

Abstract

BAIKADI, ALOK. Discovery-based Goal Recognition in Interactive Narrative Environments. (Under the direction of James Lester.)

Recent years have seen a growing interest in automated goal recognition. Goal recognition is the problem of recognizing goals by observing the actions they perform. Models of goal recognition can be used to support students in an intelligent tutoring system, reduce communication in a problem-solving system or a dialogue system, and dynamically adapt a game to encourage players strengths and interests, or exploit their weaknesses in an adversarial system.

The work presented in this dissertation expands on previous work on goal recognition by introducing a representation of user progress. *Discovery events* are events in the environment that convey key information to the user. We utilize discovery events to solve the goal recognition problem for an important class of goal recognition problems, namely, recognizing the goals of users interacting with narrative-centered learning environments. These environments contextualize the learning materials within an interactive narrative. In these environments, progress towards the goal is strongly influenced by the pacing and structure of the encompassing narrative, so by representing the discovery events within the narrative, the resulting goal recognition models are able to more fully capture the context in which actions are performed.

Our solution to the goal recognition problem uses a machine learning framework based on Markov Logic Networks (MLNs). MLNs are a statistical relational learning framework that combines probabilistic inference with first-order logical reasoning. The resulting models are trained and evaluated on two distinct narrative-centered learning environments gathered from middle school students across North Carolina. Empirical

evaluations suggest that goal recognition systems that represent discovery events models are more accurate, converge more often, and converge earlier in observation sequences than baseline models.

Discovery-based Goal Recognition in Interactive Narrative
Environments

by
Alok Baikadi

A dissertation submitted to the Graduate Faculty of
North Carolina State University
in partial fulfillment of the
requirements for the Degree of
Doctor of Philosophy

Computer Science

Raleigh, North Carolina

2014

APPROVED BY:

R. Michael Young

David Roberts

Marc Russo

James Lester
Chair of Advisory Committee

Biography

Alok Baikadi was born in San Antonio, Texas on February 3rd, 1985. He grew up in Clarks Summit, Pennsylvania, where he attended the local elementary, middle, and high schools. He was first introduced to computer science through the Boy Scouts of America, where he earned a computer science merit badge. He remained active in Boy Scouts all through high school, and received the highest award: the Eagle Scout rank. His other primary activity in high school was involvement in the music program. In addition to many school ensembles, he participated in a community orchestra, and several PMEA competitions at the district and regional level.

He graduated from Abington Heights High School in 2003, and attended the University of Illinois at Urbana-Champaign. He graduated from the University of Illinois in 2007, receiving a Bachelors of Science degree in Computer Science, as well as two minors in Mathematics and Music Performance. During his time in Illinois, he joined the local Association for Computing Machinery chapter, where he was the chairman of the SIGMusic group for two years. In addition, he performed research under the direction of Dr. Eyal Amir on representations of narrative in description logics, the preliminary steps towards his continued interest in computational models of narrative.

He began his Ph.D career at North Carolina State University in 2007. During the course of his Ph.D studies, he has continued to study narrative and creativity, both as a research interest and through activities in conjunction with National Novel Writing Month. He looks forward to continuing these interests as he moves on to the next chapter of his life.

Acknowledgements

There have been many people who've helped me get to where I am today. First of all, I would like to thank my advisor, James Lester, who has supported, encouraged, and inspired me throughout my graduate career. His advice and guidance always helped me find my way, even when I thought I was lost. I would also like to thank the members of my committee, Michael Young, Dave Roberts and Marc Russo for their thoughtful feedback and advice throughout the process.

I have also enjoyed many years of collaboration with my colleagues in the IntelliMedia group and the Digital Games Research Center. Jonathan Rowe and Bradford Mott have offered guidance and collaboration on many projects, and helped inspire the direction for my work. Eunyoung Ha and Jonathan Rowe wrote the feature extraction scripts, as well as the baseline model which enabled me to conduct the work reported in this dissertation. Julius Goth and Marc Russo collaborated on my early forays into narrative research, and provided many helpful conversations along the way. Rogelio Cardona-Rivera collaborated on a computational model of narrative that paved the way for the representation used in this work. Kristy Boyer offered invaluable advice on graduate school and on how to approach research. Christopher Mitchell, Joseph Grafsgaard, and Brent Harrison read earlier drafts of this document and provided helpful suggestions to improve the quality. I would also like to thank Joseph Wiggins, Jennifer Sabourin, Andy Smith, Sam Leeman-Munk, Wookhee Min, Kirby Culbertson, Robert Taylor, Seung Lee, Veronica Catete and Justis Peters for helpful discussions and the occasional much-needed distraction.

I would not have gotten to where I am today without the love and support of my friends and family. I would particularly like to thank Sujin Headrick, for showing me that any writers block can be defeated, Christina Sauper for inspiration, advice, and

feedback on the work and its directions, Renee Demers, Johanne Christensen and Prairie Rose Goodwin for thoughtful discussions, emotional support, and reminding me to leave campus. I would also like to thank Jennifer Robinson, Sarra Bittman, and Erica Reeder for helping me hone my writing and develop an appreciation for the craft.

Last but not least, I would like to thank my parents, Madhava and Leela, and my siblings Megha and Ashwin for their love and support, and for putting up with many long, trying hours of work and stress throughout my life. I would never have made it here without you.

This research was supported by the National Science Foundation under Grant DRL-0822200. Any opinions, findings, and conclusions expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation. Additional support was provided by the Bill and Melinda Gates Foundation, the William and Flora Hewlett Foundation, EDUCAUSE, and the Social Sciences and Humanities Research Council of Canada.

Table of Contents

List of Tables	viii
List of Figures	x
Chapter 1 Introduction	1
1.1 Thesis Statement and Hypotheses	4
1.2 Contributions	5
1.3 Organization	6
Chapter 2 Related Work	7
2.1 Plan, Goal, Activity and Intent Recognition	8
2.2 Narrative	17
Chapter 3 MLN-based Goal Recognition in Interactive Narrative	22
3.1 Task Definition	22
3.2 Approach	24
3.3 Applications	26
3.4 Markov Logic Networks	27
3.4.1 Inference in MLNs	28
3.4.2 Learning Markov Logic Networks	29
3.4.3 Tools for Markov Logic Networks	30
3.5 Goal Recognition	31
3.6 Summary	35
Chapter 4 Discovery Events	37
4.1 Narrative in Goal Recognition	38
4.2 Defining Discovery Events	40
4.3 Example Story	43
4.4 Summary	44
Chapter 5 Corpora	45
5.1 CRYSTAL ISLAND: OUTBREAK	46
5.1.1 Story & Gameplay	47
5.1.2 Corpus Description	48
5.1.3 Goal Recognition in CRYSTAL ISLAND: OUTBREAK	49
5.2 CRYSTAL ISLAND: UNCHARTED DISCOVERY	56
5.2.1 Story & Gameplay	56
5.2.2 Corpus Collection	58

5.2.3	Goal Recognition in CRYSTAL ISLAND: UNCHARTED DISCOVERY	59
5.3	Summary	65
Chapter 6	Evaluation	66
6.1	Experimental Design	67
6.2	Hypotheses	72
6.2.1	Hypothesis 1: Accuracy	72
6.2.2	Hypothesis 2: Convergence	74
6.2.3	Hypothesis 3: Transference	75
6.3	Summary	77
Chapter 7	Discussion	80
7.1	Ablated Formulae	81
7.2	Representations of Narrative State	83
7.3	Narrative Schema	86
7.4	Limitations	88
Chapter 8	Conclusions and Future Work	90
8.1	Hypotheses Revisited	91
8.2	Summary	93
8.3	Future Work	94
8.4	Concluding Remarks	97
Bibliography		98
Appendices		109
Appendix A	Model Files for CRYSTAL ISLAND: OUTBREAK	110
A.1	Data Definitions	110
A.2	State of the Art Model	111
A.3	Discovery Events Model	115
Appendix B	Model Files for CRYSTAL ISLAND: UNCHARTED DISCOVERY	119
B.1	Data Definitions	119
B.2	State of the Art Model	120
B.3	Discovery Events Model	124
Appendix C	Extra Model Files	130
C.1	Ablated Experiment	130
C.1.1	Ablated Only	130
C.1.2	Baseline with Discovery Events	132
C.2	Representation Experiment	138
C.2.1	CRYSTAL ISLAND: UNCHARTED DISCOVERY with Milestone States	138

C.2.2	CRYSTAL ISLAND: UNCHARTED DISCOVERY with Milestone States and Discovery Events	142
C.3	Schema Experiment	149
C.3.1	Data Definitions	149
C.3.2	Schema Model	150

List of Tables

Table 3.1	List of predicates for goal recognition	33
Table 5.1	Goals in CRYSTAL ISLAND: OUTBREAK, with relative frequency . . .	50
Table 5.2	Actions within Crystal Island, with the frequency of occurrence . . .	53
Table 5.3	Goal Statistics for CRYSTAL ISLAND: UNCHARTED DISCOVERY . . .	60
Table 5.4	Actions with Relative Frequencies in CRYSTAL ISLAND: UNCHARTED DISCOVERY	62
Table 6.1	Average F_1 scores for each model on the CRYSTAL ISLAND: OUT- BREAK dataset. The p-value for the Tukey test is shown for all com- parisons.	73
Table 6.2	Average convergence rate for each model on the CRYSTAL ISLAND: OUTBREAK dataset. The p-value for the Tukey test is shown for all comparisons.	75
Table 6.3	Average convergence point for each model on the CRYSTAL ISLAND: OUTBREAK dataset. The p-value for the Tukey test is shown for all comparisons.	75
Table 6.4	Average F_1 scores for each model on the CRYSTAL ISLAND: UN- CHARTED DISCOVERY dataset. The p-value for the Tukey test is shown for all comparisons.	77
Table 6.5	Average convergence rate for each model on the CRYSTAL ISLAND: UNCHARTED DISCOVERY dataset. The p-value for the Tukey test is shown for all comparisons.	77
Table 6.6	Average convergence point for each model on the CRYSTAL ISLAND: UNCHARTED DISCOVERY dataset. The p-value for the Tukey test is shown for all comparisons.	78
Table 7.1	Average F1 scores for each model on the CRYSTAL ISLAND: OUT- BREAK dataset. The p-value for the Tukey test is shown for all com- parisons.	82
Table 7.2	Average convergence rate scores for each model on the CRYSTAL IS- LAND: OUTBREAK dataset. The p-value for the Tukey test is shown for all comparisons.	82
Table 7.3	Average convergence point scores for each model on the CRYSTAL IS- LAND: OUTBREAK dataset. The p-value for the Tukey test is shown for all comparisons.	83

Table 7.4	Results of the representation experiment on the CRYSTAL ISLAND: OUTBREAK dataset.	85
Table 7.5	Results of the representation experiment on the CRYSTAL ISLAND: UNCHARTED DISCOVERY dataset.	85
Table 7.6	Results of the schema experiment on the CRYSTAL ISLAND: OUTBREAK dataset.	87

List of Figures

Figure 3.1	Single predicate formulae in MLN.	34
Figure 3.2	Multiple predicate formulae in MLN.	35
Figure 3.3	Milestone predicate formulae in MLN.	35
Figure 6.1	Graphical model for the state-of-the-art baseline	68
Figure 6.2	Baseline formulae for MLNs	69
Figure 6.3	Graphical model for the CRYSTAL ISLAND: OUTBREAK discovery events model	70
Figure 6.4	Graphical model for the CRYSTAL ISLAND: UNCHARTED DISCOVERY discovery events model	71
Figure 6.5	Discovery event formulae for CRYSTAL ISLAND: OUTBREAK	71
Figure 6.6	Discovery event formulae for CRYSTAL ISLAND: UNCHARTED DISCOVERY	72
Figure 7.1	Milestone Formulae for the Narrative State in CRYSTAL ISLAND: UNCHARTED DISCOVERY	84
Figure 7.2	Schema formulae for MLNs	86

Chapter 1

Introduction

The past several decades have seen a sharp change in human-computer interaction. Computers have gone from special purpose tools only used by a select few, highly trained individuals, to pervasively used systems designed to solve a broad range of problems. Applications have been developed over the years that range from highly specialized software to solve a single problem, to large scale, exploratory systems designed to allow a user full access to the creative space. In tandem with these innovations, techniques have been developed to support users' interactions. One of the key challenges in supporting a user in these problem solving spaces is recognizing the user's intent. *Goal recognition*, also known as *intention recognition*, is a restricted form of a classical artificial intelligence problem, *plan recognition*. Goal recognition allows the system to infer the goals, or intentions, of a user given an observation of the actions the user is performing.

One type of system that benefits from goal recognition techniques is the structured problem-solving environment. These environments allow the users to explore the problem-solving space, while still providing tools and feedback directed towards achieving a specific task. Examples include intelligent tutoring systems (Gal et al., 2012),

which guide the users towards solving a series of problems to learn some concepts and skills; dialogue systems (Blaylock and Allen, 2006), which provide natural communication paradigms for achieving various goals; and interactive narratives (Gold, 2010; Kabanza et al., 2010; Thue and Bulitko, 2012), which seek to balance player agency with authored story constructs.

Once the goals are recognized by the system, they could be used in several ways. In an intelligent tutoring system, goal recognition could be used to support a struggling student or to provide assessment to the instructor (Gal et al., 2012). In a problem-solving scenario or dialogue system, plan recognition could be used to reduce communication between the user and the system (Lesh et al., 1999). In a game environment, it could be used to dynamically adapt the game in response to the player’s strengths and interests (Thue and Bulitko, 2012) or to model more challenging opponents in a competitive environment (Kabanza et al., 2010).

One component of the goal recognition task is the ability to reason about the context in which the actions are occurring. Several approaches have modeled the context in terms of what changes the actions are making to the environment (Pynadath and Wellman, 2000; Geib, 2009; Geib and Goldman, 2009; Gold, 2010). However, relatively few systems have considered how the user experiences these events. This work addresses this limitation by creating a model of *discovery events*. Discovery events are actions taken within the interactive system that provide the user with key information towards solving the eventual task. If the system is designed to train a mechanic to repair a car, discovery events may be diagnostic actions that help the users determine what needs to be replaced. On the other hand, if the system is an interactive narrative, discovery events may instead be key story points that reveal crucial information.

For this work, we consider a specific type of structured problem solving environment,

called narrative-centered learning environments. These environments combine features of both intelligent tutoring systems and interactive narratives to provide an engaging learning experience. Narrative-centered learning environments have been shown to increase learning, motivation, and presence within the learning environment. (Ketelhut et al., 2010; Hickey et al., 2009; Johnson, 2010; Sabourin et al., 2011; Thomas and Young, 2010; Rowe et al., 2011)

Because narrative-centered learning environments embed the problem-solving task within the narrative of the game, discovery events are often delivered as narrative milestones. In this work, we investigate discovery events as a model of narrative structure. These narrative discovery events define progress towards the task as milestones in which the player discovers an aspect of the game world that is relevant to the solution. This may include crucial backstories for certain characters, interactions with the objects in the environment, or access to in-game resources that aid the player in determining their course of action.

Several narrative representations have evolved over the years to tackle different problems. Within the area of natural language processing (NLP), representations have been developed for narrative generation (Callaway and Lester, 2002; Lang, 1999; Montfort, 2006), narrative understanding (Chambers and Jurafsky, 2009; Bejan and Harabagiu, 2010), analogical reasoning (Elson, 2012a), summarization (Appling and Riedl, 2009; Mani, 2004; Lehnert, 1981) and analysis (Sack, 2012). Other modeling approaches have been developed to capture common sense (Liu and Singh, 2002), ontological (Tuffield et al., 2006; Peinado et al., 2004) and cognition (Passonneau et al., 2007).

Narrative representations have also been developed for interactive narratives. These representations are often viewed as either character-centric (Cavazza et al., 2002; Theune et al., 2003) or plot-centric (Riedl and Young, 2005). Some attempts (Riedl, 2004; Mateas

and Stern, 2002) have attempted to blend the two as well. Representations have included planning structures (Riedl and Young, 2006), scripts (Orkin et al., 2010; Li et al., 2012), ontology (Grasbon and Braun, 2001) or specialized data structures (Mateas and Stern, 2005; Barber and Kudenko, 2007).

1.1 Thesis Statement and Hypotheses

This dissertation investigates the thesis that

Narrative representation can be leveraged in a principled methodology in order to produce goal recognition systems which are more accurate and converge faster than state of the art models.

There are several desiderata for a goal recognition system. First, a model for goal recognition should be accurate. In the current framework for goal recognition, predictions for the users goal are made after every observation. Accuracy is measured by the proportion of observations that the model is able to accurately predict. Second, a model for goal recognition should eventually predict the correct goal. Especially in large exploratory domains, early actions may not be indicative of the user’s goal. However, as more information is gathered about the sequence of actions, the model should converge on the correct goal. Lastly, we are interested in how early the model is able to converge on the correct goal. Early prediction affords the system using the goal recognition model more options for reacting to the recognized goal.

In order to evaluate the effectiveness of the approach, we investigate three primary hypotheses. The phrase *Discovery Events Model* refers to a goal recognition model obtained through our framework.

Hypothesis 1 The discovery events model will be *more accurate* than baseline models.

Hypothesis 2 The discovery events model will both *converge more often* than baseline models and *converge earlier* than baseline models.

Hypothesis 3 The procedure for developing a discovery events model can be applied to another goal recognition domain with similar results.

1.2 Contributions

This dissertation reports on research that has made the following contributions to the fields of Plan, Activity and Intent Recognition, and Computational Models of Narrative:

- A proposal for a computational model of interactive narrative that captures progression within the story. It captures events that provide key insight into the problem-solving task, which can be leveraged to provide additional context for goal recognition (Baikadi and Cardona-Rivera, 2012; Baikadi et al., 2013).
- A methodology for leveraging the narrative structure of an interactive narrative experience for goal recognition. The process involves selecting key milestone events and representing them within the machine learning framework (Baikadi et al., 2012, 2013).
- An evaluation of a goal recognition model that implements the discovery events modeling procedure (Baikadi et al., 2013).
- An investigation of the discovery events modeling procedure on a second interactive narrative experience with less overarching narrative structure.

1.3 Organization

This dissertation is structured as follows. Chapter 2 presents a survey of prior work in both goal recognition and in narrative representation. Chapter 3 defines the problem of goal recognition and provides the ground work for the framework investigated in this research. Chapter 3 describes Markov Logic Networks (MLNs), the computational representation selected for this work. Chapter 3 further defines the representation of the goal recognition task within MLNs and presents the details of model construction. Chapter 4 discusses discovery events in the context of interactive narratives. Chapter 5 presents two narrative-centered learning environments used for this work. The first was used for the initial model development and evaluation, and the second was used for a validation study to ensure discovery events generalized to a new environment. Chapter 6 discusses the hypotheses designed to support the thesis statement and the experiments which were conducted. Chapter 7 discusses the limitations of the approach as well as several further experiments which were conducted to evaluate the relationship between the discovery events and the prior state of the art models. Chapter 8 summarizes the contributions of this work and discusses several possible direction for future research.

Chapter 2

Related Work

This dissertation is inspired from work in both Plan, Intent and Activity Recognition and Computational Models of Narrative. Section 2.1 discusses computational approaches to Plan, Intent, and Activity Recognition. Recent work has used rich probabilistic models to capture complex relationships between observations and goals across time. This section presents a survey of several approaches to both plan and goal recognition which use a variety of probabilistic and logical models.

Section 2.2 discusses current work in computational models of narrative. Many different narrative representations have been developed over the years, to solve problems ranging from summarization and understanding, to generation and adaptation in interactive environments. Some models seek to represent the events that occur in the world, using either logical plan-based representations, or linguistic representations based on semantic frames, while others instead form higher levels of abstraction, such as beats and plot units, which can be sequenced and elaborated upon to form a narrative. This section presents a survey of techniques from computational models of narrative, interactive narrative, and linguistics in order to capture complex narrative phenomenon.

2.1 Plan, Goal, Activity and Intent Recognition

Plan recognition, along with its sibling tasks of activity recognition and intent recognition, is a classical problem within artificial intelligence. Collectively called PAIR (Plan, Activity and Intent Recognition), these tasks enable intelligent agents and systems to reason about the behavior of other cooperative or antagonistic agents. A wide variety of techniques, ranging from symbolic abductive frameworks (Kautz and Allen, 1986; Ng and Mooney, 1992; Goldman et al., 1999; Inoue and Inui, 2011), to probabilistic methods (Charniak and Goldman, 1993; Pynadath and Wellman, 2000) and structured representations of behavior across time (Blaylock and Allen, 2006; Wu et al., 2007; Sadilek and Kautz, 2012) have been applied to the problem. It can also be applied to many tasks and applications, including dialogue (Carberry, 2001), pedagogy (Gal et al., 2012), intelligent environments (Wu et al., 2007), cybersecurity (Geib and Goldman, 2009) and story understanding (Charniak and Goldman, 1993). It has also been applied to several interactive experiences, including adventure games (Gold, 2010; Albrecht et al., 1997), real-time Strategy games (Kabanza et al., 2010), football (Lavers and Sukthankar, 2011), and capture the flag (Sadilek and Kautz, 2012).

Classical work in PAIR tasks focused on symbolic representations. One approach used often in early systems is abductive reasoning. Abduction attempts to infer an explanation for a given conclusion. Unlike deductive reasoning, abduction may infer incorrect facts. Action taxonomies (Kautz and Allen, 1986; Kautz, 1991) represents both specialization and decomposition of actions, and the plans are represented by the traversals of the hierarchies. By traversing the hierarchies, the top-level decomposition, which indicated the plan, could be inferred given ground observations. The ACCEL algorithm used a logical backwards chaining algorithm, similar to Prolog, to create a set of abductive

assumptions (Ng and Mooney, 1992). A model of narrative coherence was used to guide the abduction on a natural language story corpus.

More recently, Integer Linear Programming was used for weighted abduction (Inoue and Inui, 2011). Weighted abduction assigns a weight to each hypotheses clause and finds the set of clauses that minimize the weight. The hypotheses consist of logical facts about the environment. Each hypothesis clause is given an indicator variable and a variable to indicate if it should pay its cost. Constraints are added to indicate the relationships between the indicator and cost variables and to enforce logical reasoning through unification.

One of the challenges of plan recognition is the construction of a plan library. Cox and Kerkez (2006) used case-based reasoning to simultaneously learn a plan recognition model and a plan library given a partial plan library. Cases are constructed from abstract states, which count the number of predicate groundings in the state representation. Whenever it encountered a novel abstract state, the system would collect the nearest neighbors using a Euclidean distance and add that information to the plan library.

Logical abduction is not always well suited to reasoning about uncertainty. Probabilistic models can allow for extraneous observations, connect actions across time, and account for noise in the data. Goldman et al. (1999) developed a probabilistic model of plan execution that captured partial ordering amongst plan actions, and the chance that an action may be taken without a governing plan. The generative model could then be used for abduction, even in the presence of interleaved plans and negative evidence.

Probabilistic grammar approaches can instead be used to capture the relationships between plans. A probabilistic grammar adds weights to the production rules, and can select the most likely parse tree for a given sequence of tokens. For goal recognition, A Probabilistic State-dependent Grammar (Pynadath and Wellman, 2000) conditions the

production rules on a representation of belief state. The belief state captures the context of the previously observed actions in a compact manner by utilizing the independence assumptions inherent in the grammar expansion.

