPC — Non-Player Character

PC — Player Character

PR — Plan Recognition

SSC — State-Space Complexity

TA — Task Analysis

TCP — Transmission Control Protocol

TKS — Task Knowledge Structures

WSI — World State Indicators

WSO — World State Objects

XML — Extensible Markup Language

Appendix B: iRecognise System Database Tables

Column	Key	Type	Description
id	X	INTEGER	Primary Key, Unique ID
game		INTEGER	Reference to the Game ID (1 = RISK)
user		INTEGER	Reference to the user/player ID
start		TEXT	Date and Time Stamp of session start
end		TEXT	Date and Time Stamp of session end
ws_count		INTEGER	Number of world state iterations (turns)
comment		TEXT	Optional comment set to this session

Table 7.1 – iRecgonise Table "sessions": Stores meta-information to single game sessions.

Column	Key	Type	Description
id	X	INTEGER	Primary Key, Unique ID
session		INTEGER	Foreign key to a session ID
time_step		INTEGER	World State counter within one session
unix_time		INTEGER	UNIX time value of a world state
intention_probs		TEXT	Contains all intention probability values
ai_intention		TEXT	Which Intention the AI bot had at that time

Table 7.2 – iRecgonise Table "world_states": Stores information to a single word state.

Column	Key	Type	Description
ws_id		INTEGER	Refers to the ID of a World State
user_modified		BOOLEAN	Whether or not this object is user-modified
name		TEXT	Name of this World State object

Table 7.3 – iRecgonise Table "ws_objects": Stores meta-data to single world states.

Column	Key	Type	Description
ws_id		INTEGER	Refers to the ID of a World State
literal_name		TEXT	The name of the World State ground literal
term_name		TEXT	The name of the ground literal's term
value		TEXT	The value of the this term

Table 7.4 – iRecgonise Table "ws_terms": Stores the full data of a world state object.