Alternatively Combinatorial Categorical Grammars (CCGs) can represent plans where not all actions are observed (Geib, 2009). A CCG represents groups of actions as categories, and captures the context in which actions can be applied, relative to other actions, as production rules. By carefully selecting the plan heads, or the leftmost category in a production rule, along causal structure a least commitment approach can provide efficiency gains.

Probabilistic graphical models provide compact representations of probability distributions by capturing independence information in the connections between random variables. They can range from simple naive Bayes classifiers to complicated network structures capturing temporal relationships, as well as complicated conditional dependencies. Charniak and Goldman (1993) presented a Bayesian belief network for plan recognition. The network was constructed from a plan knowledge base, which encoded plans as complex actions, which were composed of more specific actions. Nodes and edges were introduced from observations when an action was observed to fill a slot within a plan. This allowed the plan recognition module to update its beliefs based on new evidence.

The network can also be incrementally during training (Anh and Pereira, 2011). The Bayesian network had three layers, an action layer, an intention layer, and a causality layer. The net was grown when new observations were made, and combined using a Noisy-OR. Connections between intentions were learned to encode mutual exclusion. These edges allowed the system to provide predictions of multiple simultaneous goals.

The PHATT algorithm instead uses a Bayesian model of plan selection and execution to recognize plans (Geib and Goldman, 2009). Plans are represented as a plan tree

grammar, which uses a variation on context-free grammars to represent strict ordering between elements. The model of plan selection uses pending action sets to cover multiple possible goals. The execution model can be inverted via Bayesian reasoning to produce explanations given a observation sequence.

This was later extended (Kabanza et al., 2010) to include explicit modeling of information gathering actions and time and resource management in a real-time strategy game. Both strategy level goals, such as when to build up resources, and tactical goals, such as feints and preparing an ambush, modeled. Plans were parameterized to reduce the size of the plan library. In addition to the observed actions, an influence map which monitors how much control each opponent has over various locations on the game map, was used to strengthen the tactical goal recognition.

Markov Decision Processes (MDPs) are a common framework for modeling rational agents within dynamic environments. The MDP model learns a policy which can be queried in any state to select an action which maximizes the expected reward. The Abstract Hidden Markov Model (Bui et al., 2002) uses a hierarchy of abstract MDPs to select lower-level policies. Abstract policies capture local regions in the state space, modeling paths from a set of initial states to a set of destination states. The Abstract Hidden Markov Memory Model (Bui, 2003) extends the AHMM with a memory model. The memory model is an additional distribution parameter that affects plan selection, and allows the system to capture state to record which subplan was being executed. Both models can be represented as a Dynamic Bayesian Network, for fast inference.

Sometimes, it is impractical to build a plan library for a domain. Large, exploratory domains may have many approaches to a single goal, and many goals which the user may pursue. Goal recognition, or intent recognition, relaxes plan recognition by only considering the final goal. Instead of using Bayesian reasoning to model the plan library, a

Bayesian model can capture dependencies between observations and the goal. A dynamic Bayesian network is a compact representation of a Bayesian networks that allow for reasoning across time. The network describes a current situation and is repeated across multiple moments in time, with additional edges for the temporal dependencies.

Albrecht et al. (1997) use a network which connects actions and locations to a quest that the player is trying to achieve in an adventure game. The network is able to predict any of the next action, next quest, or next location that will be observed. Mott et al. (2006) used a similar model, in which actions and locations are augmented by a notion of a narrative state, which captures the episodic nature of the narrative. The dynamic Bayesian network was used to encode a factored n-gram model, where the current goal is conditionally dependent on a fixed number of prior observations and each class of observation is encoded in its own node.

Rather than require a pre-authored goal set, Bayesian inference can also be used to capture the relationships between intermediate and terminal goals (Pattison and Long, 2011). Both intermediate and terminal goal states are represented by conjunctions over facts about the world. Heuristics describe how much progress has been made towards any given goal, in order to constrain the search space. Intermediate goals are distinguished from terminal goals by observing their relationships to mutually exclusive sets of facts. If the hypothesized goal is the only one available given the mutual exclusion relationships, then it's a terminal goal.

A Hidden Markov Model (HMM) is a special case of dynamic Bayesian network where a single observation and a single hidden state are connected across time. Several variations on HMMs have been used to capture richer information about the goal recognition problem. A set of HMMs, one for each goal, can be used in a multi-user system (Sukthankar and Sycara, 2005). Actions and locations are taken over overlapping

time windows. Unlike Mott et al. (2006) or Albrecht et al. (1997), these locations are represented as the centroid of the team over the window.

Cascading HMMs (Blaylock and Allen, 2006) use multiple layers of HMMs with each hidden layer becoming the output of the layer above it, with each transition happening simultaneously across layers. Plans are represented as these lists of goal chains, which are then used to train the different levels of the HMMs. The goals are parameterized for compactness of representation, and are recognized through an application of Dempster-Shafer Theory to combine evidence.

Variable Order HMMs (Armentano and Amandi, 2009) allow some states to have access to a larger number of previous states. They are a compromise between fixed order Markov models, such as most HMMs, and full modeling of the action sequence. The latter approach is prone to data sparsity, as well as an increase in the size and complexity of the models. The order, or degree to which the current node is allowed to access prior data, is learned simultaneously with the probabilities by considering a Probabilistic Suffix Automaton. Each goal is represented by its own automaton in order to support multiple goals.

An Input-Output HMM can be used to model both the actions the users are taking, and the feedback they are receiving from the environment. The IO-HMM is trained as a generative model, with the input as the feedback from the system, and the output as the low-level actions. Each goal is represented by a separate hidden state. An additional game context node was added to aid in the efficiency of inference and capture some of the causal relationships between actions and observations (Gold, 2010).

Partially Observable Markov Decision Processes (POMDPs) are an extension of MDPs in which the POMDP model does not have complete knowledge of the environment. In goal recognition, this approach is able to represent an observer who does not have

complete knowledge of the user’s actions within the environment (Ramírez and Geffner, 2011). Both the sequence of actions taken, and the observations the user receives from the environment are assumed to be only partially complete. The POMDP is sampled to produce a value function, which can be used with Bayes rule to sample a distribution among goals given a potentially incomplete observation sequence can be calculated.

Conditional Random Fields (CRFs) have become increasingly popular in activity recognition. Activity recognition is a similar task to goal recognition in that it attempts to predict an overarching activity that governs a series of observations. CRFs are an undirected graphical model which condition every node globally on the observations. Rather than modeling a joint distribution over observations and hidden states, a CRF models only the conditional distribution of the hidden states, given a sequence of observations.

A Linear-Chain CRF is the variation of a CRF that is conditioned on a linear sequence of observations. In a smart-home environment, they can be used to model the activities that the habitants perform (Nazerfard et al., 2010). They model the relationships between sensor data and previous activities in order to predict the current activity. Linear-Chain CRFs are able to outperform HMMs when there is significant interdependence between observed features.

Factorial CRFs (Wu et al., 2007) explicitly unroll the temporal dependencies in order to detect multiple simultaneous activities. Factorial CRFs extend Linear-Chain CRFs by modeling a structure of distributed states. This structure allows the model to avoid modeling each possible set of co-temporal activities, and produce a more compact model. Temporal dependencies between activities are modeled by explicitly unrolling the model across time.

Skip-Chain CRFs instead extend Linear-Chain CRFs by allowing for connections to span multiple time steps. This relaxes the Markov assumption in a similar way to Variable

Order HMMs. Hu and Yang (2008) use a two-layered Skip-Chain CRF to recognize multiple interleaved and concurrent goals. The lower Skip-Chain CRF considers each goal independently. Skip edges are learned from the corpus to connect actions which contribute to the same goal. This allows goals to be interleaved within the observation sequence. A correlation graph forms the second level, which measures similarity between goals for recognition of concurrent goals.

Markov Logic Networks (MLNs) are a recent trend which allows for the combination of probabilistic and logical inference. Singla and Mooney (2011) use MLNs to model the environment for abductive reasoning. The approach transforms the knowledge base from a deductive inference pattern to an abductive inference pattern by reversing implications. The Hidden Cause model introduces additional nodes to capture multiple possible explanations for goals from the network. This allows for more efficient reasoning by capturing abstractions in the underlying graphical model structure.

MLNs can also be used as a classification framework. Ha et al. (2011) extended prior work (Mott et al., 2006) to use MLNs to capture undirected dependencies for exploratory actions. As with Mott et al. (2006), Ha et al. (2011) created a model which factors the observations into separate random variables. In addition to the factored formulae, several additional features were included to allow for joint reasoning over multiple observation types.

When applied to noisy data, such as GPS locations (Sadilek and Kautz, 2012), MLNs can jointly denoise the data and perform classification for goal recognition. The model uses the MLN framework to assign raw GPS values to cells, which are then used to model the rules of a Capture the Flag game. The network can then build on the knowledge of the player's location within a cell to predict whether they were attempting to capture or free another player. The model also used a structure learning algorithm to extend

reasoning to failed attempts at the various goals.

There are two interesting trends in previous approaches that have influenced the work in this dissertation. First, several past systems have explored capturing additional state information (Pynadath and Wellman, 2000; Geib, 2009; Geib and Goldman, 2009; Gold, 2010). Many of these approaches, however, restrict the context to the effects of actions within the state. The second trend is that several approaches (Armentano and Amandi, 2009; Hu and Yang, 2008) have explored relaxing the Markov assumption. When interacting with a problem-solving environment, the user gradually builds a solution towards the goal. While the Markov assumption is useful for combating data sparsity, the additional information helps the models become more accurate. This work extends these previous approaches by expanding the context representation to capture a sense of progress for the user. These milestone events allow the system to capture limited information about the prior sequence.

One of the features of these large interactive narrative environments is the users have plenty of opportunities to explore. This creates long, noisy plans. This necessitates the use of a probabilistic representation. MLNs have shown to be effective on noisy data sets (Sadilek and Kautz, 2012), and have achieved a state of the art performance on the CRYSTAL ISLAND: OUTBREAK data set described in Chapter 5. For this reason, MLNs were chosen for the base representation. The expressivity of MLNs allow them to capture many of the same phenomenon as other graphical representations. One of the benefits from MLNs for the Discovery Events representation is that it allows a relaxation of the Markov assumption. When representing discovery events, it is possible to capture the history without reprocessing the data.

2.2 Narrative

Computationally modeling narrative structures poses significant representational challenges. Narrative is a complex phenomenon, which covers a large variety of artifacts. Over the years, many representations have arisen that focus on different aspects. Several common themes include reasoning about events over time, character behavior, and adaptive representations for interactive narratives.

A key component of narrative is reasoning about time. Time can be represented explicitly or implicitly. Many of the explicit representations are based on interval logic (Allen, 1981). Interval logic defines six modal operators that can describe the relationship two events have to each other. The TimeML markup language (Pustejovsky et al., 2003) extends interval logic for natural language reasoning. Other systems capture time by imposing a (partial) ordering on the events observed.

Extensions of the TimeML system for temporal reasoning have been examined for narrative event reasoning (Chambers and Jurafsky, 2008, 2009). The narrative event chains model extracts event information from texts and analyzes co-occurrence information, along with temporal reasoning supported by TimeML, to discover likely sequences of narrative events (Chambers and Jurafsky, 2008). However, narrative event chains impose representational restrictions by assuming chains involve only a single protagonist. The narrative schemas model extends narrative event chains by allowing for multiple protagonists and general protagonist roles (Chambers and Jurafsky, 2009).

Story threads (Gervás, 2012) are a very similar representation used for narrative generation. Each thread is a sequence of events that are observable by a single protagonist. By focusing on different characters throughout the narrative, these threads are woven together to form a narrative yarn. Instantiating the woven threads allows the system to

generate narratives following multiple points of view.

Story curves (León and Gervás, 2011) instead represent a narrative as a curve through a narrative space. These trajectories are used for language generalization by selecting events that evaluate to the curve. The dimensions of the curves are set by the system generating the narrative.

Within the computational linguistics community, it has become popular to represent events in terms of semantic frames (Manshadi et al., 2008; Bejan and Harabagiu, 2010). Semantic frames are a description of a type of event, relation, or entity and the participants in it. When understanding a more goal-oriented text, such as how-to articles, explicitly representing the goals, actions, and context has been shown to be effective (Jung et al., 2010). Analogical reasoning has also been used to sequence frames for generation (Ontanon and Zhu, 2011). An alternative from the linguistic community is to use rich models of discourse, such as Rhetorical Structure Theory (RST) (Nakasone and Ishizuka, 2006). Storytelling Rhetorical Structure Theory uses RSTs to represent actions within the narrative, but augments this knowledge with a concept ontology and a structural ontology of acts and scenes.

The Restaurant Game (Orkin et al., 2010) utilizes a model based on scripts (Shank and Abelson, 1975) in order to represent an interactive scenario in restaurants. Interactive gameplay traces are annotated using possibly-overlapping task labels. A Hierarchical Task Network (HTN) is learned using the annotations. The network can then be recognized by a sequence alignment algorithm for task recognition.

More recently, the Scheherazade system (Li et al., 2012) developed a plot-graph approach which can be mined from simple stories using a restricted language input. These plot graphs are clustered which is designed to support interaction by reasoning about which events cannot take place within the same narrative.

Plot units (Lehnert, 1981) describe a plot as a sequence of discrete states, which are tagged with affective information. The representation was developed for summarization, since adjacent plot units with the same label could be merge. This was later extended to include mental, as well as affective, states (Mani, 2004). Alternatively, Plot Units can be used to model reader reactions to events or characters within a narrative (Mani, 2010).

Attempts have been made to learn plot units from text. Conditional Random Fields capture temporal dependences between events to learn plot units (Appling and Riedl, 2009). Sentiment analysis, along with a rule-based production system to capture transitions, can also be used to recognize affective states (Goyal et al., 2012).

Several frameworks have also been developed specifically for annotation of large corpora of text. *Story Intention Graphs* represents events on three layers: the propositions which describe the story world, text spans which relate to these propositions, and the beliefs, intentions, and affective state of the characters (Elson, 2012a,b). *Story content units* have been developed to capture children’s’ retelling of narrations over the course of several days (Passonneau et al., 2007). The *Story Workbench* (Finlayson, 2011) instead offers many different representations which can be used within the annotation tool.

Emergent narratives arise from naturalistic interaction between virtual agents. Representing the narrative structure in these cases often involves using models of agent belief, desire and intention in order to guide virtual characters. Such systems often need to employ some other form of structural knowledge, such as constraints (Porteous et al., 2010; Meech, 1999), interactive drama managers (Theune et al., 2003), or planning (Si et al., 2008; Riedl et al., 2003), in order to guide characters’ behavior.

Morphological approaches are largely based off of the morphology of Vladamir Propp (1968). Within an interactive system, the morphology can be used to guide which content is selected next to display to the user (Grasbon and Braun, 2001). It can also be used for

narrative generation by supplying each function with textual skeletons (Peinado et al., 2004). Case-based approaches can match events from Propp’s morphology to the interactive scenario (Fairclough and Cunningham, 2003). Analogical reasoning has also been used to recover the morphology, which can then be extended to other genres (Finlayson, 2009).

Some representations also seek to explicitly model the causal and temporal relationships between events in the narrative. Case-based reasoning (Swartjes et al., 2007) can use creativity metrics to guide the generation process. Planning systems can leverage partial ordering among events in order to maintain a consistent story (Magerko, 2005). Mediation trees (Riedl and Young, 2006) allow the system to travel between plans with partial ordering and causal information in order to react to the users actions.

Rather than model the world states, some approaches instead model actions the system can take in order to repair the narrative structure. Search-based approaches (Weyhrauch, 1997) explore the space of possible stories, but may not generalize to all narratives (Nelson and Mateas, 2005). *Drama Managers* can guide the characters dialogues along dramatic beats (Mateas and Stern, 2005), guide the conflict towards interesting dilemmas (Barber and Kudenko, 2007), or model the distribution of interesting narratives to provide unique experiences (Roberts et al., 2006).

There are several limitations with previous work within the context of goal recognition. Narrative structure serves to contextualize the actions the player has taken so far. Generative models, such as the emergent narrative approaches, are not always well suited for recognition. Plan-based approaches to interactive narrative are very synergistic with goal recognition. The goals recognized could aid with proactive mediation. However, the representation of narratives in such systems is primarily encoded in the causal relationships between events, and the preconditions and effects of the various actions the

user may take. In addition to a significant knowledge engineering effort, it is not always clear how to extract the narrative context from the current world state unless additional fluents are injected.

The event-based representations are often well suited to the task. However, many of these are targeted to the natural language domain. In an interactive game, the data may not carry all the context that natural language does. Some features may be easily mapped, such as the protagonist of a narrative, but features such as protagonist affect are not easy to discover in a non-invasive manner. Scripts have been used in a goal-recognition context before (Orkin et al., 2010), but require a significant annotation effort.

The representation presented in Chapter 4 is most closely related to Plot Units and Beats (Mateas and Stern, 2005). Rather than focus on character affect or units of drama, Discovery Events present a framework for realizing several notions of plot progression.

Chapter 3

MLN-based Goal Recognition in Interactive Narrative

Goal recognition and its sibling tasks of plan and activity recognition have been under active study within the Artificial Intelligence community for several years. Goal recognition, sometimes called intent recognition, is the task of recognizing an agent's intent through observation. Plan recognition, in contrast, seeks to understand the agent's plan, as well as the goal. Activity recognition is closer to goal recognition, in that it attempts to determine how individual observations relate to the overarching task.

3.1 Task Definition

Goal recognition uses information about the agent, such as actions taken within the environment, to predict the goal the agent is trying to achieve. Within interactive narrative systems, the agent is taken to be the player. The observations may include actions taken within the virtual world, location of a virtual avatar, or features of the state of the virtual

world. Formally, goal recognition is defined as follows. Given a sequence of observations and a set of goals, predict the goal that the observed agent is attempting to accomplish.

Within interactive narrative experiences, there are particular challenges that need to be considered. In some interactive systems, the goals are not communicated with the player. In these sorts of environments, players may engage in exploratory behavior. Some of this exploratory behavior will reveal the next goal in the process. However, when such exploratory behavior occurs, the players do not have a well-formed plan that is suitable to many goal recognition techniques. They may not even be aware of a goal until it is achieved. This creates an ambiguous causal relationship between the actions observed and the goals the player is trying to achieve.

A second challenge is that the players may make progress towards many goals at once. This may be intentional, indicating multitasking behavior, or it may be a result of the exploratory actions the player is taking. Actions may also serve multiple goals, such as moving into an area in which multiple goals may be accomplished. This may cause an ambiguous relationship between the actions and goals, which can be difficult for the goal recognition system to correctly model.

The goals are also not independent of one another. Some goals will need to be completed before others are available to the player. In interactive narrative experiences, the goals that the system would target for recognition are often milestones that the player must accomplish during the course of the narrative. Even among goals which have no order restrictions, proximity within the virtual environment can influence which goals are completed next.

In order to address these challenges within an interactive narrative, we make several simplifying assumptions. In order to address the ambiguous relationships between actions and goals, both because of player exploration and because of the possibility of multiple

interleaved goals, we selected an undirected probabilistic model. The probabilistic model allows for uncertainty in how predictive an observation is of the associated goal. Since the model is undirected, it also is robust to changes in how the actions and goals inform each other. However, to use a probabilistic model, we assume that the distribution among goals and observations is fixed across players.

The next two simplifying assumptions address the complexity of annotating the observations. In order to address interleaved goals, we assume that players pursue a single goal at a time, and that the completion of a goal serves as a proxy for the player’s intent. The goals are never explicitly presented to the player, and prompting the player would be an invasive process. Under these assumptions, the data can be automatically annotated by scanning for goal completion points, and extending the goal backwards over all the actions that lead up to it.

Lastly, we make a common assumption for goal recognition systems, called keyhole goal recognition. Under this assumption, the players are unaware that their actions are being monitored by the system. They do not take any measure to either obfuscate or elucidate their behavior. This simplifies the models, since it creates the strongest association between the observations and the intended goals.

3.2 Approach

Goal recognition is well suited to classification when the set of goals is known in advance. In plan recognition, complicated reasoning is often required to align the observations with the plan library. In contrast, goal recognition emphasizes the associations between sequences of observations and the goals. Machine learning classifiers capture these associations to learn a predictive model. When viewed as a classification problem, the

observations are processed to form the features of the classification instance. The class that is assigned to the set of features is the goal that the player was attempting to accomplish at that moment.

One limitation of this approach is that sequences of observations may be sparse within the data, especially if the data is drawn from real-world interactions, instead of simulation. A common way to address this is to use a Markov assumption. Under a Markov assumption, each observation is assumed to only depend on a finite number of prior observations. In this framework, each observation is transformed into a set of features, which may refer to previous steps, for a machine learning classifier. The classifier then predicts a goal based on the set of features.

Within an interactive narrative, observations tend to be features of the game environment. One of the most natural observations is the action the players choose to take. The location within the game world that the action is performed, along with any objects involved in the interaction, may be useful indicators of the context in which the action is applied. Other features of the environment, such as the location of some salient game objects or progress within the game’s narrative, can also be used. A full description of the game features used in this work will be presented in Section 3.5.

Once the features are selected, the goals must be associated with the observations. The first step is to consider which goals will be represented. Within a problem-solving environment, good candidates for goal selection are milestones that lie along the critical path to solving the problem. The goals should also be recognizable within the environment. This can prevent a significant manual annotation process, or an invasive prompt which requires the players to enter their goals.

Given a corpus of observations, such as the one described in Chapter 5, the goals can be annotated through the following process. First, the players’ observation sequences are

scanned for points in which goals are achieved. These observations are annotated with the goal. Since we assume there are no interleaved goals, every remaining action is annotated with the next goal observed. Lastly, the original goal-achieving observations are removed, since it would be trivial to recognize the goal from the goal-achieving observation.

3.3 Applications

Within the context of an interactive narrative experience, there are several applications for goal recognition. In a broad sense, accurate goal recognition would allow the system to react in a disciplined way to the actions the player takes within the environment.

In interactive narrative experience, goal recognition could be used to inform various adaptations. It could allow a director agent to control the pacing of elements within the story. For example, consider an interactive mystery in which the narrative's pacing depends on how clues about the mystery's solution are revealed to the player. If a director agent detects that the player is likely to skip an important clue, thereby jumping to a later point in the story, it can preemptively adjust the narrative's pacing by altering the clue's placement in the virtual environment. Alternatively, if the director agent recognizes that the player is pursuing a "red herring" that would disrupt the narrative's pacing, it could deliver pointed clues to lead the player back to a more promising path. The system could instead mediate narrative conflict, choosing an appropriate conflict for the player's current course of action, or personalize the story experience based on information about the player.

Instead of mediating the interaction, the system could instead support the player. Such models are of particular interest within Narrative-Centered Learning Environments. By monitoring the player's goals, the system could support the player's problem-solving

process. It could offer appropriate hints and feedback to make sure the player is able to learn from the interactive experience. Alternatively, it could be used to test the player’s problem-solving mastery through the same monitoring process.

Lastly, an analysis of the goal behavior of a player could be used as a post-hoc analysis. This analysis could be used by software designers in an iterative design process. The software designers can use the recognized goals to inform decisions about which areas of the environment to expand, which content needs to be generated, and which areas need to be made more enticing to the player.

3.4 Markov Logic Networks

Markov Logic Networks (MLNs) are a relatively new graphical modeling framework (Domingos et al., 2006). A graphical model is a compact method for representing the joint probability distribution over a set of random variables. The model is an extension of Markov Networks. A *Markov Network*, or *Markov random field*, is a graphical model where each node represents a random variable, and each undirected edge represents dependency. The joint probability function is represented as the product of *potential functions*. These potential functions assign probabilities over cliques in the underlying graph structure.

Markov Logic Networks can be used for statistical relational learning. Statistical relational learning models domains that exhibit both uncertainty and complex relational structure. This invites methods leveraging both probabilistic and first-order relational inference. MLNs are expressed through a set of first order logical formulae along with a set of real-valued weights. These weights offer some intuition for how deterministic the relationship is. As the weights approach infinity, the reasoning approaches first order

logical deduction. The Markov network is generated from the variables appearing in the clauses of the first-order formulae. Each variable is its own node, and the logical connectives define the edges, with potential functions derived from the number of true groundings of each clause.

3.4.1 Inference in MLNs

One of the canonical tasks for Statistical Relational Learning is to find the most probable world state, given some evidence. This is known as finding the Maximum a Posteriori estimator (MAP) or the Most Probable Estimator (MPE). Within MLNs, MAP inference can be reduced to Weighted MAX-SAT (Domingos et al., 2006). MAP inference subsumes both probabilistic reasoning and logical deduction. There are several algorithms that have been developed to support this inference. The MaxWalkSAT algorithm (Kautz et al., 1997) serves as the basis for most of the more specialized algorithms which have been developed for Markov Logic Networks in particular. MaxWalkSAT is a local search strategy that will either flip a random variable from true to false, or flip the variable that minimizes the weight of the remaining unsatisfied clauses.

MaxWalkSAT operates on the fully grounded Markov network, which introduces some memory limitations on large networks. A modification of the MaxWalkSAT algorithm, called LazySAT (Singla and Domingos, 2006) was proposed. LazySAT makes use of the sparsity of the ground network by only considering clauses which may become unsatisfied while performing the search. This allows it to only ground the portions of the network that are required for performing the inference.

If the graph structure has relatively few cycles, Lifted Belief Propagation (Singla and Domingos, 2008) can be used instead. Lifted Belief Propagation is a first-order reasoning algorithm which relies on message passing from variable nodes to nodes representing the

clauses in which the variables participate. The Lifted version abstracts the grounded network into a first order network by combining nodes and features that send the same messages. Belief propagation is then performed on the lifted network, allowing for inference.

If it is possible to separate the observable evidence variables from the hidden query variables, Cutting Plane Inference (Riedel, 2008) can be used on top of the MAP inference for greater efficiency gains. It iteratively adds violated formulae to the knowledge base while solving groundings of the initial problem. Any other MAP inference algorithm may be used to solve each of the subsequent groundings.

Another inference task that can be performed over Markov Logic Networks is to find the probability of a given clause. This can be computed using a sampling algorithm, such as Markov Chain Monte Carlo (MCMC) algorithms. This approach however breaks down in the presence of near-deterministic constraints. When this is the case, MCMC can be combined with a SAT solver to perform inference. MC-SAT (Poon and Domingos, 2006) samples auxiliary variables representing each clause to determine if the clause needs to be satisfied on the next iteration. The next state is then sampled from the set of states that satisfies the selected clauses.

3.4.2 Learning Markov Logic Networks

Learning in a MLN falls under one of two categories: learning the weights for a given set of formulae, or learning the formulae for a given set of evidence. The weight learning task takes as input a set of formulae depicting a MLN and a relational database comprising of the evidence. Generative weight learning uses a closed-world assumption, all groundings not explicitly in the database are assumed to be false, and seeks to maximise the probability of the database.

Discriminative weight learning (Singla and Domingos, 2005) can be applied in cases where it is known a priori which predicates are observable evidence, and which will be hidden query predicates. By separating the predicates, the Conditional Log-Likelihood can be estimated instead. MAP inference can be used to efficiently estimate the expected number of groundings involving the query predicates.

Learning the structure of an MLN can be performed using standard Inductive Logic Programming (ILP) techniques. Evaluation is performed by a weighted pseudo-log-likelihood, which assigns a weight to each first order predicate based on the user’s goals (Kok and Domingos, 2005). Both Beam search and a shortest-first search that explored shorter clauses before longer ones were used in evaluation.

3.4.3 Tools for Markov Logic Networks

Two software tools have been released for learning and inference with MLNs. Alchemy (Kok et al., 2007) is provided by the University of Washington, and is considered the official software package for MLNs. It is the most complete implementation and provides many of the algorithms described in Section 3.4.1 and Section 3.4.2. It does not, however, support Cutting Plane Inference because of the syntactic limitations on the network.

The other software package is Markov: TheBeast (Riedel, 2008). TheBeast is the only toolkit that supplies Cutting Plane Inference (see Section 3.4.1 for a discussion of Cutting Plane Inference). Cutting Plane Inference performs faster on many domains. The software does not support Structure Learning, and only uses Integer Linear Programming or MaxWalkSAT for the base MAP solver.

For this dissertation, we chose Markov: TheBeast for learning and evaluating MLNs. Markov: TheBeast has been successfully used within the Natural Language Processing community (Ha et al., 2010) as a classification framework. It was also the implemen-

tation chosen for prior work in goal recognition on the CRYSTAL ISLAND: OUTBREAK corpus (Ha et al., 2011). Preliminary explorations with the Alchemy toolkit encountered runtime and memory issues due to the size of the grounded network. The structure of the classification task, as described below, is a good fit for the efficiency gains within Markov: TheBeast.

3.5 Goal Recognition

While MLNs are well suited to different types of reasoning, implementing goal recognition requires some special considerations. With graphical models, each node in the graph represents a random variable. In order to represent goal recognition using a graphical model, one or more random variables must be used to capture the goals. Within MLNs, the predicates that capture the goals are implemented as hidden predicates. Hidden predicates are inferred from the groundings observed for the remaining predicates using the methods described above.

One of the assumptions of machine learning is that instances are independent and identically distributed. This causes some problems in goal recognition for narrative because a single player’s actions may not be independent of each other. The players are, however, largely independent of each other. For this reason, each classification instance is an entire transcript from a single player. In order to make predictions at every time step, each predicate is given an extra time step argument. Different groundings of the predicates indicate the time step on which they occur.

Classification within MLNs is similar to other probabilistic graphical models. A random variable, or predicate in MLNs, is created for the predicted classes. Within MLNs, the first order representation allows for a single predicate to be used. When

inference is run on a set of observations, groundings for the hidden predicate are predicted. Since each play trace is a single instance, groundings for the hidden predicate for each time step are predicted. Formulae were included to ensure that only one goal was predicted for any given time step. In this work, goals are represented by a single *goal* predicate. Each goal is represented by a constant symbol. By using a single goal predicate, the model can leverage the assumptions to improve the accuracy.

Actions have two components within the representation. The *action* predicate represents the actions being taken by the player. As with the goals, each available action is a unique constant symbol. Some actions may also take an argument. The *argument* predicate takes an argument type, such as an object in the world or a topic of conversation, and the argument value. Argument types and values were also represented by constant symbols.

The *location* predicate behaves similarly to the previous predicates. While individual coordinates within the virtual environment could be used, it would create very sparse data. Instead, a grid-based system was used to segment the environment into cells. These cells may be communicated to the player through an in-game map, or highlight salient points within the environment. This reduces the number of groundings the MLNs would have to reason over, and allow distinctions between cells to be more meaningful when contextualizing the observations.

The last feature used for the current work is narrative progress. There were two different representations used. One is a milestone-based formula construction, which will be detailed below. The other is a *state* predicate which captures the larger form of the narrative. The state is represented internally by a bit vector. This bit vector is turned into an integer number before being grounded in the predicate. Each bit in the narrative state represents a significant milestone in the narrative. These could be comparable to

acts in a play or movie, or points in which the expected behavior changes, such as the reveal of a big clue. There may be some overlap between the states and the goals, but the goals are generally more fine grained to be of use.

When extracting these features from a corpus of observations, first a new possible world instance is created for each player. Within each instance, each observation for that player is processed to extract the predicate arguments. These are collected into a sequence in the same order that they were observed. These are indexed according to the sequence of observations, and a grounding is created for each predicate and its arguments at each time step. Within the PML input language, each predicates groundings are listed separately. Table 3.1 shows each of the predicates and their arguments.

Table 3.1: List of predicates for goal recognition

Predicate	Semantics
$action(t, a)$	Observed player performing action a at time t
$argument(t, a)$	Object a was involved in the player's action at time t
$location(t, l)$	Observed player at location l at time t
$state(t, s)$	Player has passed the milestones for state s sometime before time t
$goal(t, s)$	Player is attempting goal g at time t

While Cutting Plane Inference can be much faster under the right circumstances, TheBeast is not a complete implementation of MLNs. There are some syntactic limitations that are imposed on the formulae. Cutting Plane Inference requires that the hidden predicate (goal) is separated from the observed predicates (actions, arguments, locations and state). There are also some operators that have incomplete implementations. The existential operator is not supported. Negation is only supported if negative atoms are observed, and disjunction is experimentally supported in the latest subversion revision.

$$\forall a, g, t : \text{action}(t, a) \Rightarrow \text{goal}(t, g) * w_1(a, g) \quad (3.1)$$

$$\forall l, g, t : \text{location}(t, l) \Rightarrow \text{goal}(t, g) * w_2(l, g) \quad (3.2)$$

$$\forall s, g, t : \text{state}(t - 1, s) \Rightarrow \text{goal}(t, g) * w_3(s, g) \quad (3.3)$$

Figure 3.1: Single predicate formulae in MLN.

Within TheBeast, a formula has three parts. First, there is a list of universally quantified variables. Next, the antecedent is composed only of observed predicates. Finally, the consequent is composed of a formula over hidden predicates and a weight function. The variables are used to compute groundings of the observed formula that occur in the corpus. The observed groundings are then used to predict the groundings of the hidden formula.

There were two types of formulae constructed for this work. In the first, the antecedents are clauses over the observations. Universal variables are used to indicate both the time step and the value of the observations. The simplest formulae are built by using a single observed predicate as an antecedent. Examples are shown in Figure 3.1. Both the current time step and previous time step can appear in the antecedent, to satisfy the first-order Markov assumption. More complicated formulae may involve multiple observed predicates. Examples are shown in Figure 3.2. This allows the ground network to capture information about the interaction between two observations, in so much as they explain the goal.

The second type of formulae are milestone predicates. Milestone predicates attempt to capture information about earlier actions in the sequence. The cardinality constraint allows formulae to capture how many groundings of a given formula are present in the world. As mentioned above, an entire student’s play trace is a single instance of the

$$\forall a, s, g, t : action(t, a) \wedge state(t, s) \Rightarrow goal(t, g) * w_4(a, s, g) \quad (3.4)$$

$$\forall a_1, a_2, g, t : action(t-1, a_1) \wedge action(t, a_2) \Rightarrow goal(t, g) * w_5(a_1, a_2, g) \quad (3.5)$$

$$\forall a, g_1, g_2, t : action(t, a) \Rightarrow [goal(t-1, g_1) \Rightarrow goal(t, g_2)] * w_6(a, g_1, g_2) \quad (3.6)$$

Figure 3.2: Multiple predicate formulae in MLN.

$$\forall t, g : |\forall t_2 < t : action(t_2, \text{“ACTION”})| \geq 1 \Rightarrow goal(t, g) * w_7(g) \quad (3.7)$$

$$\forall t, g : \left| \forall t_2 < t : \begin{array}{l} action(t_2, \text{“ACTION”}) \wedge \\ argument(t_2, \text{“ARG”}) \end{array} \right| \geq 1 \Rightarrow goal(t, g) * w_8(g) \quad (3.8)$$

Figure 3.3: Milestone predicate formulae in MLN.

world. Milestone predicates leverage this by searching for groundings in the previously observed time steps. These predicates relax the Markov assumption, but may be used to indicate progress within the narrative. Due to sparsity concerns from relaxing the Markov assumption, they must be used sparingly. Chapter 4 describes the selection of candidate milestones. Examples are shown in Figure 3.3.

3.6 Summary

This chapter presents the problem of goal recognition, along with the proposed approach. Goal recognition, within the context of an interactive narrative experience, is the process of observing a sequence of player actions within the environment and determining which of a set of goals are they most likely to be pursuing. Within exploratory domains, such as

interactive narrative experiences, there are additional simplifying assumptions that must be made. We assume that each player is only pursuing a single goal at a time, that the distribution between observations and goals is fixed across players, and that the player is not aware that the system is monitoring their goals.

When the set of goals is known, goal recognition is well-suited to representation as a classification task. In classification, each observation is extracted to form a set of features which can be labeled as a goal. For this work, we use Markov Logic Networks, a statistical relational learning framework, to perform classification. Within MLNs, each player's observations are considered independently of the other players. The classifier predicts groundings for each of the instances of the *goal* predicate over time.

Input for the MLN is a set of first order logical formulae, along with weights. In order to represent classification within MLNs, three types of formulae were presented. The simplest formulae leverage a single predicate as a predictor of a goal. More complicated formulae can express conjunctions over observed predicates as a predictor of the goal. This work extends these with the inclusion of Milestone formulae, which allow the system to relax the Markov assumption and capture events which happened earlier in the sequence. These will be used to capture the enhanced notion of narrative which is presented in Chapter 4.

Chapter 4

Discovery Events

Narrative is a complex phenomenon. It can encompass any number of facets, each serving any number of purposes within the story. It can be driven by complex characters, by a strong sense of plot, or even be a true account of real-world events. Literary narratives and narratives crafted for entertainment are often driven by models of conflict or tension, or may be designed to follow some aesthetic guidelines. One element that is often seen to be underlying many different forms of narratives is the idea that events are taking place across time. The passage of time is conveyed within the narrative by creating a sense of progression towards the ending.

Over the years, different ways of structuring this sense of progression have been explored. In the classical three-act and five-act structures, shifts in dramatic tension mark key stages of plot progress. In computational models of narrative, several measures of progress have been investigated. Some narrative representations denote transitions between affective states (Lehnert, 1981). Other narrative representations focus on characters, such as their beliefs, goals, or conflicts between them (Ware et al., 2012). Logical representations, where narrative is conceptualized as the plan that transitions between

world states, have been investigated extensively (Elson, 2012b; Riedl and Young, 2005).

4.1 Narrative in Goal Recognition

In a structured problem-solving environment, the users take steps towards their eventual goal. As they explore the environment, a narrative of their actions emerges from the observations. In interactive narrative experiences, this sense of narrative is strengthened through its connections with the presented narrative. Consider an interactive medical mystery story, where the players are trying to uncover the source of an infection. As they take actions, such as inspecting the various objects or speaking with virtual characters, the system gives them information that is contextualized in the medical mystery scenario. This information structures the task and provides important context within which the actions are performed.

As an example, consider a situation where a player enters the infirmary and reads one of the books on the shelf. There may be several goals in the Infirmary, such as speaking with the nurse about the mystery or speaking with a sick patient about her symptoms. Even with two actions and the location, it is difficult to predict which goal is intended. However, if the system knows that the player has already spoken with the camp nurse and has already determined the transmission vector of the infection, then speaking with the sick patient is more likely.

While this rich context is being co-created with the player, the system often is unaware of what has happened before. The loss of context is primarily due to the Markov assumption. The player's actions are generally not independent of each other during the same session. The Markov assumption alleviates some of the sparsity concerns, but other means must be used to capture the goal recognition context. Since the task context is

closely tied to the encompassing narrative in an interactive narrative, narrative structure offers a natural choice for capturing this context.

There are several criteria for representations within a goal recognition framework. In order to support the goal recognition process, our representation must satisfy several criteria:

Recognizable: The representation must afford the system a way to recognize the structure from the observable actions. The goal recognition system would only have limited knowledge available about the narrative structure.

Adaptable: The representation must make allowances for the player’s actions. Goal recognition is most beneficial in environments where the player’s actions can have significant impact on the environment rather than in a linear narrative with no player input.

Granularity: Large macro-level structure can be very informative for establishing context for the actions performed. However, if the unit of representation is smaller than the goals, the system will be able to leverage them as building blocks for the goals they lead up to.

Many generative models are unsuited for the task. While they may be able to capture rich information about player adaptations, they don’t always provide a suitable mechanism for recognition. Recognizing the narrative situations is an essential component of goal recognition.

A functional representation, such as Propp’s morphology (Propp, 1968), could provide much of the framework required. In a similar vein, the Story Intention Graph representation (Elson, 2012a) specifically encodes intent of the various events that take place in the narrative environment. The drawback is that this information often is not

readily available within the narrative environment. Learning these models would require extensive annotation effort.

Structural representations also offer some attractive properties. Prior work on goal recognition in an interactive narrative environment (Ha et al., 2011) used a measure of the progress along the narrative as an observable feature for the goal recognition task. The features used in the prior work are very closely related to an Aristotelian Three-Act structure. While this approach can offer some insight into the context into which the goals are being accomplished, many of the goals of interest to a system designer may occur in multiple acts.

This work draws inspiration from several previous representations. Similar to Plot Units (Lehnert, 1981), discovery events focus on transitions between protagonist states. We also draw inspiration from our previous work on the narreme (Baikadi and Cardona-Rivera, 2012) and Story Curves (León and Gervás, 2011), as discovery events can describe trajectories through a narrative space. However, discovery events are novel in that they model the sense of progression from the player’s point of view by focusing on which events present information to the player.

4.2 Defining Discovery Events

Discovery events capture the structure of a narrative by representing progress towards answering a central question. In many narratives, the central question drives the narrative events that move the story forward. Central questions can revolve around the nature of the setting, or some fact about the world, what characters will do or have done, or what the consequences will be of a major event (Card, 1988). For example, the Lord of the Rings trilogy asks, among other things, “What lies beyond the Shire?” Murder mysteries

often focus on the question of “Who did it?” Romance stories ask “Will they, or won’t they?” Many disaster stories ask how the world will change after some great disaster.

Partial answers to the central question structure the narrative. These *discovery events* provide milestones along the way where players can feel the sense progression as they move closer to the eventual goal. The narrative context can be conceptualized as the sequence of discovery events experienced by the player so far. More complicated narratives may have several questions raised during the course of the interactions. In these cases, it’s also possible for an event to provide a partial answer to multiple questions.

There are several types of questions and associated discovery events that can occur during an interactive narrative. In his book of writing advice, *Character’s and Viewpoint*, popular science fiction author Orson Scott Card (1988) wrote about his MICE Quotient. The MICE Quotient describes the four basic types of questions that are often asked in narrative: Milieu, Idea, Character or Event.

Milieu: The Milieu story is about the physical location of the story as well as the cultural context and other setting details. Examples of a Milieu story are travelogues, *Dune* by Frank Herbert, and the *Lord of the Rings* trilogy by J. R. R. Tolkien. Discoveries include facts about the setting and the influences that affect the events.

Idea: Idea stories are about a question or a problem. The classic examples of Idea stories are mysteries and detective stories. Discoveries are clues, including red herrings that may mislead the player.

Character: A Character story is about a person trying to change their place in life. Discoveries are lessons that the player learns along the way or revelations as to their nature.

Event: An Event story is about changes in the world, that must be either restored

or adjusted to. Disaster stories fall into this category. Discoveries include the challenges that the player faces during the course of the event.

A type of character question that has been of interest to the narrative community is character affect. Plot Units (Lehnert, 1981) can be used to ask how the main character’s emotions are changing over time. In an interactive narrative, it could be useful to infer the player’s affect. This would allow the system to help alleviate states such as boredom or frustration. In this case, discoveries are the events that prompt the change of affect, in either the player or character.

Within an educational system, or even during a mystery story, the character’s knowledge can be an important component. Knowing which clues have been presented to the player is an essential part of controlling the pacing and to avoid inconsistencies in an adaptive mystery story. As with Idea stories, the discoveries are knowledge components that the player must gather.

Another common view on structuring narrative is according to the conflict. Computational representations such as Beats (Mateas and Stern, 2005), Dilemmas (Barber and Kudenko, 2007), and conflicting plans (Ware et al., 2012) capture the intuition behind classical representations, such as the classical three-act and five-act story structures. In these situations, the central question is “Will the protagonist achieve his goal, despite the actions of the antagonist?”, and discoveries along the way are challenges to overcome.

Discovery events may also extend beyond fictional narratives. Within the goal-recognition context, the central question is instead the overarching task the user is trying to accomplish. Discoveries in these cases are actions the users may take that uncover information or solve a subproblem. Using the milestone formulae from Section 3.5, the sequence of discovery events can allow the system to reason about which information the players have, or what they have attempted in order to solve the goal.

4.3 Example Story

Consider the well known children’s story of Goldilocks and the Three Bears as told from the point of view of the bears. In this story, three bears come home from a walk to find that someone had been inside their home. What follows is a mystery story in which the bears try and ascertain who has been in their house. They first start at their porridge. The Mama bear and the Papa bear both discover that their porridge has been tasted, but the Baby bear discovers that his porridge has been finished. Next they look in their living room. Again, the Mama and Papa bear discover that their chairs had been sat upon. Baby bear, however, discovers that his chair has been broken. Lastly, they check the bedroom. Both the beds of Mama bear and Papa bear have been slept in, but Baby bear solves the mystery by discovering Goldilocks still napping in his bed. The story ends as Goldilocks makes her escape.

The three bears experience nine different discovery events during the course of their investigation. First, they observe the three porridge bowls. Along with the temperatures (too hot, too cold, and just right), the bears are given a clue to the tastes of the person. Next, they observe the chairs. Along with the sizes (too big, too small, and just right), the bears are given a clue to the size of the person. Lastly, the bears observe the three beds, and are able to solve the mystery. In the majority of retellings, these nine events are kept stable while many other aspects, such as the reason for Goldilocks’ intrusion in their home, and the eventual means of her escape, are often left to creative interpretation. In this sense these discoveries capture the core structure of the story.

4.4 Summary

This chapter presents discovery events, the narrative representation developed for this dissertation. Within a goal-recognition context, the actions the players take within the environment form a narrative as they progress towards a goal. Within narrative-centered learning environments, this sense of narrative is strengthened as all information the system delivers is delivered through the narrative scenario. For a narrative representation to aid goal recognition, it must be recognizable from the observations, it must be able to adapt to player actions, and should be at a level of abstraction that allows the representation to serve as building blocks towards goals.

Discovery events represent the narrative as a sequence of partial answers to a central question. These central questions drive the events that occur within the narrative. Within a goal-recognition context, the central question is often related to the task the player is attempting to accomplish. Milestone formulae can be used to capture discovery events within the model, and allow the system access to the progress the player has made towards the goal.

Chapter 5

Corpora

When authoring Markov Logic Networks (MLNs), as described in Chapter 3, there are two steps which must take place. First, the formulae must be authored to capture the relevant dynamics. Second, weights must be assigned to each formula. These weights may be authored directly, or learned from a corpus. This dissertation follows the more common approach of learning the weights from a corpus. This allows us to achieve a higher fidelity model of goal-seeking interactions within an interactive narrative experience.

When selecting a corpus of observations, there are several features to consider. First, machine learning techniques assume that the observations are independent and identically distributed. These techniques attempt to capture knowledge about the underlying distribution in order to form valid predictions. Secondly, there must be sufficient observations to learn a model. How many observations are required will vary depending on the complexity of the model. However, it is beneficial to collect as many observations as possible in order to ensure sufficiency of data.

For the problem of goal recognition with the discovery events model proposed in this dissertation, there are additional features to consider. While simulation is occasionally

useful for generating large corpora, human interactions are a complex phenomenon. For this reason, corpora for goal recognition benefit observing human participants interacting with the target system. Secondly, discovery events capture the context in the presence of exploratory behavior. For this reason, it is most applicable to systems with longer goal sequences, and those in which several goal sequences are achieved in the pursuit of a central task.

This chapter presents two corpora that were collected from middle-school students interacting with two different interactive narrative experiences. For each corpus, the narrative context and gameplay are presented, followed by a discussion of the corpus collection. Lastly, each of the corpora are presented in the context of goal recognition, with a discussion of the actions, locations, narrative state, and goals observed within the environment.

5.1 Crystal Island: Outbreak

CRYSTAL ISLAND: OUTBREAK is a narrative-centered learning environment designed to teach basic microbiology concepts. It follows the North Carolina State mandated curriculum for 8th grade science level. Now in its third major iteration, CRYSTAL ISLAND: OUTBREAK is built using the Valve Source™ Engine, the 3D game platform for Half-Life 2. CRYSTAL ISLAND: OUTBREAK has served as a testbed application for studying Motivation, Off-task Behavior, and Goal Recognition. Studies have shown that interactions with Crystal Island can have a positive effect on learning, motivation and presence.

5.1.1 Story & Gameplay

CRYSTAL ISLAND: OUTBREAK takes place on a remote island, where several members of the research outpost have fallen ill. The player takes on the role of a visitor to the island after hearing her father's fallen ill. When she arrives, she is instructed to meet with Kim, the camp nurse. Kim tells the player that many of the research team has fallen ill, and that she requires the player's aid in discovering the mystery. From there, the player is able to explore a rich 3D environment, and interactive narrative experience is shaped by the player's investigation.

Many of the gameplay mechanics are designed to support the scientific method. A poster in the Infirmary area introduces the player to the scientific method. Following that introduction, the players have access to a Diagnosis Worksheet, which allows them to record information about their hypotheses. Virtual laboratory equipment allows the players to perform experiments on the various objects situated in the environment. The results of these experiments can be recorded in the Diagnosis Worksheet for further inspection.

In addition to the diagnosis worksheet, several other resources are available to the players in game. Books and posters are placed throughout the environment that contain information on various microbiology concepts. Several of the Non-Player Characters (NPCs) are also identified as experts on Bacteria or Viruses. These NPCs will impart knowledge about their subject area through a menu-driven dialogue system. Other NPCs will report symptoms of the illness that the players can use to arrive at the correct solution to the mystery. The players' concept mastery is tested in game through a labeling activity. In order to correctly solve the mystery, the players must identify the components of the contaminant detected by the lab equipment. The information required to identify the

components is all available in the game, through the various posters, books, and NPC dialogue.

5.1.2 Corpus Description

The data for this work was gathered from 153 8th grade players, aged 12-15 ($M=13.3$, $SD=0.48$) in the Wake County public school system. Sixteen players were removed due to incomplete data or prior experience with *CRYSTAL ISLAND: OUTBREAK*. Of the remaining 137 players, 77 were male, 60 were female. Approximately 3% identified as American Indian or Alaskan Native, 2% Asian, 32% African American, 13% Hispanic or Latino, and 50% White. 41.6% of the players were able to complete the mystery in the allotted time. Twenty of the 137 players experienced a game crash, and were instructed to restart the game. Both sets of observations will be used for this work.

The players were first presented with an overview of Crystal Island, which introduced the backstory and task description and the game controls. They were also given handouts, which contained that information, as well as a map of the island and a description of the NPCs in the virtual environment. The players were then given a pre-survey, which assessed their science content knowledge, in addition to various psychological instruments. The players interacted with the system for a maximum of 60 minutes. At the end of 60 minutes, or when they completed the mystery, the players were given a post-survey. The whole interaction lasted about 120 minutes. A full description of the methodology, as well as the instruments contained in the pre-survey and post-survey, can be found in (Rowe et al., 2011).

Feature extraction was performed by processing the log files generated by the game. Each time the player launched the game, a new log file was created. A unique ID was generated for each player, which could be used to associate the gameplay traces with

other instruments taken during the interaction. The goals described in Section 5.1.3 logged within the system, and so they could be extracted automatically.

As mentioned in Section 3.2, each observation was associated with the next goal completed. All actions taken after the last goal completed were discarded, since they could not be correctly attributed to a goal. The last action prior to the goal completion was taken to be the action which achieved the goal, and was removed. Each player's trace log was transformed into the PML input language for TheBeast, before being assigned to one of the 10 folds for cross-validation. See Chapter 6 for details.

5.1.3 Goal Recognition in Crystal Island: Outbreak

Following Ha et al. (2011), there are seven goals that lie along the critical path for solving the mystery. These goals are never directly presented to the player. Instead, the player may discover them through gameplay.

Speak with the camp nurse: The camp nurse, Kim, presents the situation on the island to the player. This establishes the task description.

Speak with the sick patient: One of the sick patients, Theresa, explains her symptoms to the player. These symptoms can then be matched against the in-game information about the candidate illnesses.

Speak with the camp cook: The camp cook, Quentin, provides a list of food and drink consumed by the patients. This allows the player to select objects to test.

Speak with the camp bacteria expert: The camp's bacteria expert, Robert, provides information about the appearance, components, and behavior of bacteria. This information is required for the labeling activity.

Speak with the camp virus expert: The camp’s virus expert, Ford, provides similar information about viruses. This information is also required for the labeling activity.

Test the contaminated object: In order to correctly identify the contaminated objects, tests must be run using the virtual lab equipment. The testing procedure is explained in greater detail below.

Submit the final diagnosis: Once the player has tested the correct object, the final diagnosis may be submitted to the camp nurse. In addition to having the correct entry identified on the diagnosis worksheet, the player must successfully complete a labeling activity which identifies the contaminant as a virus or bacteria. Once this is completed, the camp nurse will congratulate the player, and begin work on a treatment plan.

Table 5.1 shows the proportion of actions attributed to each goal in the corpus. As in the prior goal recognition study, every action is attributed to the next goal accomplished, whether it is in direct service of the goal or not. Since the players are never made directly aware of the goal, they may perform some exploratory actions before completing a goal.

Table 5.1: Goals in CRYSTAL ISLAND: OUTBREAK, with relative frequency

Speak with the camp nurse	6.42%
Speak with a sick patient	11.01%
Speak with the virus expert	11.16%
Submit final diagnosis	17.06%
Speak with the bacteria expert	12.51%
Speak with the camp cook	15.22%
Test the contaminated object	26.62%

There are nineteen distinct types of actions that are recorded by the system. These actions capture eleven different types of interactions with the environment.

Stow and Retrieve Item The player can only carry one item in her times at a time. in addition, interacting with the doors and lab equipment require that the players hands be free. In order to facilitate the testing process, the player is equipped with a backpack, which can hold one additional item. Stowing the object moves the object from the player's hands into the backpack. Retrieving the item allows them to take it out from the backpack for further use.

Pickup and Drop Item: The player is able to pick up and put down certain objects in the 3D environment.

Diagnosis Worksheet: The diagnosis worksheet was explained in section 5.1.1.

Notes: The player is able to take freeform textual notes within the virtual environment. This can be used to record information that cannot be recorded in the diagnosis worksheet.

Move: The player is able to move freely within the 3D virtual environment. In order to support goal recognition, the island is partitioned into thirty-nine distinct locations. The move action is recorded when the player moves from one of these locations to another.

Interact with PDA: The player has a small PDA device, which can be used to take notes (with the Note action), read about microbiology terms, and access in-game quizzes. Three actions capture this interaction: opening up the PDA, accessing the content, and closing the PDA.

Open and Close Doors: Doors separate the indoor regions from the outdoor regions.

Test an Object for contaminants: This action records use of the virtual lab equipment to test one of the objects. Running a test requires that the player enter in a hypothesis for the type of contaminant (bacteria or virus), as well as a hypothesis for the transmission vector (food, drink, or contact). One action is used to record the request for a test, and one is used to record the results.

Talk with a Character: This action records the start of a conversation with a virtual character. One action initiates the conversation, and another is used to guide the topic of conversation.

Labeling Activity: There are two labeling activities the player must complete in order to solve the mystery. This action involves looking at a photograph of bacterial and viral structures, and labeling the various components.

Read Books: There are several virtual books which can be read. They contain information about the various microbiology concepts which are being supported by the learning environment. One action is used to open the book, and another to close it.

A list of the actions and their relative frequencies can be found in Table 5.2.

CRYSTAL ISLAND: OUTBREAK has seven main areas, which are further subdivided into thirty-nine distinct, non-overlapping regions.

Start: The start location is where the character first enters the virtual environment. It is a short distance from the camp.

Infirmary: The infirmary is where the camp nurse, and the sick patients are. It contains

Table 5.2: Actions within Crystal Island, with the frequency of occurrence

Read Book	0.56%	Close Book	0.56%
Pickup Item	6.90%	Drop Item	6.90%
Stow Item	0.95%	Retrieve Item	1.07%
Test Object	1.12%	Read Test Results	1.89%
Labeling Activity	1.64%	Talk to a Character	2.27%
Select Conversation Topic	11.97%	Open PDA	3.27%
Interact with PDA	9.34%	Take Notes	0.85%
Close PDA	3.25%	Diagnosis Worksheet	3.79%
Open Door	7.01%	Close Door	0.44%
Move	36.22%		

the sub-locations *Kim, Theresa, Far Left, Far Right, Empty Bed, Medicine Cabinet, and Bathroom*.

Dining Hall: The dining hall is where the camp cook can be found. In addition, many of the objects that may be tested for contaminants are found in the dining hall. It contains the sub-locations *Front, Middle Table, Back Middle, Fridge, Back Soup table, Left Table, and Right Table*.

Laboratory: The laboratory is where the virtual testing equipment can be found. It contains the sub-locations *Front, Middle Right, Library, Sinks, and Back*.

Bryce’s Quarters: Bryce’s quarters are where the player’s father, Bryce, lives. It contains the sub-locations *Hallway, Bedroom, Office, Guest Room, and Sitting Area*

Living Quarters: The living quarters are where the rest of the staff can be found. It contains the sub-locations *Hallway, Sitting Area, Robert’s Room, and Ford’s Room*

Outdoors: The outdoors location contains 10 sub-locations, arranged in a rough grid over the environment. This is primarily used to capture movement between the

indoor areas.

The narrative of *CRYSTAL ISLAND: OUTBREAK* has four different turning points. These turning points are represented in the goal recognition models using the state predicate, as in Section 3.5. After each one of these points, it is expected that the player changes their behavior.

Talk to the Camp Nurse Talking to the camp nurses starts off the mystery, and sets the player on the path. From here, the next two milestones may be entered in either order.

Testing the Contaminated Object One major aspect of the narrative is the quest for the contaminated object. This involves inspecting various items around the island, and bringing them back to the testing station. Once the testing has returned a positive result, it is assumed that the player will continue on to another task. Recent work has shown, however, that this is not always the case (Rowe, 2013).

Submitting the Diagnosis Worksheet The other aspect of the narrative is collecting clues from the posters, books, and characters around the island. This information, is recorded in the Diagnosis Worksheet, and may be submitted for review.

Game Over Once the correct diagnosis has been rendered, the game is considered completed. This may also happen when the player runs out of the allotted time.

The central question of *CRYSTAL ISLAND: OUTBREAK* is “What is making the research team members sick?”. The question is raised when the players first talk to Kim, the camp nurse. Before that point, any team members they speak to will simply guide them back to the nurse. Once they are presented with the questions, there are ways to gain further information.

Ask the sick team members about their eating habits There are several sick patients the players can ask for more information. Asking the patients about their eating habits allows the players to reduce the number of objects that need to be tested since the illnesses in the game are spread through food and drink.

Ask the sick team members about their symptoms In addition to their eating habits, the patients can also be asked about their symptoms. This information helps the players reduce the number of candidate diseases. Posters scattered throughout the buildings provide symptoms for various diseases. They need to select the correct disease on the diagnosis worksheet to solve the mystery.

Learn about bacteria The eventual solution to the mystery is a bacterial infection. As the players learn about bacteria, they are able to piece together the information to reduce the number of candidate diseases. There are two places where the players learn about bacteria. They can read one of the books on the subject, or they can speak with the camp's bacteria expert.

Learn about the disease In addition to the posters, there are also books found throughout the island which offer more in-depth information about the various candidate diseases. Reading the book about the disease helps the player identify the disease and consider various treatment plans.

Use the diagnosis worksheet The Diagnosis worksheet is used for two purposes within the game. On the one hand, it is required for the player to finish the game, since they must enter the correct information on it to solve the mystery. However, the worksheet also helps the player offload some of the information to the system, and make connections about the information they have gathered so far.

Judicious use of the diagnosis worksheet has been shown to impact the players' eventual learning. (Sabourin et al., 2011)

Experiment with the Testing equipment Testing the contaminated objects is one of the goals for the game. However, negative tests also provide clues towards the transmission source, since it eliminates some possibilities. This event recognizes when the players have tried to test anything, and are therefore familiar with the equipment, and the response it can give.

5.2 Crystal Island: Uncharted Discovery

CRYSTAL ISLAND: UNCHARTED DISCOVERY is a narrative-centered learning environment designed to teach basic landforms and navigation. It follows the North Carolina State mandated curriculum for the 5th grade science level. CRYSTAL ISLAND: UNCHARTED DISCOVERY is built using the Unity game engine. Players exhibited significant learning gains and problem solving skills increase (Lester, Spires, Nietfeld, Minogue, Mott, and Lobene, Lester et al.).

5.2.1 Story & Gameplay

CRYSTAL ISLAND: UNCHARTED DISCOVERY takes place on a fictional island in the Oceania region of the Pacific Ocean. The players take on the role of a child who has been shipwrecked on an island after a tropical storm, along with a crew of explorers. After an introductory cinematic, the player joins a cast of non-player characters (NPCs) on the island in establishing a new life. From there, the players are able to explore a rich 3D environment as they perform tasks for the various island inhabitants.

Gameplay is focused on the exploration of the environment. As the players learn about landforms, navigation and modeling, they are asked to perform several quests to evaluate their skills. Following a brief tutorial segment to familiarize the players with the interface and gameplay, they have access to several in-game resources to help them. The players are provided with a tablet device, which gives them access to the in-game applications that they can use.

IslandPedia The IslandPedia app provides educational content about the various landforms that they may encounter. This is delivered through multimedia presentations with voice-over narration, and the player may view the content at any time.

Problem-Solving The problem-solving app provides a guide through problem solving activities that the player may want to use in order to fulfill the quests.

Camera The Camera app is used for several of the in-game quests. In addition, the players are free to take photos of anything of interest they find throughout the environment.

Photo Journal the Photo journal allows the players to view any of the photos they have taken so far. In addition, it provides some in-game note taking ability, as they can annotate a photo with some text, or create notes independent of photos they have taken.

Text Message The text message app allows key NPCs to contact the players about resources they may want to review or quests that are available.

Map The map app allows the player to view an overhead map of the area. The map is segmented into a grid, which aids in many of the navigation challenges.

5.2.2 Corpus Collection

The corpus was collected from eight participating middle schools. Each fifth-grade classroom interacted with the software over a 4-week period. Approximately eight hundred players participated in the corpus collection. 49% percent of participants were male; 62% Caucasian, 14% African American, 8% Asian and other. The schools represented urban (40%), suburban (20%), and rural settings (40%) (Lester, Spires, Nietfeld, Minogue, Mott, and Lobene, Lester et al.).

The players interacted with the software six times over the 4-week period, supplemented with six teacher-lead lessons. Each interaction was 50 minutes long.

Feature extraction was performed by processing the log data generated by the game. Player sessions were recorded to a MySQL database during the interactions. Each time the player logged into the game, a new session was created. Goals were annotated using predicates defined over the observations, as described in Section 5.2.3.

For this work, only sessions from the first two weeks were considered. During the first two weeks, only four of the quests were available to the players. These quests will be discussed in Section 5.2.3. This was done to limit the scope of the data analyzed for this work. The tutorial data was also removed.

As in the CRYSTAL ISLAND: OUTBREAK corpus, each observation was associated with the next goal completed. All sessions for a given player were merged together into a single sequence of observations. All actions taken after the last goal completed were discarded, since they could not be correctly attributed to a goal. The observation which satisfied the goal predicates were removed, as they were the action that achieved the goal. Each player's trace log was transformed into the PML input language for TheBeast, before being assigned to one of the 10 folds for cross-validation.

5.2.3 Goal Recognition in Crystal Island: Uncharted Discovery

Within the first two weeks of the study, players had access to the first four quests. Two of the quests are about understanding landforms, such as plateaus, dams and waterfalls. Two of the quests involve understanding navigation, both through reading a map and following a heading for a specified amount of distance. Each quest has three goals that were used as the goals for this work. Once a quest was begun, any of its goals could be accomplished in any order. Quests could also be repeated, since faster times were given in-game rewards in the forms of trophies.

Landform Identification The player is given 3 landforms drawn on a blackboard, and is asked to place a sign next to each one.

Landform Photography The player is given a list of three landforms, and is asked to take a photo of each one.

Map Navigation The player is given 3 map cells, and asked to pick up the flag at each location.

Orienteering The player is given directions, using both direction and scale, to 3 locations and are asked to take a photo of the animal located there.

Table 5.3 shows the proportion of actions attributed to each goal in the corpus. As in the prior goal recognition study, every action is attributed to the next goal accomplished, whether it is in direct service of the goal or not. Since the players are never made directly aware of the goal, they may perform some exploratory actions before completing a goal.

There are thirty-seven distinct actions that the player can take. There are nine categories of events that are not associated with any specific quest.

Table 5.3: Goal Statistics for CRYSTAL ISLAND: UNCHARTED DISCOVERY

Photograph Animal at 175E 25N	7.76%
Pick up Dark Blue Flag	11.52%
Photograph Lake	10.22%
Pick up Burgundy Flag	6.80%
Photograph Animal at 75E	5.88%
Photograph Animal at 200W 75S	6.32%
Place Sign at Waterfall	8.83%
Pick up Green Flag	6.06%
Place Sign at Plateau	8.16%
Photograph Delta	12.96%
Place Sign at Volcano	6.76%
Photograph Tributary	8.73%

Enter Buildings There are several buildings in the environment that the player can enter. These house the various NPCs that give quests to the player.

Access Inventory The players have an inventory that can hold three items. They are able to open up the inventory, select an object, drop an object, and close the inventory. The selected object will be available for use during the game.

Access Tablet As described in Section 5.1.1, the player has access to a virtual tablet with several apps. They may open the tablet, switch between apps, use an app, and close the tablet.

Manage Quests The player is able to view the quests they have active at any time. They can also quit a quest once it has begun. Speaking with the NPC who delivered the quest, they are able to start a quest, check a quest, or complete a quest.

Collect sand dollars An in-game currency, called sand dollars, is available for collection. These coins are scattered through the environment and can be found in

treasure chests.

Treasure Chests Some treasure chests have also washed up on the island. These chests contain sand dollars that can be used to purchase in-game assets. The treasure chests are locked by a quiz question.

Trading Post There is an in-game trading post where the player can purchase various trinkets.

Dialogue The player is able to use a menu-based dialogue system to interact with the NPCs.

Pause, Continue and Quit the game Since the corpus collection lasted several days, the player also has the option of quitting the game and returning to it later.

Each of the quests has its own set of actions.

Manage signs One of the quests involves placing signs. Events are fired when the player enters and leaves the designated areas where the signs must be placed.

Manage flags One of the quests involves picking up flags. Events are fired when the player picks up and drops a flag.

Modeling One of the quests not considered for this work involves manipulating a scale model of the encampment, and place model buildings in the correct place.

Tutorial Interactions The tutorial also has its own set of interactions which were not considered for this work.

A list of the actions and their relative frequencies can be found in Table 5.4.

Table 5.4: Actions with Relative Frequencies in CRYSTAL ISLAND: UNCHARTED DISCOVERY

Close Modeling GUI	0.03%	Continue Game	0.04%
Use Problem Solving App	3.79%	Open Tablet	6.00%
Close Treasure Chest	1.32%	Remove Item	2.53%
Open Modeling GUI	0.03%	Collect Sand Dollars	1.98%
Select Equipment	0.76%	Close Tablet	5.82%
Leave Trading Post	0.13%	Complete Quest	0.17%
Answer Quiz Question	0.95%	Place Sign	0.20%
Take Photo	0.72%	View Photo	1.35%
Change Tablet App	9.49%	Pick up Flag	0.24%
Use Modeling App	0.54%	Leave Signpost Area	3.21%
Open Treasure Chest	1.32%	Pick up Sign	0.23%
Quit Quest	0.13%	Ride the Elevator	0.20%
Open Inventory	1.14%	Use IslandPedia	0.35%
Enter Signpost Area	3.85%	Check Quest Progress	0.97%
Select Dialog Topic	10.33%	Begin a Quest	0.56%
Enter a Building	2.11%	View Photos and Notes	10.04%
Close Inventory	1.14%	Move	21.95%
Receive Sand Dollars	2.59%	Add Item to Inventory	2.22%
View Messages	1.19%	Drop Flag	0.35%

The map application presents the player with a grid of the island. The grid segments the area into 60 reachable cells. Each of these cells is recorded as a location observation. A move action was injected into the play trace whenever the player changed between two cells.

In addition to the external cells, there are several internal buildings which the player can access.

Modeling hut The modeling hut contains the scale model of the area.

Cartographers hut The cartographer's hut is where the player finds the NPC who manages the landform quests.

Trading post The trading post allows access to the in-game trinkets that can be purchased.

The narrative of *CRYSTAL ISLAND: UNCHARTED DISCOVERY* is about the exploration of the island. The quests that the players get sent on serve to guide them through the environment. In this way, the quests serve as the major milestones for the narrative. Completing a quest is a turning point in the understanding of the island. Each quest is taken as a milestone for representing the state predicate.

Since the quests guide the player through the narrative, the discovery events for *CRYSTAL ISLAND: UNCHARTED DISCOVERY* are tied to the quests. There are some events that deal with how the player interacts with the quests themselves, following the precedent set by the state predicate. The navigation quests and landform quests have their own set of discovery events that are related to how the quest's question is resolved. Lastly, there are some general discoveries that may aid in any quest.

Starting Quests The first discovery for any of the quests is the quest itself. By asking

the player to help, the NPC starts a narrative related to that quest's accomplishment.

Quitting a Quest When a quest is quit, it presents a negative answer to the question of whether or not the player will complete the quest.

Take notes As with the Diagnosis Worksheet, the in-game notes provide some facility for the player to record information and create inferences.

View the Island Map The map provides information for all four quests used in this work. For the landform quests, the map shows an overhead view of the island. This allows the player to find the required landforms and navigate to them. For the navigation quests, the map is the only route to finding the solution.

Use the IslandPedia For the landform quests, the IslandPedia allows the player to discover what the landform is and offers clues to where it may appear on the island.

Pickup and Drop flags There are red herring flags placed in nearby areas of the island, to lead the player astray. The player can only hold three flags at a time, these red herrings must be eliminated to advance in the quest.

Pick up a placed sign For the landform identification quest, the signs may also be placed in the wrong area. This serves the same red herring purpose as the extra flags.

Take a photo For both the Orienteering and Landform Photography task, photograph's must be taken. Discovering how to operate the camera is a crucial part of achieving these quests.

5.3 Summary

In this chapter, two goal recognition corpora were described. The first, *CRYSTAL ISLAND: OUTBREAK*, was collected from over 150 middle school students. The game presents a medical mystery narrative, where the player is attempting to divine the type and source of an infection. The eleven goals were selected to highlight the major problem solving steps necessary to correctly solve the mystery. The game supports a wide variety of actions, and several locations of interest to explore. The central question of the narrative is “What is making the research team members sick?” Six discovery events were identified through which the player is able to gain clues towards solving the mystery.

The second corpus, *CRYSTAL ISLAND: UNCHARTED DISCOVERY*, was collected from over 800 students across eight middle schools. The game presents a exploration-based narrative, where the student is part of a team shipwrecked on a remote island. The twelve goals were selected from four different quests that are presented to the player as they explore the island. These quests each contain their own question, and challenge the players ability to navigate using a map, and recognize the various landforms on the island. The game also supports a wide variety of actions, and a larger map than its predecessor. Eight discovery events were identified through which the player gains information about the quests and their solutions.

Both corpora were collected under the assumption that each student had access to the same portions of the game, satisfying the assumptions of machine learning. The games direct the student towards solving several goals, which makes them well suited to goal recognition techniques. Both were processed and annotated using the same techniques, which allows them to be modeled through standard machine learning techniques. The evaluations of these models are presented in the next chapter.

Chapter 6

Evaluation

In order to evaluate the thesis of the dissertation, three hypotheses are posited. First, it is hypothesized that the discovery events would enable models that are more accurate than baseline models. By capturing discovery events, the model has access to more knowledge about what has passed in the game. It is hypothesized that this would aid the model in eliminating hypotheses that are not adequately supported by discovery events that have occurred, or hypotheses that have already been accomplished by the player.

Second, it is hypothesized that the discovery events would enable models that converge faster and more often than a baseline. Convergence behavior is governed by the amount of information the model is able to accumulate over the course of the observation sequence. The discovery events allow the model to relax the Markov assumption, and search for key events in the observation sequence. This allows the model to accumulate more information, which may be used for more accurate prediction as the observation sequence approaches the goal. In addition, achieving a discovery event may be indicative of the goal intended, allowing for the model to converge on a goal faster.

Lastly, it is hypothesized that the procedure for discovery events can be used to model

goal recognition in another domain, with similar results. Discovery events are a general representation for modeling progress within a structured problem-solving environment. The corpus selected for the second modeling task is the CRYSTAL ISLAND: UNCHARTED DISCOVERY corpus described in Section 5.2. The evaluation of this hypothesis would show that the procedure is able to generalize to new data, and not capturing specific features of the CRYSTAL ISLAND: OUTBREAK corpus.

6.1 Experimental Design

Each corpus from Chapter 5 was segmented into 10 independent folds for 10-fold cross-validation. Since the observations from a single player are not likely to be independent, all of the observations for a player were placed in the same fold to satisfy the independence assumption. Each of the three metrics described below were calculated over each fold, and a one-way ANOVA test, with a post-hoc Tukey HSD test for multiple comparisons. These analyses were performed using the R statistical package (R Core Team, 2013).

Two naive baselines and a state-of-the-art baseline were constructed for this work. The first naive baseline is the most frequent goal. This goal explained the greatest proportion of observations within the corpora. Since each observation is classified independently, the most frequent baseline is a useful baseline for the accuracy statistics. In CRYSTAL ISLAND: OUTBREAK, the most common goal was *Testing the contaminated object*, which occurred 26% of the time. In CRYSTAL ISLAND: UNCHARTED DISCOVERY, the most common goal was *Taking a photo of the Delta*, which occurred 13% of the time.

The second naive baseline was the most common goal accomplished. Within CRYSTAL ISLAND: UNCHARTED DISCOVERY, a goal could be accomplished multiple times. For both CRYSTAL ISLAND: OUTBREAK and CRYSTAL ISLAND: UNCHARTED DISCOVERY,

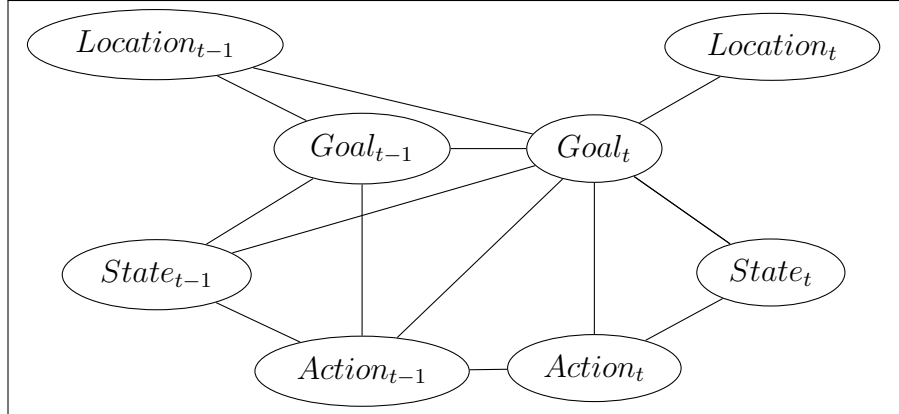


Figure 6.1: Graphical model for the state-of-the-art baseline

a goal which explained more observations, but which was completed by relatively fewer players, would unfairly bias the convergence statistics. Since this goal was completed most often, it would explain the largest percent of converged sequences, making it a useful baseline for the convergence metrics. In *CRYSTAL ISLAND: OUTBREAK*, the most commonly completed goal was *Talking to the Camp Nurse*, while in *CRYSTAL ISLAND: UNCHARTED DISCOVERY* it was *Picking up the Dark Blue flag*.

The state-of-the-art baseline was a previously published model for goal recognition on the *CRYSTAL ISLAND: OUTBREAK* corpus (Ha et al., 2011). This model utilized MLNs, as described in Section 3.4. It was comprised of 13 logical formulae, each with its own weight function which was learned using Markov: The Beast (Riedel, 2008). Figure 6.1 shows the first-order graphical model generated by the baseline formulae. Each node represents a predicate, and edges between nodes represent dependency.

The same formulae form the basis of both the state-of-the-art baseline and the discovery events model for both the *CRYSTAL ISLAND: OUTBREAK* and *CRYSTAL ISLAND: UNCHARTED DISCOVERY* corpora. The full set of formulae are given in Table 6.2. The first formula enforces the restriction that only one goal may be predicted at any time.

$\forall t : \forall g : goal(t, g) = 1$	(6.1)
$\forall t, g : goal(t, g) * w_2(g)$	(6.2)
$\forall t, a, g : action(t, a) \Rightarrow goal(t, g) * w_3(a, g)$	(6.3)
$\forall t, l, g : location(t, l) \Rightarrow goal(t, g) * w_4(l, g)$	(6.4)
$\forall t, s, g : state(t, s) \Rightarrow goal(t, g) * w_5(s, g)$	(6.5)
$\forall t, a, s, g : action(t, a) \wedge state(t, s) \Rightarrow goal(t, g) * w_6(a, s, g)$	(6.6)
$\forall t, a, g : action(t - 1, a) \Rightarrow goal(t, g) * w_7(a, g)$	(6.7)
$\forall t, l, g : location(t - 1, l) \Rightarrow goal(t, g) * w_8(l, g)$	(6.8)
$\forall t, s, g : state(t - 1, s) \Rightarrow goal(t, g) * w_9(s, g)$	(6.9)
$\forall t, a, s, g : action(t - 1, a) \wedge state(t - 1, s) \Rightarrow goal(t, g) * w_{10}(a, s, g)$	(6.10)
$\forall t, a_1, a_2, g : action(t - 1, a_1) \wedge action(t, a_2) \Rightarrow goal(t, g) * w_{11}(a_1, a_2, g)$	(6.11)
$\forall t, a_1, a_2, g_1, g_2 : \left(\begin{array}{l} action(t - 1, a_1) \wedge action(t, a_2) \\ \Rightarrow [goal(t - 1, g_1) \Rightarrow goal(t, g_2)] \end{array} \right) * w_{12}(a_1, a_2, g_1, g_2)$	(6.12)

Figure 6.2: Baseline formulae for MLNs

The second formula captures a prior distribution over the goals. The next four formulae capture the interactions within the current time step, and the four following are the same formulae for the previous time step. The last two capture interactions between the previous and current time steps.

The *discovery events* model extends the previous models in two ways. First, the model was simplified to only consider the current timestep. The first six formulae from Figure 6.2 describe the relationships between the observations and the goal, and form the core of the model for both the CRYSTAL ISLAND: OUTBREAK and CRYSTAL ISLAND: UNCHARTED DISCOVERY corpora. In addition, a new set of formulae were included to capture the discovery events, as described in Chapter 4. The discovery events allow the model to determine which key events the player has already attempted. For each

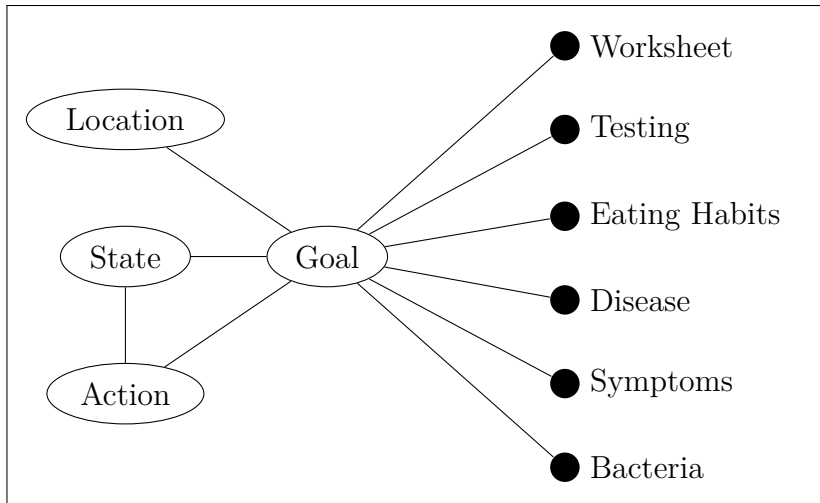


Figure 6.3: Graphical model for the CRYSTAL ISLAND: OUTBREAK discovery events model

discovery event, a milestone formula was created. Each discovery event was expressed in terms of the predicate defined in Section 3.5. For example, the *Testing* discovery event for the CRYSTAL ISLAND: OUTBREAK corpus is defined as $action(j, TESTING)$, while the *Eating Habits* discovery event is defined as $argument(j, EATING_HABBITS)$. These were placed within the milestone formulae, which serve as indicator nodes to determine if the discovery event had happened in the interaction history.

The model for CRYSTAL ISLAND: OUTBREAK is shown in Figure 6.3, and the model for CRYSTAL ISLAND: UNCHARTED DISCOVERY is shown in Figure 6.4. The nodes on the left hand side are generated from the six current step formulae, and the nodes on the right hand side are the indicator nodes generated by the discovery event milestone formulae. In CRYSTAL ISLAND: OUTBREAK, the formulae in Table 6.5 capture the discovery events from Section 5.1.3. In CRYSTAL ISLAND: UNCHARTED DISCOVERY, the formulae in Table 6.6 were used to capture the discovery events from Section 5.2.3.

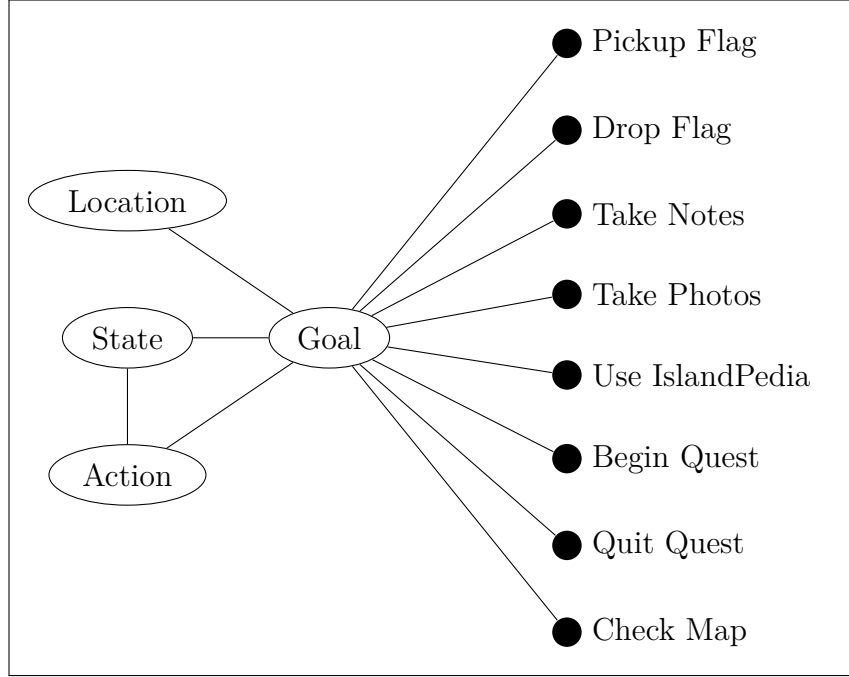


Figure 6.4: Graphical model for the CRYSTAL ISLAND: UNCHARTED DISCOVERY discovery events model

$$\forall t, g : |\forall t_2 < t : action(t_2, \text{"Worksheet"})| \geq 1 \Rightarrow goal(t, g) * w_{13}(g) \quad (6.13)$$

$$\forall t, g : |\forall t_2 < t : action(t_2, \text{"Test"})| \geq 1 \Rightarrow goal(t, g) * w_{14}(g) \quad (6.14)$$

$$\forall t, g : |\forall t_2 < t : argument(t_2, \text{"Eating Habits"})| \geq 1 \Rightarrow goal(t, g) * w_{15}(g) \quad (6.15)$$

$$\forall t, g : \left| \forall t_2 < t : \begin{array}{l} action(t_2, \text{"Read"}) \wedge \\ argument(t_2, \text{"Salmonellosis"}) \end{array} \right| \geq 1 \Rightarrow goal(t, g) * w_{16}(g) \quad (6.16)$$

$$\forall t, g : |\forall t_2 < t : argument(t_2, \text{"Symptoms"})| \geq 1 \Rightarrow goal(t, g) * w_{17}(g) \quad (6.17)$$

$$\forall t, g : |\forall t_2 < t : argument(t_2, \text{"Bacteria"})| \geq 1 \Rightarrow goal(t, g) * w_{18}(g) \quad (6.18)$$

Figure 6.5: Discovery event formulae for CRYSTAL ISLAND: OUTBREAK

$$\forall t, g : |\forall t_2 < t : action(t_2, \text{“Pickup Flag”})| \geq 1 \Rightarrow goal(t, g) * w_{13}(g) \quad (6.19)$$

$$\forall t, g : |\forall t_2 < t : action(t_2, \text{“Drop Flag”})| \geq 1 \Rightarrow goal(t, g) * w_{14}(g) \quad (6.20)$$

$$\forall t, g : |\forall t_2 < t : action(t_2, \text{“Take Notes”})| \geq 1 \Rightarrow goal(t, g) * w_{15}(g) \quad (6.21)$$

$$\forall t, g : |\forall t_2 < t : action(t_2, \text{“Take Photo”})| \geq 1 \Rightarrow goal(t, g) * w_{16}(g) \quad (6.22)$$

$$\forall t, g : |\forall t_2 < t : action(t_2, \text{“Use IslandPedia”})| \geq 1 \Rightarrow goal(t, g) * w_{17}(g) \quad (6.23)$$

$$\forall t, g : |\forall t_2 < t : action(t_2, \text{“Begin Quest”})| \geq 1 \Rightarrow goal(t, g) * w_{18}(g) \quad (6.24)$$

$$\forall t, g : |\forall t_2 < t : action(t_2, \text{“Quit Quest”})| \geq 1 \Rightarrow goal(t, g) * w_{19}(g) \quad (6.25)$$

$$\forall t, g : |\forall t_2 < t : action(t_2, \text{“Check Map”})| \geq 1 \Rightarrow goal(t, g) * w_{20}(g) \quad (6.26)$$

Figure 6.6: Discovery event formulae for CRYSTAL ISLAND: UNCHARTED DISCOVERY

6.2 Hypotheses

Each of the three hypotheses were tested to evaluate their validity according to the experimental setup above.

6.2.1 Hypothesis 1: Accuracy

The first experiment evaluates the accuracy of the approach. It was hypothesized that a discovery-based approach to goal recognition will achieve *greater accuracy* than the baseline models. Accuracy is measured by the F_1 score.

The F_1 score is defined as the harmonic mean between two other metrics, *precision* and *recall*. In a typical binary classification situation, *precision* is defined as $\frac{\#\{\text{True Positives}\}}{\#\{\text{Labeled Positives}\}}$, where a *True Positive* is a positive instance which is labeled as positive. *Recall* is defined as $\frac{\#\{\text{True Positives}\}}{\#\{\text{Actual Positives}\}}$. The goal recognition problem, however, is a multi-class classification problem, since one of several goals must be selected at each time step. There are several ways to extend the notions of precision and recall to the multi-class scenario. For this

work, the evaluation was performed within the Markov Logic Network framework, and therefore uses the same definition that is common within that community. *Precision* is defined as $\frac{\#\{\text{True Groundings Predicted}\}}{\#\{\text{Predicted Groundings}\}}$ and *recall* is defined as $\frac{\#\{\text{True Groundings Predicted}\}}{\#\{\text{All True Groundings}\}}$. In this way, *precision* gives us a measure of how many of the groundings we predicted were true, and *recall* gives us a measure of how many of the true groundings we predicted.

In the Goal Recognition framework, the only grounding being predicted is the student’s current goal. Since the formulae ensure that there is only one goal at any given time step, the number of predicted groundings and the number of true groundings are equal. Therefore, the *precision* and *recall* will be equal as well. For this evaluation, we will only report the F_1 score, which is equivalent to both *precision* and *recall* across all folds.

The results of the first experiment are given in Table 6.1. The ANOVA test revealed significant differences between the accuracy scores of the four models ($F(3,36)=294.4$). The post-hoc Tukey HSD test showed that each of the differences between the models were statistically significant, with the Discovery Events model performing the best.

Table 6.1: Average F_1 scores for each model on the CRYSTAL ISLAND: OUTBREAK dataset. The p-value for the Tukey test is shown for all comparisons.

Model	F_1	p-value			
		Frequent	Converged	Base	Discovery
Baseline: Frequent	0.26611		$\leq 10^{-7}$	$\leq 10^{-7}$	$\leq 10^{-7}$
Baseline: Converged	0.06483	$\leq 10^{-7}$		$\leq 10^{-7}$	$\leq 10^{-7}$
Base	0.4875	$\leq 10^{-7}$	$\leq 10^{-7}$		≤ 0.015
Discovery Events	0.5463	$\leq 10^{-7}$	$\leq 10^{-7}$	≤ 0.015	

6.2.2 Hypothesis 2: Convergence

The second experiment evaluates the convergence behavior. It was hypothesized that a discovery-based approach to goal recognition would converge more often than the baseline models, and that it would converge sooner than the state of the art baseline. Two metrics were used to compare the convergence behavior: convergence rate, and convergence point.

The *convergence rate* is the ratio of sequences that eventually found the correct goal. Any sequence whose final action is predicted as belonging to the correct goal is said to have converged on the goal. The most frequently completed baseline model was designed to produce a reasonable naive baseline for this score.

The *convergence point* measures how early in the sequence the system was able to converge on the correct goal. The convergence point is the earliest point after which only the correct goal is predicted. For sequences which did not predict the correct goal in the end, this measure is undefined. The convergence point metric is reported as the percent of the plan that had to be observed on average before the correct goal was consistently predicted.

Table 6.2 shows the results of the convergence rate analysis. The ANOVA test showed significant differences for the convergence rate between models ($F(3,36) = 334.5$). The post-hoc Tukey HSD test showed that all of the pairwise comparisons were statistically significant as well, with the discovery events model converging on about half of the sequences.

Table 6.3 shows the results of the convergence point analysis. The ANOVA test showed significant differences for the convergence point between models ($F(3,36) = 250.5$). The post-hoc Tukey HSD test showed that all of the pairwise comparisons were statistically significant as well. For this measure, the lower numbers indicate better per-

Table 6.2: Average convergence rate for each model on the CRYSTAL ISLAND: OUT-BREAK dataset. The p-value for the Tukey test is shown for all comparisons.

Model	Conv. Rate	p-value			
		Frequent	Converged	Base	Discovery
Baseline: Frequent	13.29806		$\leq 10^{-7}$	$\leq 10^{-7}$	$\leq 10^{-7}$
Baseline: Converged	18.19188	$\leq 10^{-7}$		$\leq 10^{-7}$	$\leq 10^{-7}$
Base	30.90598	$\leq 10^{-7}$	$\leq 10^{-7}$		≤ 0.0025
Discovery Events	50.05555	$\leq 10^{-7}$	$\leq 10^{-7}$	≤ 0.0025	

formance. Since the two naive baselines are constant predictors, they converge after a single observation, and so represent a gold standard for the convergence point. The discovery events model, however, did perform significantly better than the state-of-the-art baseline.

Table 6.3: Average convergence point for each model on the CRYSTAL ISLAND: OUT-BREAK dataset. The p-value for the Tukey test is shown for all comparisons.

Model	Conv. Point	p-value			
		Frequent	Converged	Base	Discovery
Baseline: Frequent	1.116290		≤ 0.0025	$\leq 10^{-7}$	$\leq 10^{-7}$
Baseline: Converged	9.12618	≤ 0.0025		$\leq 10^{-7}$	$\leq 10^{-7}$
Base	50.865067	$\leq 10^{-7}$	$\leq 10^{-7}$		$\leq 10^{-6}$
Discovery Events	35.861892	$\leq 10^{-7}$	$\leq 10^{-7}$	$\leq 10^{-6}$	

6.2.3 Hypothesis 3: Transference

The last experiment evaluates the generalizability of the discovery-based approach. We hypothesized that the procedure for discovery events could be used to model goal recogni-

tion in another domain, with similar results. We performed the previous two experiments on a second corpus, The Crystal Island: Uncharted Discovery corpus described in Section 5.2. The naive baselines were selected from the new corpus. The same formulae for the state-of-the-art baseline were used, but the model was trained on the new corpus. For the discovery events model, new formulae were developed to capture the narrative structure.

The same three metrics were computed on the new models. Table 6.4 shows F_1 scores for each of the models. While the average F_1 scores were lower than in the CRYSTAL ISLAND: OUTBREAK corpus, there were significant differences between the models ($F(3,36) = 285.7$). The state-of-the-art models outperformed the baselines significantly. The baselines were also much closer together than on the CRYSTAL ISLAND: OUTBREAK corpus.

Table 6.5 shows the convergence rate for each of the models. There were significant differences ($F(3,36) = 345.1$), but some of the individual differences were not significant by the Tukey HSD test. The state-of-the-art baseline performed almost identically to the most frequently completed naive baseline. The discovery events model, however, outperformed every other model at a statistically significant level.

Lastly, Table 6.6 shows the convergence point for each of the models. There were significant differences ($F(3,36) = 6000$) between models. The naive baselines performed comparably. As with the CRYSTAL ISLAND: OUTBREAK corpus, the naive baselines are constant and so converge after a single observation. The discovery events model outperformed every other model at a statistically significant level.

Table 6.4: Average F_1 scores for each model on the CRYSTAL ISLAND: UNCHARTED DISCOVERY dataset. The p-value for the Tukey test is shown for all comparisons.

Model	F_1	p-value			
		Frequent	Converged	Base	Discovery
Baseline: Frequent	0.1292285		≤ 0.1	$\leq 10^{-7}$	$\leq 10^{-7}$
Baseline: Converged	0.1155746	≤ 0.1		$\leq 10^{-7}$	$\leq 10^{-7}$
Base	0.226	$\leq 10^{-7}$	$\leq 10^{-7}$		≤ 0.02
Discovery Events	0.244	$\leq 10^{-7}$	$\leq 10^{-7}$	≤ 0.02	

Table 6.5: Average convergence rate for each model on the CRYSTAL ISLAND: UNCHARTED DISCOVERY dataset. The p-value for the Tukey test is shown for all comparisons.

Model	Conv. Rate	p-value			
		Frequent	Converged	Base	Discovery
Baseline: Frequent	10.72035		≤ 0.01	≤ 0.25	$\leq 10^{-7}$
Baseline: Converged	12.96237	≤ 0.01		≤ 0.5	$\leq 10^{-7}$
Base	11.91493	≤ 0.025	≤ 0.5		$\leq 10^{-7}$
Discovery Events	29.97324	$\leq 10^{-7}$	$\leq 10^{-7}$	$\leq 10^{-7}$	

6.3 Summary

In this chapter, we presented an evaluation of the discovery events approach to goal recognition. The thesis of this dissertation states that

Narrative representation can be leveraged in a principled methodology in order to produce goal recognition systems which are more accurate and converge faster than state-of-the-art models.

Three hypotheses were generated in order to support this claim. First, we hypothesized that discovery events would enable models of goal recognition that were more

Table 6.6: Average convergence point for each model on the CRYSTAL ISLAND: UNCHARTED DISCOVERY dataset. The p-value for the Tukey test is shown for all comparisons.

Model	Conv. Point	p-value			
		Frequent	Converged	Base	Discovery
Baseline: Frequent	3.002247		≤ 1	$\leq 10^{-7}$	$\leq 10^{-7}$
Baseline: Converged	3.629238	≤ 1		$\leq 10^{-7}$	$\leq 10^{-7}$
Base	87.785529	$\leq 10^{-7}$	$\leq 10^{-7}$		$\leq 10^{-7}$
Discovery Events	79.349973	$\leq 10^{-7}$	$\leq 10^{-7}$	$\leq 10^{-7}$	

accurate than a baseline approach, since the approach allows the model to maintain more knowledge about what has passed in the game environment. The discovery events model was evaluated against three baseline approaches, according to the F_1 measure for accuracy. The state-of-the-art model was drawn from prior work in goal recognition on this corpus. The two naive baselines were the most frequently labeled goal, and the most frequently converged goal. Statistical testing across the 10-fold cross-validation experiment showed a statistically significant improvement over the baselines.

Secondly, we hypothesized that discovery events would enable models of goal recognition that converge faster and more often than a baseline model, since the additional context allows for a relaxation of the Markov assumption. Two metrics were used: the convergence rate metric measures how often the model converges on the correct goal, and the convergence point measures the portion of the sequence that had to be observed before convergence. The discovery events model was compared against the same three baseline approaches. Statistical testing across the 10-fold cross-validation experiment showed a statistically significant improvement over the baseline on both metrics.

Lastly, we hypothesized that the procedure for discovery events could be used to model goal recognition in another domain with similar results. For this evaluation, the

CRYSTAL ISLAND: UNCHARTED DISCOVERY corpus was used. The naive baselines were recomputed using the new corpus and the state-of-the-art baseline and the discovery events model were both retrained on the second corpus after adding new formulae for the discovery events. The same three metrics of F_1 score, convergence rate, and convergence point, were used for evaluation. Statistical testing across the 10-fold cross-validation experiment showed a statistically significant improvement over the baseline on all three metrics.

Chapter 7

Discussion

The results of the experiments show some interesting trends. On the accuracy metric, the discovery events model statistically outperformed all competitors. It achieved a 12% increase over the state-of-the-art model, and more than 100% increase over the naive baselines. In the second domain, there are similar trends, with the discovery events model achieving 10% increase over the state-of-the-art, and a 92% increase over the naive baseline.

On the convergence metrics, the discovery events model outperformed the state-of-the-art baseline. For the convergence rate, it achieved a 61% increase over the state-of-the-art baseline, and again more than a 100% increase over the naive baseline. For the convergence point, the naive baseline models are not informative since they maintain a consistent prediction. However, the discovery events model did achieve a 41% increase over the state-of-the-art baseline. As before, these results transfer to the second domain. One interesting thing to note is that the state-of-the-art baseline was not statistically different from the naive baseline on the convergence rate metric. As before, the model achieved more than a 100% increase over all baselines on the convergence rate metric,

and an 11% increase over the state-of-the-art baseline on the convergence point metric.

7.1 Ablated Formulae

One concern with the models is that the discovery events model both removed formulae from the state-of-the-art baseline, and added formulae to capture discovery events. Because of this, it is necessary to analyze the interactions between the two components. A secondary experiment was run to determine if one of them alone could account for the benefits.

Four models were considered. The first is the state-of-the-art baseline, as presented in Section 6.1. Secondly, the full baseline with the discovery events formulae were added. An ablated model captured only the current time step predicates, without any of the additional formulae. Lastly, an ablated model with the discovery events formulae added. The ablated formulae are the first six formulae from Figure 6.2, and the milestone formulae from Figure 6.5 were used to capture discovery events. For this experiment, the CRYSTAL ISLAND: OUTBREAK corpus was used for the evaluation. The same three metrics were computed over each of the 10 folds for cross-validation.

The results of the experiment are shown in Tables 7.1, 7.2 and 7.3. Table 7.1 shows the results of the accuracy experiment. There were statistical differences between the models ($F(3,36) = 6.007$). The two models without the discovery events performed similarly to each other, as did the two models with the discovery events. The ablated model with the discovery events, however, outperformed both the models without it at a significant level.

Table 7.2 shows the results of the convergence rate analysis. There were statistical differences for convergence rate by model ($F(3,36) = 60$). The ablated model with

Table 7.1: Average F1 scores for each model on the CRYSTAL ISLAND: OUTBREAK dataset. The p-value for the Tukey test is shown for all comparisons.

Model	F_1	p-value			
		Baseline	Ablated	Baseline + DE	Ablated + DE
Baseline	0.4875		≤ 1	≤ 0.1	≤ 0.05
Ablated	0.4770	≤ 1		≤ 0.025	≤ 0.01
Baseline + DE	0.5374	≤ 0.1	≤ 0.025		≤ 1
Ablated + DE	0.5463	≤ 0.05	≤ 0.01	≤ 1	

discovery events outperformed every other model. The ablated model significantly outperformed baseline, but the addition of the discovery events added another 35% increase in performance.

Table 7.2: Average convergence rate scores for each model on the CRYSTAL ISLAND: OUTBREAK dataset. The p-value for the Tukey test is shown for all comparisons.

Model	Conv. Rate	p-value			
		Baseline	Ablated	Baseline + DE	Ablated + DE
Baseline	30.90598		≤ 0.0025	≤ 0.6	$\leq 10^{-6}$
Ablated	37.00105	≤ 0.0025		≤ 0.1	$\leq 10^{-6}$
Baseline + DE	32.90513	≤ 0.6	≤ 0.1		$\leq 10^{-6}$
Ablated + DE	050.05555	$\leq 10^{-6}$	$\leq 10^{-6}$	$\leq 10^{-6}$	

Table 7.3 shows the results of the convergence point analysis. There were statistical differences for convergence point by model ($F(3,36) = 20.33$). The ablated models both statistically outperformed the baseline models. The ablated models performed similarly to each other, as did the baseline models.

As the results show, the removal of formulae is largely responsible for the convergence

Table 7.3: Average convergence point scores for each model on the CRYSTAL ISLAND: OUTBREAK dataset. The p-value for the Tukey test is shown for all comparisons.

Model	Conv. Point	p-value			
		Baseline	Ablated	Baseline + DE	Ablated + DE
Baseline	50.86507		$\leq 10^{-4}$	≤ 1	$\leq 10^{-5}$
Ablated	37.62764	$\leq 10^{-4}$		$\leq 0.10^{-4}$	≤ 1
Baseline + DE	51.29825	≤ 1	$\leq 0.10^{-4}$		$\leq 10^{-5}$
Ablated + DE	35.86189	$\leq 10^{-5}$	≤ 1	$\leq 10^{-5}$	

point differences. However, both the accuracy and convergence rate gained significant benefit from the inclusion of the discovery events. One possible explanation for the convergence behavior is that the baseline models are much more complex, with many more weights and nodes to estimate during training. This may inject noise into the data, or be prone to overfitting. While 10-fold cross-validation controls for some of the overfitting, the simple models did perform better. There is also a practical benefit to the ablated models: they were also training and evaluating models much more quickly than the baseline models.

7.2 Representations of Narrative State

A second concern is the two representations of the narrative state. Both the binary vector representation and the milestone representation were used to capture different parts of the narrative. Another secondary experiment was run to analyze the effect of the two representations. Two evaluations were run. The first experiment used the CRYSTAL ISLAND: OUTBREAK corpus and represents all the narrative state as a bit-vector. The formulae used are the same as the state-of-the-art baseline on two corpora. The other

$$\forall t, g : |\forall t_2 < t : action(t_2, \text{“Complete Tutorial”})| \geq 1 \Rightarrow goal(t, g) * w_{13}(g) \quad (7.1)$$

$$\forall t, g : |\forall t_2 < t : action(t_2, \text{“Complete 1st Landform Quest”})| \geq 1 \Rightarrow goal(t, g) * w_{14}(g) \quad (7.2)$$

$$\forall t, g : |\forall t_2 < t : action(t_2, \text{“Complete 2nd Landform Quest”})| \geq 1 \Rightarrow goal(t, g) * w_{15}(g) \quad (7.3)$$

$$\forall t, g : |\forall t_2 < t : action(t_2, \text{“Complete 1st Navigation Quest”})| \geq 1 \Rightarrow goal(t, g) * w_{16}(g) \quad (7.4)$$

$$\forall t, g : |\forall t_2 < t : action(t_2, \text{“Complete 2nd Navigation Quest”})| \geq 1 \Rightarrow goal(t, g) * w_{17}(g) \quad (7.5)$$

Figure 7.1: Milestone Formulae for the Narrative State in CRYSTAL ISLAND: UNCHARTED DISCOVERY

extended the representation of the narrative state to include the discovery events before converting to a number.

The second experiment used the CRYSTAL ISLAND: UNCHARTED DISCOVERY corpus to represent the whole narrative as the milestone formulae. In this experiment, all of the formulae that referenced the state were removed in favor of a set of milestone formulae describing the state. The baseline set has only the milestones which describe the quest completion, as described in Section 5.2.3. These formulae are shown in Figure 7.1. The second set has the full discovery events described in Section 5.2.3. In the CRYSTAL ISLAND: OUTBREAK corpus, the goals overlap too closely with the narrative states used in the previous work. Because of this, and the fact that the goal achieving actions were removed, the milestone predicates could not be generated for those events, so the second corpus was used instead.

Table 7.4 shows the results of a one-way ANOVA test on the CRYSTAL ISLAND: OUT-

BREAK corpus. The results are statistically significant for accuracy ($F(1,18) = 18.89$), but not for convergence rate ($F(1,18) = 0.247$) or convergence point ($F(1,18) = 3.317$). Table 7.5 shows the results of the one-way ANOVA on the CRYSTAL ISLAND: UNCHARTED DISCOVERY corpus. Here the differences are statistically significant on all three metrics (F_1 : $F(1,18) = 21.02$, Conv. Rate: $F(1,18) = 469$, Conv. Point: $F(1,18) = 59.18$).

Table 7.4: Results of the representation experiment on the CRYSTAL ISLAND: OUT-BREAK dataset.

Model	F_1		Conv. Rate		Conv. Point	
	Avg.	p-value	Avg.	p-value	Avg.	p-value
Baseline	0.4875	≤ 0.001	30.90598	≤ 1	50.86507	≤ 0.1
Bit Vector	0.5848		31.87227		58.40224	

Table 7.5: Results of the representation experiment on the CRYSTAL ISLAND: UNCHARTED DISCOVERY dataset.

Model	F_1		Conv. Rate		Conv. Point	
	Avg.	p-value	Avg.	p-value	Avg.	p-value
Baseline	0.2265	≤ 0.001	12.86019	$\leq 10^{-13}$	87.86622	$\leq 10^{-6}$
Milestone	0.2491		30.53951		78.98265	

The choice of the two representations does not seem to have an effect on accuracy. The presence of the extra discovery events is the greatest factor. However, the representation has much more of an impact on the convergence behavior. One possible explanation stems from the way inference works in MLNs. MLNs place an edge in the grounded network between predicates which are shared between formulae. It can then estimate the weight of that edge during learning. In the milestone representation, each milestone

$$\forall t : |\forall g : goal(t, g)| = 1 \tag{7.6}$$

$$\forall t, g : goal(t, g) * w_2(g) \tag{7.7}$$

$$\forall t, a, g : action(t, a) \Rightarrow goal(t, g) * w_3(a, g) \tag{7.8}$$

$$\forall t, l, g : location(t, l) \Rightarrow goal(t, g) * w_4(l, g) \tag{7.9}$$

$$\forall t, s, g : state(t, s) \Rightarrow goal(t, g) * w_5(s, g) \tag{7.10}$$

$$\forall t, a, s, g : action(t, a) \wedge state(t, s) \Rightarrow goal(t, g) * w_6(a, s, g) \tag{7.11}$$

$$\forall t, s, g : schema(t, s) \Rightarrow goal(t, g) * w_7(s, g) \tag{7.12}$$

Figure 7.2: Schema formulae for MLNs

has its own connection between the action predicate and the goal predicate, where as the other representation connects with a new predicate for the narrative state.

7.3 Narrative Schema

There are many other ways of capturing discovery events. One alternative that was explored during the course of this work is to directly add the discovery events directly as an observation. The discovery events were added as a new predicate, schema, inspired by the Narrative Schema’s representation (Chambers and Jurafsky, 2009). This predicate captured the last discovery event the player encountered. For this experiment, the state-of-the-art baseline was compared against the ablated model from 7.1 with the addition of a single predicate antecedent formula to capture the new predicate. Figure 7.2 show the formulae for this model. The ablated model was chosen for its simplicity.

The corpus used for this experiment was the CRYSTAL ISLAND: OUTBREAK corpus, with some additional data cleaning. Some observations are redundant, as an artifact of the logging framework used when developing the game. For example, running a lab-

oratory test produces two observations, one for the action of performing the test, and one for the observance of the results. This information was not removed in the original evaluation because the extra information was used to represent which objects were used in the discovery events.

The results of the experiment are shown in Table 7.6. No significant difference was found for accuracy ($F(1,18) = 0.458$), but convergence rate ($F(1,18) = 49.05$) and convergence point ($F(1,18) = 16.06$) were both statistically significant. While the schema model performed better on the convergence rate metric, it performed significantly worse on the convergence point metric. As discussed in Section 7.1, the increased convergence rate is partially explained by the ablated formulae.

Table 7.6: Results of the schema experiment on the CRYSTAL ISLAND: OUTBREAK dataset.

Model	F_1		Conv. Rate		Conv. Point	
	Avg.	p-value	Avg.	p-value	Avg.	p-value
Baseline	0.5113		32.59627	$\leq 10^{-5}$	47.95511	
Schema	0.4897	≤ 1	49.5779		59.07027	≤ 0.001

One possible explanation for these results is that the schema are an incomplete context. The representations in Section 7.2 fully characterize the discovery events encountered by the player. The representation mentioned here, however, only looks at the most recent one. This leads to the hypothesis that extra context is needed to derive the benefits. It is also worth noting that the averages above are different from other discussions of the baseline model. This is due to the fact that extra predicates were extracted for this analysis. MLNs may suffer from the interactions of extra predicates, since the orphaned predicates will be considered cliques by themselves, and factor into the potential

functions.

7.4 Limitations

There are several limitations to the work. First, there are some restrictions placed on the types of interactive environments in which this approach can be used. The environment must be goal oriented. In order to tie the sense of progression to the goals being recognized, the game has to provide a task to the player that they are expected to be pursuing. In *CRYSTAL ISLAND: OUTBREAK*, the high level task is presented by the Kim, the Camp Nurse, while in *CRYSTAL ISLAND: UNCHARTED DISCOVERY* several tasks are presented through the quest system. This presents the central question to the player, and allows for milestones towards that answer to be defined.

A second limitation is the assumption that there are no concurrent goals. Especially because of the exploratory nature of the game, this is unlikely to be true within the environment. Within *CRYSTAL ISLAND: OUTBREAK*, the goals are not explicitly presented, so it is possible for players to make progress towards multiple goals at once. In *CRYSTAL ISLAND: UNCHARTED DISCOVERY*, players may start multiple quests at once, and may plan to satisfy goals from various quests while within the same area. This would lead to a large amount of overlap in the actions which lead towards each goal. Relaxing this assumption would require a large annotation effort, to determine how various actions contribute to the goals being recognized. This is a promising direction for future work.

In large games focused on exploration, it may be difficult to create a sense of progression. This sense of progression is needed to define the discovery events in the way presented here. There may be other options for capturing similar milestones in the environment. If the player is exploring a large map, the player's movement through the

virtual space may provide more context by considering which areas had been visited before. Both of the environments used had more of a sense of narrative by design, which made it easier to define the discovery events.

Lastly, there is the question of how granular the discovery events should be. In the extreme, the goals themselves define stepping stones along the path of the narrative. It may be the case that simply knowing which goals have been completed would provide enough context for goal recognition to perform at a satisfactory level. At the other extreme, each action is giving the players some feedback to the environment and their place within it. The discovery events presented in this work were chosen for the information they provide to both the player and the system. However, further exploration is needed to characterize how the granularity of the events affects the results.

Chapter 8

Conclusions and Future Work

As computers become pervasive, applications have been developed that allow a user to solve problems and explore a creative space. In tandem with these changes, techniques have been developed to support users' interactions. One of the key challenges in designing these user supportive systems is *goal recognition*. Goal recognition attempts to model a user's intent based on the actions they take within an interactive system.

In structured problem-solving environments, goal recognition models can be used to support students in an intelligent tutoring system, or reduce communication in a problem-solving system or a dialogue system, or dynamically adapt a game to encourage players strengths and interests, or exploit their weaknesses in an adversarial system. While systems have been developed to capture changes in the environment due to the user's actions, relatively few have considered the flow of information between the user and the environment.

Modeling the flow of information is of particular interest in large open-ended environments, such as interactive narrative experiences. In these environments, users may perform exploratory actions as they attempt to uncover the goals for themselves. These

exploratory actions may add too much noise for these approaches to handle elegantly. Instead, the approach presented in this dissertation models the user’s progress towards the task. Using Markov Logic Networks as a modeling framework, we developed a goal recognition model that uses the narrative context of the environment to measure progress.

The model was learned in a supervised learning method using data from two gameplay corpora. The model uses observations about the players interactions within the environment, such as the actions they are taking and the location they are in, along with a novel measure of progress to predict the next problem-solving goal the player will achieve. Discovery events, the measure of progress, capture partial answers to the task’s central question. In doing so, they create a series of milestones which can be represented within MLNs to provide the model access to more history than a standard Markov assumption would allow.

8.1 Hypotheses Revisited

This dissertation investigated the thesis that *Narrative representation can be leveraged in a principled methodology in order to produce goal recognition systems which are more accurate and converge faster than state of the art models*. Three hypotheses were investigated to support this thesis statement.

Hypothesis 1 A discovery-based approach to goal recognition will achieve greater accuracy than the baseline models.

- Results on the CRYSTAL ISLAND: OUTBREAK corpus showed that a model augmented with discovery events achieved a statistically significant improvement over three baselines: A state of the art implementation for this corpus,

the most frequently labeled goal, and the most frequently converged goal. Further experiments analyzed the contribution of the formulae removal and narrative representations independently.

Hypothesis 2 A discovery-based approach to goal recognition would converge more often than the baseline models and that it would converge sooner than the state of the art baseline.

- Two metrics were calculated for convergence: convergence rate and convergence point. Results on the CRYSTAL ISLAND: OUTBREAK corpus showed that a model augmented with discovery events achieved a statistically significant improvement over three baselines: A state of the art implementation for this corpus, the most frequently labeled goal, and the most frequently converged goal. Further analysis revealed that the addition of the discovery events worked in concert with the removal of some formulae to increase the rate of convergence. However, the removal of formulae was crucial to the increase in early prediction.

Hypothesis 3 A discovery-based approach to goal recognition would outperform a baseline model when trained and evaluated on a second interactive narrative corpus.

- All three metrics, F_1 score, convergence rate, and convergence point, were measured on the CRYSTAL ISLAND: UNCHARTED DISCOVERY corpus. The results showed that a model augmented with discovery events achieved a statistically significant improvement over three baselines: The same state of the art model from the prior hypothesis trained on the new corpus, the most frequently labeled goal, and the most frequently converged goal.

8.2 Summary

This dissertation investigated models of goal recognition for interactive narrative environments. Goal recognition is the problem of observing a users interaction with a virtual environment, and predicting which goal they are most likely attempting to achieve. In narrative-centered learning environments, these goals are often problem-solving milestones that are key to solving an overarching problem.

For this work, goal recognition is modeled as a classification problem. Using Markov Logic Networks as a classifier, a model was built using three classes of observations: The actions taken within the environment, the location in the environment in which the action takes place, and the narrative context in which the action was performed. The narrative context was measured in two parts: a predicate that indicated the broad narrative structure, in line with previous work on the corpus, and milestone features which were added to recognize key events within the narrative of the environment for a more fine-grained representation. These discovery events model a sense of progression within the game environment by modeling partial answers to the task’s central question. By representing the narrative as a sequence of these milestones, the goal recognition model is able to capture the progress the player has made towards the goal.

The model was compared against a baseline on two corpora along with two other naive baselines. Both corpora were drawn from gameplay traces of middle school students with two different interactive environments. Both environments offer a rich environment with several actions and locations for the players to explore. The models were trained using 10-fold cross validation. The results showed that the discovery events models outperformed the baseline models on three metrics: accuracy, convergence rate, and convergence point.

8.3 Future Work

There are several promising directions for future work. One of the first steps is to investigate further the contributions of discovery event formulae. While the additional features proved to be useful, it may be that some of them were more useful than others or that there were other milestones that were missed in the modeling process. Feature selection procedures could offer some insight into the details of this relationship. The ablated experiment showed that removal of formulae can have positive effects, so it would be promising to attempt feature selection on the models.

Another option is to integrate the goal recognition module into a running interactive narrative. This would offer some insight to how players react to the goal recognition. In particular, it is unknown whether or not the increase in accuracy affords a more enjoyable experience to the user. Implementing goal recognition into a playable experience would allow some intuition as to how accurate goal recognition models need to be in order to benefit interactive systems.

One of the simplifying assumptions was that the goals achieved are the goals the user intended. This may not necessarily be true. It may be possible to use online approaches, and elicit feedback from the player as they are interacting with the system in a semi-supervised manner. There is some risk of bias, since the player may be made aware of previously unknown goals. This could enable models of goal discovery or point developers towards other goals that were not considered during the additional design.

To further explore the contributions of this dissertation, alternate views of both goals and progress could be investigated. In this work, I chose a narrative view of progress because it aligned with the task. There are many other views of narrative. Character and player affect have historically been identified by the narratological community, but tension

and conflict can both play a role. Within learning environments, the player's knowledge is another key component. It would be promising to explore how these interact with each other in the context of narrative-centered learning environments. There may be other dimensions of progress that are relevant to a narrative-centered learning environment as well.

In this work, the goals were chosen to closely relate to the task. However, the goals of the player may not always align with the goals of the creator. Off-task goals occur when the player begins an investigation into some portion of the environment, such as the rooftops, that the developer did not initially intend. These goals may be just as important to model as task-oriented goals in order to keep players engaged in the game content. It may be useful to recognize and model these goals independently of the task-oriented goals. In addition, the granularity of goals should be explored further. The goals for *CRYSTAL ISLAND: OUTBREAK* were chosen to align with previous work on the corpus, but the milestones recognized may even make better goals. For *CRYSTAL ISLAND: UNCHARTED DISCOVERY*, goals were chosen that were both part of the problem-solving task, and would provide sufficient number of goals for analysis. However, the quests and trophies may be more salient goals to the user.

It may also be fruitful to explore the relationships between the users and the goals. Different problem-solving approaches to the game's task may have an impact on learning outcomes. It would be interesting to see if there is a relationship between the goal recognition confidence and the learning outcomes as well. Alternatively, perhaps individual differences, such as player demographics, prior content knowledge, or prior gameplay experience may offer some insights into the goal-recognition process.

Both of the corpora chosen for this dissertation came from narrative-centered learning environments. However, progress could be measured in other structured problem-solving

environments. Many intelligent tutoring systems could benefit from the ability to recognize the students' goals as they work through a problem. `CRYSTAL ISLAND: UNCHARTED DISCOVERY` was less structured, narratively, than `CRYSTAL ISLAND: OUTBREAK`. It seems promising that the approach could work on an even less narratively structured environment, as long as the notion of progress within the task is well defined.

Another simplifying assumption for this work is that the players pursue one goal at a time. This is most likely not true within the environments. Within `CRYSTAL ISLAND: OUTBREAK`, goals were never presented to the player, and players may have partially satisfied many goals. Within `CRYSTAL ISLAND: UNCHARTED DISCOVERY`, players could pursue multiple quests at once. In some cases, objects needed for two different quests were near each other, and so many of the actions leading up to the first object may just as easily have been attributed to the other goal. One approach would be to annotate each action with any goal it would have been in service of. Unsupervised or semi-supervised approaches may reveal interactions between actions and goals that were previously unclear. One option that offers some promise is the schema-based approach. One of the benefits of narrative schema was that they could be learned in an unsupervised method by estimating the probability of co-occurrence. By analyzing actions which occur together within a goal label, interaction probabilities may be able to be estimated.

Finally, there are several options for generalizing goal recognition. Plan recognition is a more general problem that seeks to uncover not only the goal, but how the player is intending to pursue it. This may be useful for many of the same applications, especially when the actions that the player might take may impact the choice of intervention. Another option is a relatively new technique called Inverse Reinforcement Learning (Ng and Russell, 2000). Inverse Reinforcement Learning extracts a reward function given observations of an agent within an environment. Similar to goal recognition, an inverse

reinforcement learning model could observe multiple players interacting with a virtual environment, and discover functions over the domain which the player is seeking to maximize. This could offer insight into a model of goal recognition which could automatically learn the goals, as well as the model which recognizes them.

8.4 Concluding Remarks

Goal recognition has a long history of study within artificial intelligence. Within structured problem solving environments, such as narrative-centered learning environments, goal recognition models can enable systems to be more personalized to the individual users and their behaviors. While there have been many approaches to goal recognition, few have been developed which capture the exploratory nature of these open-ended environments. The work presented in this dissertation represents a step towards goal recognition models that can represent these exploratory actions, and enable systems that aid users in solving increasingly complex goals within virtual environments.

Bibliography

- Albrecht, D. W., I. Zukerman, A. E. Nicholson, and A. Bud (1997). Towards a Bayesian Model for Keyhole Plan Recognition in Large Domains. In A. Jameson, C. Paris, and C. Tasso (Eds.), *Proceedings of the Sixth International Conference on User Modeling*, Vienna, New York, pp. 365–376. Springer Wien New York.
- Allen, J. F. (1981). An Interval-Based Representation of Temporal Knowledge. In P. J. Hayes (Ed.), *Proceedings of the Seventh International Joint Conference on Artificial Intelligence*, Menlo Park, CA, pp. 221–226. AAAI Press.
- Anh, H. T. and L. M. Pereira (2011). Corpus-Based Incremental Intention Recognition via Bayesian Network Model Construction. In D. Pattison, D. Long, and C. Geib (Eds.), *Proceedings of the First Workshop on Goal, Activity and Plan Recognition*, Menlo Park, CA, pp. 1–8. AAAI Press.
- Appling, D. S. and M. O. Riedl (2009). Representations for Learning to Summarize Plots. In S. Louchart, M. Mehta, and D. L. Roberts (Eds.), *Proceedings of the AAAI Spring Symposium on Intelligent Narrative Technologies II*, Menlo Park, CA, pp. 1–4. AAAI Press.
- Armentano, M. G. and A. Amandi (2009). Goal Recognition with Variable-Order Markov Models. In C. Boutilier (Ed.), *Proceedings of the Twenty-First International Joint Conference on Artificial Intelligence*, Menlo Park, CA, pp. 1635–1640. AAAI Press.
- Baikadi, A. and R. E. Cardona-Rivera (2012). Towards Finding the Fundamental Unit of Narrative: A Proposal for the Narreme. In M. A. Finlayson, P. Gervás, D. Yuret, and F. Bex (Eds.), *Proceedings of the Third Workshop on Computational Models of Narrative*, Paris, France, pp. 42–44. European Language Resource Association.
- Baikadi, A., J. P. Rowe, B. W. Mott, and J. C. Lester (2012). Toward Narrative Schema-Based Goal Recognition Models for Interactive Narrative Environments. In S. G. Ware, J. Zhu, and R. Hodhod (Eds.), *Proceedings of the Fifth Workshop on Intelligent Narrative Technologies*, Menlo Park, CA, pp. 2–7. AAAI Press.
- Baikadi, A., J. P. Rowe, B. W. Mott, and J. C. Lester (2013). Improving Goal Recognition in Interactive Narratives with Models of Narrative Discovery Events. In M. Si, M. Cavazza, and A. Zook (Eds.), *Proceedings of the Sixth Workshop on Intelligent Narrative Technologies*, Menlo Park, CA. AAAI Press.
- Barber, H. and D. Kudenko (2007). Dynamic Generation of Dilemma-based Interactive Narratives. In J. Schaeffer and M. Mateas (Eds.), *Proceedings of the Third Conference*

- on *Artificial Intelligence in Interactive Digital Entertainment*, Menlo Park, CA, pp. 2–7. AAAI Press.
- Bejan, C. A. and S. Harabagiu (2010). Unsupervised Event Coreference Resolution with Rich Linguistic Features. In J. Hajic, S. Carberry, and S. Clark (Eds.), *Proceedings of the Fourty-Eighth Annual Meeting of the Association for Computational Linguistics*, Paris, France, pp. 1412–1422. Association for Computational Linguistics.
- Blaylock, N. and J. Allen (2006). Hierarchical Instantiated Goal Recognition. In G. Kaminka, D. V. Pynadath, and C. W. Geib (Eds.), *Proceedings of the Workshop on Modeling Others from Observations*, Menlo Park, CA, pp. 8–15. AAAI Press.
- Bui, H. H. (2003). A General Model for Online Probabilistic Plan Recognition. In G. Gottlob and T. Walsh (Eds.), *Proceedings of the Eighteenth International Joint Conference for Artificial Intelligence*, San Francisco, CA, pp. 1309–1318. Morgan Kaufmann.
- Bui, H. H., S. Venkatesh, and G. West (2002). Policy Recognition in the Abstract Hidden Markov Model. *Journal of Artificial Intelligence Research* 17(1), 451–499.
- Callaway, C. B. and J. C. Lester (2002). Narrative Prose Generation. *Artificial Intelligence* 139(2), 213–252.
- Carberry, S. (2001). Techniques for Plan Recognition. *User Modeling and User-Adapted Interaction* 11(1-2), 31–48.
- Card, O. S. (1988). *Characters and Viewpoint*. Cincinnati, OH: Writer’s Digest Books.
- Cavazza, M., F. Charles, and S. J. Mead (2002). Planning Characters’ Behaviour in Interactive Storytelling. *The Journal of Visualization and Computer Animation* 13(2), 121–131.
- Chambers, N. and D. Jurafsky (2008). Unsupervised Learning of Narrative Event Chains. In K. McKeown, J. D. Moore, S. Teufel, J. Allan, and S. Furui (Eds.), *Proceedings of the Fourty-Sixth Annual Meeting of the Association for Computational Linguistics*, pp. 789–797. Association for Computational Linguistics.
- Chambers, N. and D. Jurafsky (2009). Unsupervised Learning of Narrative Schemas and their Participants. In K.-Y. Su, J. Su, and J. Wiebe (Eds.), *Proceedings of the Fourty-Seventh Annual Meeting of the Association for Computational Linguistics*, pp. 602–611. Association for Computational Linguistics.
- Charniak, E. and R. P. Goldman (1993). A Bayesian Model of Plan Recognition. *Artificial Intelligence* 64(1), 53–79.

- Cox, M. and B. Kerkez (2006). Case-Based Plan Recognition With Novel Input. *Control and Intelligent Systems* 34(2), 96–104.
- Domingos, P., S. Kok, H. Poon, M. Richardson, and P. Singla (2006). Unifying Logical and Statistical AI. In Y. Gil and R. J. Mooney (Eds.), *Proceedings of the Twenty-First National Conference on Artificial Intelligence*, Menlo Park, CA, pp. 2—7. AAAI Press.
- Elson, D. K. (2012a). Detecting Story Analogies from Annotations of Time, Action and Agency. In M. A. Finlayson, P. Gervás, D. Yuret, and F. Bex (Eds.), *Proceedings of the Third Workshop on Computational Models of Narrative*, pp. 89–97. European Language Resource Association.
- Elson, D. K. (2012b). Dramabank: Annotating Agency in Narrative Discourse. In N. Calzolari, K. Choukri, T. Declerck, M. U. D. Piperidis, B. Maegaard, J. Mariani, J. Odiijk, and Stelios (Eds.), *Proceedings of the Eighth International Conference on Language Resources and Evaluation*, pp. 2813–2819. European Language Resource Association.
- Fairclough, C. R. and P. Cunningham (2003). A Multiplayer Case Based Story Engine. In Q. H. Mehdi, N. E. Gough, and S. Natkin (Eds.), *Proceedings of the Fourth International Conference on Intelligent Games and Simulation*, pp. 41–46. EUROSIS.
- Finlayson, M. A. (2009). Deriving Narrative Morphologies via Analogical Story Merging. In B. Kokinov, K. Holyoak, and D. Gentner (Eds.), *New Frontiers in Analogy Research (Proceedings of the 2nd International Conference on Analogy)*, Sofia, pp. 127–136. New Bulgarian University Press.
- Finlayson, M. A. (2011). The Story Workbench : An Extensible Semi-Automatic Text Annotation Tool. In E. Tomai, J. P. Rowe, and D. K. Elson (Eds.), *Proceedings of the Fourth Workshop on Intelligent Narrative Technologies*, Menlo park, CA, pp. 21–24. AAAI Press.
- Gal, Y., S. Reddy, S. M. Shieber, A. Rubin, and B. J. Grosz (2012). Plan Recognition in Exploratory Domains. *Artificial Intelligence* 176(1), 2270–2290.
- Geib, C. W. (2009). Delaying Commitment in Plan Recognition Using Combinatory Categorical Grammars. In C. Boutilier (Ed.), *Proceedings of the Twenty-First International Joint Conference on Artificial Intelligence*, Menlo Park, CA, pp. 1702–1707. AAAI Press.
- Geib, C. W. and R. P. Goldman (2009). A Probabilistic Plan Recognition Algorithm Based on Plan Tree Grammars. *Artificial Intelligence* 173(11), 1101–1132.

- Gervás, P. (2012). From the Fleece of Fact to Narrative Yarns : a Computational Model of Composition. In M. A. Finlayson, P. Gervás, D. Yuret, and F. Bex (Eds.), *Proceedings of the Third Workshop on Computational Models of Narrative*, pp. 123–131. European Language Resource Association.
- Gold, K. (2010). Training Goal Recognition Online from Low-Level Inputs in an Action-Adventure Game. In G. M. Youngblood and V. Bulitko (Eds.), *Proceedings of the Sixth Conference on Artificial Intelligence in Interactive Digital Entertainment*, Menlo Park, CA, pp. 21–26. AAAI Press.
- Goldman, R. P., C. W. Geib, and C. A. Miller (1999). A New Model of Plan Recognition. In K. B. Laskey and H. Prade (Eds.), *Fifteenth Conference Annual Conference on Uncertainty in Artificial Intelligence*, San Francisco, CA, pp. 245–254. Morgan Kaufmann.
- Goyal, A., E. Riloff, and H. I. Daume (2012). A Computational Model for Plot Units. *Computational Intelligence* 29(3), 466–488.
- Grasbon, D. and N. Braun (2001). A Morphological Approach to Interactive Storytelling. In M. Fleischmann and W. Strauss (Eds.), *Proceedings of CAST01 // Living in Mixed Realities*, Schloss Birlinghoven, pp. 337–340. FhG – ZPS.
- Ha, E. Y., A. Baikadi, C. J. Licata, B. W. Mott, and J. C. Lester (2010). Exploring the Effectiveness of Lexical Ontologies for Modeling Temporal Relations with Markov Logic Modeling Temporal Relations with Markov Logic. In *Proceedings of the Eleventh Annual SIGDIAL Meeting on Discourse and Dialogue*, Tokyo, Japan, pp. 75–78.
- Ha, E. Y., J. P. Rowe, B. W. Mott, and J. C. Lester (2011). Goal Recognition with Markov Logic Networks for Player-Adaptive Games. In V. Bulitko and M. O. Riedl (Eds.), *Proceedings of the Seventh Conference on Artificial Intelligence in Interactive Digital Entertainment*, Menlo Park, CA, pp. 32–39. AAAI Press.
- Hickey, D. T., A. A. Ingram-Goble, and E. M. Jameson (2009). Designing Assessments and Assessing Designs in Virtual Educational Environments. *Journal of Science Education and Technology* 18(2), 187–208.
- Hu, D. H. and Q. Yang (2008). CIGAR : Concurrent and Interleaving Goal and Activity Recognition. In D. Fox and C. P. Gomes (Eds.), *Proceedings of the Twenty-Third National Conference on Artificial Intelligence*, Menlo Park, CA, pp. 1363–1368.
- Inoue, N. and K. Inui (2011). ILP-Based Reasoning for Weighted Abduction. In D. Patti-son, D. Long, and C. Geib (Eds.), *Proceedings of the First Workshop on Goal, Activity and Plan Recognition*, Menlo Park, CA, pp. 25–32. AAAI Press.

- Johnson, W. L. (2010). Serious Use of a Serious Game for Language Learning. *Journal of Artificial Intelligence in Education* 20(2), 175–195.
- Jung, Y., J. Ryu, K.-m. Kim, and S.-H. Myaeng (2010). Automatic Construction of a Large-Scale Situation Ontology by Mining How-To Instructions from the Web. *Web Semantics: Science, Services and Agents on the World Wide Web* 8(2-3), 110–124.
- Kabanza, F., P. Bellefeuille, F. Bisson, A. R. Benaskeur, and H. Irandoust (2010). Opponent Behaviour Recognition for Real-Time Strategy Games. In G. Sukthankar, C. W. Geib, D. V. Pynadath, and H. H. Bui (Eds.), *Proceedings of the Workshop on Plan, Activity, and Intent Recognition*, Menlo Park, CA, pp. 29–36. AAAI Press.
- Kautz, H., B. Selman, and Y. Jiang (1997). *A General Stochastic Approach to Solving Problems with Hard and Soft Constraints*, Volume 35. American Mathematical Society.
- Kautz, H. A. (1991). A Formal Theory of Plan Recognition and its Implementation. In J. F. Allen, H. A. Kautz, R. Pelavin, and J. Tenenbergs (Eds.), *Reasoning About Plans*, Chapter 2, pp. 69–126. San Mateo, CA: Morgan Kaufmann.
- Kautz, H. A. and J. F. Allen (1986). Generalized Plan Recognition. In T. Kehler (Ed.), *Proceedings of the Fifth National Conference on Artificial Intelligence*, San Francisco, CA, pp. 32–37. Morgan Kaufmann.
- Ketelhut, D. J., C. Dede, J. Clarke, and B. C. Nelson (2010). A Multi-User Virtual Environment for Building Higher Order Inquiry Skills in Science. *British Journal of Educational Technology* 41(1), 56–68.
- Kok, S. and P. Domingos (2005). Learning the Structure of Markov Logic Networks. In L. D. Raedt and S. Wrobel (Eds.), *Proceedings of the Twenty-Second International Conference on Machine Learning*, New York, NY, pp. 441–448. ACM Press.
- Kok, S., M. Sumner, M. Richardson, P. Singla, H. Poon, D. Lowd, and P. Domingos (2007). The Alchemy System for Statistical Relational AI. Technical report, Department of Computer Science and Engineering, University of Washington, Seattle, WA.
- Lang, R. R. (1999). A Declarative Model for Simple Narratives. In M. Mateas, P. Sengers, K. Dautenhahn, C. Elliott, J. C. Lester, and C. Nehaniv (Eds.), *Proceedings of the AAAI Fall symposium on Narrative Intelligence*, Menlo Park, CA. AAAI Press.
- Laviers, K. and G. Sukthankar (2011). A Real-Time Opponent Modeling System for Rush Football. In T. Walsh (Ed.), *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence*, Menlo Park, CA, pp. 2476–2481. AAAI Press.

- Lehnert, W. G. (1981). Plot Units and Narrative Summarization. *Cognitive Science* 5(4), 293–331.
- León, C. and P. Gervás (2011). Prototyping the Use of Plot Curves to Guide Story Generation. In M. A. Finlayson, P. Gervás, D. Yuret, and F. Bex (Eds.), *Proceedings of the Third Workshop on Computational Models of Narrative*, pp. 152–156. European Language Resource Association.
- Lesh, N., C. Rich, and C. Sidner (1999). Using Plan Recognition in Human-Computer Collaboration. In J. Kay (Ed.), *Proceedings of the Seventh International Conference on User Modeling*, Secaucus, NJ, pp. 23–32. Springer-Verlag New York.
- Lester, J. C., H. a. Spires, J. L. Nietfeld, J. Minogue, B. W. Mott, and E. V. Lobene. Designing Game-Based Learning Environments for Elementary Science Education: A Narrative-Centered Learning Perspective.
- Li, B., S. Lee-urban, and M. O. Riedl (2012). Toward Autonomous Crowd-Powered Creation of Interactive Narratives. In S. G. Ware, J. Zhu, and R. Hodhod (Eds.), *Proceedings of the Fifth Workshop on Intelligent Narrative Technologies*, Menlo Park, CA, pp. 20–25. AAAI Press.
- Liu, H. and P. Singh (2002). MAKEBELIEVE : Using Commonsense Knowledge to Generate Stories. In R. Dechter and R. S. Sutton (Eds.), *Proceedings of the Eighteenth National Conference on Artificial Intelligence*, Menlo Park, CA, pp. 957–958. AAAI Press.
- Magerko, B. (2005). Story Representation and Interactive Drama. In R. M. Young and J. E. Laird (Eds.), *Proceedings of the First Conference on Artificial Intelligence in Interactive Digital Entertainment*, Menlo Park, CA, pp. 87–92. AAAI Press.
- Mani, I. (2004). Narrative Summarization. *Journal Traitement Automatique des Langues (TAL)*: 45(1), 15–38.
- Mani, I. (2010). Predicting Reader Response in Narrative. In A. Jhala, M. O. Riedl, and D. L. Roberts (Eds.), *Proceedings of the Third Workshop on Intelligent Narrative Technologies*, INT3 '10, New York, NY, pp. 1–5. ACM Press.
- Manshadi, M., R. Swanson, and A. S. Gordon (2008). Learning a Probabilistic Model of Event Sequences From Internet Weblog Stories. In D. Wilson and H. C. Lane (Eds.), *Proceedings of the Twenty-first Annual Meeting of the Florida Artificial Intelligence Research Society*, Menlo Park, CA, pp. 159–165. AAAI Press.
- Mateas, M. and A. Stern (2002). Towards Integrating Plot and Character for Interactive Drama. In K. Dautenhahn (Ed.), *Proceedings of the AAAI Fall Symposium on Socially Intelligent Agents*, Menlo Park, CA, pp. 113–118. AAAI Press.

- Mateas, M. and A. Stern (2005). Structuring Content in the Façade Interactive Drama Architecture. In R. M. Young and J. E. Laird (Eds.), *Proceedings of the First Conference on Artificial Intelligence and Interactive Digital Entertainment*, Menlo Park, CA, pp. 93–98. AAAI Press.
- Meech, J. (1999). Narrative Theories as Contextual Constraints for Agent Interaction. In M. Mateas, P. Sengers, K. Dautenhahn, C. Elliott, J. C. Lester, and C. Nehaniv (Eds.), *Proceedings of the AAAI Fall symposium on Narrative Intelligence*, Menlo Park, CA, pp. 38–43. AAAI Press.
- Montfort, N. (2006). Natural Language Generation and Narrative Variation in Interactive Fiction. In H. Liu and R. Mihalcea (Eds.), *Proceedings of the Workshop on Computational Aesthetics*, Menlo Park, CA, pp. 45–52. AAAI Press.
- Mott, B. W., S. Lee, and J. C. Lester (2006). Probabilistic Goal Recognition in Interactive Narrative Environments. In Y. Gil and R. J. Mooney (Eds.), *Proceedings of the Twenty-First National Conference on Artificial Intelligence*, Menlo Park, CA, pp. 187–192. AAAI Press.
- Nakasone, A. and M. Ishizuka (2006). SRST : A Storytelling Model Using Rhetorical Relations. In S. Göbel, R. Malkewitz, and I. Iurgel (Eds.), *Proceedings of the Third International Conference on Technologies for Interactive Digital Storytelling and Entertainment*, Berlin Heidelberg, Germany, pp. 127–138. Springer-Verlag.
- Nazerfard, E., B. Das, L. B. Holder, and D. J. Cook (2010). Conditional Random Fields for Activity Recognition in Smart Environments. In U. V. Catalyurek, G. Luo, H. Andrade, and N. R. Smalheiser (Eds.), *Proceedings of the First ACM International Health Informatics Symposium*, New York, NY, pp. 282–286. ACM Press.
- Nelson, M. J. and M. Mateas (2005). Search-Based Drama Management in the Interactive Fiction Anchorhead. In R. M. Young and J. Laid (Eds.), *Proceedings of the First Conference on Artificial Intelligence and Interactive Digital Entertainment*, Menlo Park, CA, pp. 99–104. AAAI Press.
- Ng, A. and S. Russell (2000). Algorithms for Inverse Reinforcement Learning. In P. Langley (Ed.), *Seventeenth International Conference on Machine Learning*, Volume 0, pp. 663–670. Morgan Kaufmann.
- Ng, H. T. and R. J. Mooney (1992). Abductive Plan Recognition and Diagnosis: A Comprehensive Empirical Evaluation. In *Third International Conference on Principles of Knowledge Representation and Reasoning*, Cambridge, MA, pp. 499–508.
- Ontanon, S. and J. Zhu (2011). The SAM Algorithm for Analogy-Based Story Generation. In V. Bulitko and M. O. Riedl (Eds.), *Proceedings of the Seventh Conference*

- on Artificial Intelligence in Interactive Digital Entertainment*, Menlo Park, CA, pp. 67–72. AAAI Press.
- Orkin, J., T. Smith, H. Reckman, and D. Roy (2010). Semi-Automatic Task Recognition for Interactive Narratives with EAT & RUN. In A. Jhala, M. O. Riedl, and D. L. Roberts (Eds.), *Proceedings of the Third Workshop on Intelligent Narrative Technologies*, New York, NY, pp. 1–8. ACM Press.
- Passonneau, R. J., A. Goodkind, and E. T. Levy (2007). Annotation of Children’s Oral Narrations: Modeling Emergent Narrative Skills for Computational Applications. In D. Wilson and G. Sutcliffe (Eds.), *Proceedings of the Twentieth Annual Meeting of the Florida Artificial Intelligence Research Society*, Menlo Park, CA, pp. 253–258. AAAI Press.
- Pattison, D. and D. Long (2011). Accurately Determining Intermediate and Terminal Plan States Using Bayesian Goal Recognition. In D. Pattison, D. Long, and C. Geib (Eds.), *Proceedings of the First Workshop on Goal, Activity and Plan Recognition*, Menlo Park, CA, pp. 32–37. AAAI Press.
- Peinado, F., P. Gervás, and B. Díaz-Agudo (2004). A Description Logic Ontology for Fairy Tale Generation. In T. Veale, A. Cardoso, F. C. Pereira, and P. Gervás (Eds.), *Proceedings of the Workshop on Language Resources for Linguistic Creativity*, pp. 56–61. European Language Resource Association.
- Poon, H. and P. Domingos (2006). Sound and Efficient Inference with Probabilistic and Deterministic Dependencies. In Y. Gil and R. J. Mooney (Eds.), *Proceedings of the Twenty-First National Conference on Artificial Intelligence*, Menlo Park, CA, pp. 458–463. AAAI Press.
- Porteous, J., M. Cavazza, and F. Charles (2010). Narrative Generation through Characters’ Point of View. In W. van der Hoek, G. A. Kaminka, Y. Lespérance, M. Luck, and S. Sen (Eds.), *Proceedings of the Ninth International Conference on Autonomous Agents and Multiagent Systems*, pp. 1297–1304. International Foundation for Autonomous Agents and Multiagent Systems.
- Propp, V. A. (1968). *Morphology of the Folktale*. Austin, TX: University of Texas Press.
- Pustejovsky, J., J. Castaño, R. Ingria, R. Saurí, R. Gaizauskas, A. Setzer, and G. Katz (2003). TimeML : Robust Specification of Event and Temporal Expressions in Text. In H. Bunt, Y. Girard, E. Krahmer, R. Morante, R. Muskens, I. van der Sluis, and E. Thijsse (Eds.), *Fifth International Workshop on Computational Semantics*, pp. 1–12.

- Pynadath, D. V. and M. P. Wellman (2000). Probabilistic State-Dependent Grammars for Plan Recognition. In C. Boutilier and M. Goldszmidt (Eds.), *Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence*, Corvallis, OR, pp. 507–514. IJAI Press.
- R Core Team (2013). *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing.
- Ramírez, M. and H. Geffner (2011). Goal Recognition over POMDPs : Inferring the Intention of a POMDP Agent. In T. Walsh (Ed.), *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence*, Menlo Park, CA, pp. 2009–2014. AAAI Press.
- Riedel, S. (2008). Improving the Accuracy and Efficiency of MAP Inference for Markov Logic. In D. A. McAllester and P. Myllymäki (Eds.), *Proceedings of the Twenty-Fourth Annual Conference on Uncertainty in Artificial Intelligence*, Corvallis, OR, pp. 468–475. IJAI Press.
- Riedl, M., C. Saretto, and R. Young (2003). Managing Interaction between Users and Agents in a Multi-Agent Storytelling Environment. In J. S. Rosenschein, M. Wooldridge, T. Sandholm, and M. Yokoo (Eds.), *Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems*, New York, NY, pp. 741–748. ACM Press.
- Riedl, M. O. (2004). *Narrative Planning: Balancing Plot and Character*. Ph.d thesis, North Carolina State University.
- Riedl, M. O. and R. M. Young (2005). Open-World Planning for Story Generation. In L. P. Kaelbling and A. Saffiotti (Eds.), *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence*, Menlo Park, CA, pp. 1719–1720. AAAI Press.
- Riedl, M. O. and R. M. Young (2006). From Linear Story Generation to Branching Story Graphs. *IEEE computer graphics and applications* 26(3), 23–31.
- Roberts, D. L., M. J. Nelson, C. L. Isbell, M. Mateas, and M. L. Littman (2006). Targeting Specific Distributions of Trajectories in MDPs. In Y. Gil and R. J. Mooney (Eds.), *Proceedings of the Twenty-First National Conference on Artificial Intelligence*, Menlo Park, CA, pp. 1213–1218. AAAI Press.
- Rowe, J. P. (2013). *Narrative-Centered Tutorial Planning with Concurrent Markov Decision Processes*. Ph.d thesis, North Carolina State University.

- Rowe, J. P., L. R. Shores, B. W. Mott, and J. C. Lester (2011). Integrating Learning, Problem Solving, and Engagement in Narrative-Centered Learning Environments. *International Journal of Artificial Intelligence in Education* 21(1-2), 115–133.
- Sabourin, J., B. Mott, and J. Lester (2011). Discovering Behavior Patterns of Self-Regulated Learners in an Inquiry-Based Learning Environment. In *Proceedings of the Sixteenth International Conference on Artificial Intelligence in Education*, pp. 209–218.
- Sabourin, J., J. Rowe, B. Mott, and J. Lester (2011). When Off-Task is On-Task: The Affective Role of Off-Task Behavior in Narrative-Centered Learning Environments. In *Proceedings of the Fifteenth International Conference on Artificial Intelligence in Education*, Auckland, New Zealand, pp. 534–536.
- Sack, G. (2012). Character Networks for Narrative Generation. In S. G. Ware, J. Zhu, and R. Hodhod (Eds.), *Proceedings of the Fifth Workshop on Intelligent Narrative Technologies*, Menlo Park, CA, pp. 38–43. AAAI Press.
- Sadilek, A. and H. Kautz (2012). Location-Based Reasoning about Complex Multi-Agent Behavior. *Journal of Artificial Intelligence Research* 43, 87–133.
- Shank, R. C. and R. P. Abelson (1975). *Scripts, plans, and knowledge*. Yale University.
- Si, M., S. C. Marsella, and M. O. Riedl (2008). Integrating Story-centric and Character-centric Processes for Authoring Interactive Drama. In C. Darken and M. Mateas (Eds.), *Proceedings of the Fourth Conference on Artificial Intelligence in Interactive Digital Entertainment*, Menlo Park, CA, pp. 203–208. AAAI Press.
- Singla, P. and P. Domingos (2005). Discriminative Training of Markov Logic Networks. In M. M. Veloso and S. Kambhampati (Eds.), *Proceedings of the Twentieth National Conference on Artificial Intelligence*, Menlo Park, CA, pp. 868–873. AAAI Press.
- Singla, P. and P. Domingos (2006). Memory-Efficient Inference in Relational Domains. In Y. Gil and R. J. Mooney (Eds.), *Proceedings of the Twenty-First National Conference on Artificial Intelligence*, Menlo Park, CA, pp. 488–493. AAAI Press.
- Singla, P. and P. Domingos (2008). Lifted First-Order Belief Propagation. In D. Fox and C. P. Gomes (Eds.), *Proceedings of the Twenty-Third National Conference on Artificial Intelligence*, Number Pearl, Menlo Park, CA, pp. 1094–1099. AAAI Press.
- Singla, P. and R. J. Mooney (2011). Abductive Markov Logic for Plan Recognition. In W. Burgard and D. Roth (Eds.), *Twenty-fifth National Conference on Artificial Intelligence*, Menlo Park, CA, pp. 1069–1075. AAAI Press.

- Sukthankar, G. and K. Sycara (2005). Automatic Recognition of Human Team Behaviors. In G. A. Kaminka, D. V. Pynadath, and C. W. Geib (Eds.), *Proceedings of the Workshop on Modeling Others from Observations*, Menlo Park, CA. AAAI Press.
- Swartjes, I., J. Vromen, and N. Bloom (2007). Narrative Inspiration: Using Case Based Problem Solving for Emergent Story Generation. In A. Cardoso and G. A. Wiggins (Eds.), *Proceedings of the Fourth Workshop on Computational Creativity*, pp. 21–30.
- Theune, M., S. Faas, A. Nijholt, and D. Heylen (2003). The Virtual Storyteller: Story Creation by Intelligent Agents. In S. Göbel, N. Braun, and U. Spierling (Eds.), *Proceedings of the First International Conference on Technologies for Interactive Digital Storytelling and Entertainment*, Stuttgart, Germany, pp. 204–215. Fraunhofer IRB Verlag.
- Thomas, J. M. and R. M. Young (2010). Annie: Automated Generation of Adaptive Learner Guidance for Fun Serious Games. *IEEE Transactions on Learning Technologies* 3(4), 329–343.
- Thue, D. and V. Bulitko (2012). Procedural Game Adaptation: Framing Experience Management as Changing an MDP. In M. O. Reidl and G. Sukthankar (Eds.), *Proceedings of the Eighth Conference on Artificial Intelligence and Interactive Digital Entertainment*, Menlo Park, CA, pp. 44–50. AAAI Press.
- Tuffield, M. M., D. E. Millard, and N. R. Shadbolt (2006). Ontological Approaches to Modelling Narrative. In *Proceedings of the Second Advanced Knowledge Technologies Doctoral Consortium*, pp. 62–69.
- Ware, S. G., R. M. Young, B. Harrison, and D. L. Roberts (2012). Four Quantitative Metrics Describing Narrative Conflict. In D. Oyarzun, F. Peinad, R. M. Young, A. Elizalde, and G. Méndez (Eds.), *Proceedings of the Fifth International Conference on Interactive Digital Storytelling*, pp. 18–29. Springer.
- Weyhrauch, P. (1997). *Guiding Interactive Drama*. Ph.d thesis, Carnegie Mellon University.
- Wu, T.-y., C.-c. Lian, and J. Y.-j. Hsu (2007). Joint Recognition of Multiple Concurrent Activities using Factorial Conditional Random Fields. In D. V. Pynadath and C. W. Geib (Eds.), *Proceedings of the Workshop on Plan, Activity, and Intent Recognition*, Menlo Park, CA, pp. 82–88. AAAI Press.

Appendices

Appendix A

Model Files for Crystal Island: Outbreak

A.1 Data Definitions

```
/*  
MLN Definition for our Plan Reco work, which includes  
1. Predicates (observed, hidden, and global)  
2. Formulae  
3. Declaration of observed, hidden, global for the predicates  
*/  
  
/* 1. Define predicates */  
// (1) Observed predicates  
predicate userId: Id;  
predicate action: Int x Action;
```

```

predicate loc:  Int x Loc;
predicate state: Int x State;
predicate arg:  Int x ArgType x ArgValue;

// (2) Hidden predicates
predicate goal: Int x Goal;

include "model.pml";

/* 3. Define which predicates are hidden, observed and global. */
observed:  userId, action, loc, state, arg;
hidden:    goal;

```

A.2 State of the Art Model

```

/**** Hard Formulae ****/
/* (1) */
factor[2]: for Int i, Action a
if action(i, _) & i >= 0:
|Goal g: goal(i, g)| >= 1;

factor[3]: for Int i, Action a
if action(i, _) & i >= 0:
|Goal g: goal(i, g)| <= 1;

```

```

/**** Soft Formulae ****/

/* (2) */

weight prior:  Goal -> Double;

factor:  for Int i, Goal g
if action(i, _) & i >= 0
add [goal(i, g)] * prior(g);

/* (3) */

weight cur_action:  Action x Goal -> Double;

factor:  for Int i, Action a, Goal g
if action(i, a) & i >= 0
add [goal(i, g)] * cur_action(a, g);
set collector.cutoff.cur_action = 10;

/* (4) */

weight cur_loc:  Loc x Goal -> Double;

factor:  for Int i, Loc l, Goal g
if loc(i, l) & i >= 0
add [goal(i, g)] * cur_loc(l, g);
set collector.cutoff.cur_loc = 10;

/* (5) */

weight cur_state:  State x Goal -> Double;

factor:  for Int i, State s, Goal g

```

```

if state(i, s) & i >= 0
add [goal(i, g)] * cur_state(s, g);
set collector.cutoff.cur_state = 10;

/* (6) */
weight cur_action_state: Action x State x Goal -> Double;
factor: for Int i, Action a, State s, Goal g
if action(i, a) & i >= 0 & state(i, s)
add [goal(i, g)] * cur_action_state(a, s, g);
set collector.cutoff.cur_action_state = 10;

/* (7) */
weight prev_action: Action x Goal -> Double;
factor: for Int i, Action a, Goal g
if action(i-1, a) & i >= 0 & action(i, _)
add [goal(i, g)] * prev_action(a, g);
set collector.cutoff.prev_action = 10;

/* (8) */
weight prev_loc: Loc x Goal -> Double;
factor: for Int i, Loc l, Goal g
if loc(i-1, l) & i >= 0 & loc(i, _)
add [goal(i, g)] * prev_loc(l, g);
set collector.cutoff.prev_loc = 10;

```

```

/* (9) */
weight prev_state: State x Goal -> Double;
factor: for Int i, State s, Goal g
if state(i-1, s) & i >= 0 & state(i, _)
add [goal(i, g)] * prev_state(s, g);
set collector.cutoff.prev_state = 10;

/* (10) */
weight prev_action_state: Action x State x Goal -> Double;
factor: for Int i, Action a, State s, Goal g
if action(i-1, a) & i >= 0 & state(i-1, s) & action(i, _)
add [goal(i, g)] * prev_action_state(a, s, g);
set collector.cutoff.prev_action_state = 10;

/* (11) */
weight prev_cur_action: Action x Action x Goal -> Double;
factor: for Int i, Action a, Action a_prev, Goal g
if action(i, a) & i >= 0 & action(i-1, a_prev)
add [goal(i, g)] * prev_cur_action(a_prev, a, g);
set collector.cutoff. prev_cur_action = 10;

/* (12) */
weight prev_cur_goal: Goal -> Double+;
factor[1]: for Int i, Action a1, Action a2, Goal g1, Goal g2
if action(i-1, a1) & action(i, a2) & i >= 0 & a1 != a2 & g1 == g2

```

```

add [goal(i-1, g1) => goal(i, g2)] * prev_cur_goal(g1);

/* (13) */

weight preva_cura_prevg_curg: Action x Action x Goal -> Double+;
factor[1]: for Int i, Action a1, Action a2, Goal g1, Goal g2
if action(i-1, a1) & action(i, a2) & i >= 0 & a1 != a2 & g1 == g2
add [goal(i-1, g1) => goal(i, g2)] * preva_cura_prevg_curg(a1, a2, g1);

```

A.3 Discovery Events Model

```

/**** Hard Formulae ****/

/* (1) */

factor[2]: for Int i, Action a
if action(i, _) & i >= 0:
|Goal g: goal(i, g)| >= 1;

factor[3]: for Int i, Action a
if action(i, _) & i >= 0:
|Goal g: goal(i, g)| <= 1;

factor[4]: for Int i, Goal g
if action(i,_) & action(i-1,_) & action(i+1,_) & i >= 0:
goal(i-1,g) & goal(i+1,g) => goal(i,g);

```

```

/**** Soft Formulae ****/

/* (2) */
weight prior: Goal -> Double;
factor: for Int i, Goal g
if action(i, _) & i >= 0
add [goal(i, g)] * prior(g);

/* (3) */
weight cur_action: Action x Goal -> Double;
factor: for Int i, Action a, Goal g
if action(i, a) & i >= 0
add [goal(i, g)] * cur_action(a, g);
set collector.cutoff.cur_action = 10;

/* (4) */
weight cur_loc: Loc x Goal -> Double;
factor: for Int i, Loc l, Goal g
if loc(i, l) & i >= 0
add [goal(i, g)] * cur_loc(l, g);
set collector.cutoff.cur_loc = 10;

/* (5) */
weight cur_state: State x Goal -> Double;
factor: for Int i, State s, Goal g
if state(i, s) & i >= 0

```

```

add [goal(i, g)] * cur_state(s, g);
set collector.cutoff.cur_state = 10;

weight worksheet: Goal -> Double;
factor: for Int i, Goal g
if action(i,_) & i >= 1
& (|Int j: (j <= i & action(j,"WORKSHEET")) | >= 1)
add [goal(i,g)] * worksheet(g);
set collector.cutoff.worksheet = 10;

weight tested_objects: Goal -> Double;
factor: for Int i, Goal g
if action(i,_) & i >= 1
& (|Int j: (j <= i & action(j,"TESTCOMPUTER")) | >= 1)
add [goal(i,g)] * tested_objects(g);
set collector.cutoff.tested_objects = 10;

weight eating_habits: Goal -> Double;
factor: for Int i, Goal g
if action(i,_) & i >= 1
& (|Int j : (j <= i & arg(j,_, "eating-habbits")) | >= 1)
add [goal(i,g)] * eating_habits(g);
set collector.cutoff.eating_habits = 10;

weight read_salmonellosis: Goal -> Double;

```

```

factor: for Int i, Goal g
if action(i,_) & i >= 1
& (|Int j: (j <= i & action(j,"READ") i
& arg(j,"obj","salmonellosis"))| >= 1)
add [goal(i,g)] * read_salmonellosis(g);
set collector.cutoff.read_salmonellosis = 10;

weight observed_symptoms: Goal -> Double;
factor: for Int i, Goal g
if action(i,_) & i >= 1
& (|Int j: (j <= i & action(j, "DIALOG")
& arg(j, "dialog", "symptom"))| >= 1)
add [goal(i,g)] * observed_symptoms(g);
set collector.cutoff.observed_symptoms = 10;

weight learned_bacteria: Goal -> Double;
factor: for Int i, Goal g
if action(i,_) & i >= 1 & (|Int j: (j <= i & arg(j,_, "bacteria"))| <= 1)
add [goal(i,g)] * learned_bacteria(g);
set collector.cutoff.learned_bacteria = 10;

```

Appendix B

Model Files for Crystal Island: Uncharted Discovery

B.1 Data Definitions

```
/*  
MLN Definition for our Plan Reco work, which includes  
1. Predicates (observed, hidden, and global)  
2. Formulae  
3. Declaration of observed, hidden, global for the predicates  
*/  
  
/* 1. Define predicates */  
// (1) Observed predicates  
predicate userId: Id;  
predicate action: Int x Action;
```

```

predicate location:  Int x Loc;
predicate argument:  Int x Arg;
predicate extra:    Int x Key x Value;

// (2) Hidden predicates
predicate goal:     Int x Goal;

include "model.pml";

/* 3. Define which predicates are hidden, observed and global. */
observed:  userId, action, location, argument, extra;
hidden:    goal;

```

B.2 State of the Art Model

```

/**** Hard Formulae ****/
/* (1) */
factor[2]:  for Int i, Action a
if action(i, _) & i >= 0:
|Goal g:  goal(i, g)| >= 1;

factor[3]:  for Int i, Action a
if action(i, _) & i >= 0:
|Goal g:  goal(i, g)| <= 1;

```

```

/**** Soft Formulae ****/

/* (2) */

weight prior:  Goal -> Double;

factor:  for Int i, Goal g
if action(i, _) & i >= 0
add [goal(i, g)] * prior(g);

/* (3) */

weight cur_action:  Action x Goal -> Double;

factor:  for Int i, Action a, Goal g
if action(i, a) & i >= 0
add [goal(i, g)] * cur_action(a, g);
set collector.cutoff.cur_action = 10;

/* (4) */

weight cur_loc:  Loc x Goal -> Double;

factor:  for Int i, Loc l, Goal g
if location(i, l) & i >= 0
add [goal(i, g)] * cur_loc(l, g);
set collector.cutoff.cur_loc = 10;

/* (5) */

weight cur_state:  State x Goal -> Double;

factor:  for Int i, State s, Goal g

```

```

if state(i, s) & i >= 0
add [goal(i, g)] * cur_state(s, g);
set collector.cutoff.cur_state = 10;

/* (6) */
weight cur_action_state: Action x State x Goal -> Double;
factor: for Int i, Action a, State s, Goal g
if action(i, a) & i >= 0 & state(i, s)
add [goal(i, g)] * cur_action_state(a, s, g);
set collector.cutoff.cur_action_state = 10;

/* (7) */
weight prev_action: Action x Goal -> Double;
factor: for Int i, Action a, Goal g
if action(i-1, a) & i >= 1 & action(i, _)
add [goal(i, g)] * prev_action(a, g);
set collector.cutoff.prev_action = 10;

/* (8) */
weight prev_loc: Loc x Goal -> Double;
factor: for Int i, Loc l, Goal g
if location(i-1, l) & i >= 1 & location(i, _)
add [goal(i, g)] * prev_loc(l, g);
set collector.cutoff.prev_loc = 10;

```

```

/* (9) */
weight prev_state: State x Goal -> Double;
factor: for Int i, State s, Goal g
if state(i-1, s) & i >= 0 & state(i, _)
add [goal(i, g)] * prev_state(s, g);
set collector.cutoff.prev_state = 10;

/* (10) */
weight prev_action_state: Action x State x Goal -> Double;
factor: for Int i, Action a, State s, Goal g
if action(i-1, a) & i >= 0 & state(i-1, s) & action(i, _)
add [goal(i, g)] * prev_action_state(a, s, g);
set collector.cutoff.prev_action_state = 10;

/* (11) */
weight prev_cur_action: Action x Action x Goal -> Double;
factor: for Int i, Action a, Action a_prev, Goal g
if action(i, a) & i >= 1 & action(i-1, a_prev)
add [goal(i, g)] * prev_cur_action(a_prev, a, g);
set collector.cutoff. prev_cur_action = 10;

/* (12) */
weight prev_cur_goal: Goal -> Double+;
factor[1]: for Int i, Action a1, Action a2, Goal g1, Goal g2
if action(i-1, a1) & action(i, a2) & i >= 1 & a1 != a2 & g1 == g2

```

```

add [goal(i-1, g1) => goal(i, g2)] * prev_cur_goal(g1);

/* (13) */

weight preva_cura_prevg_curg: Action x Action x Goal -> Double+;
factor[1]: for Int i, Action a1, Action a2, Goal g1, Goal g2
if action(i-1, a1) & action(i, a2) & i >= 1 & a1 != a2 & g1 == g2
add [goal(i-1, g1) => goal(i, g2)] * preva_cura_prevg_curg(a1, a2, g1);

```

B.3 Discovery Events Model

```

/**** Hard Formulae ****/

/* (1) */

factor[2]: for Int i, Action a
if action(i, _) & i >= 0:
|Goal g: goal(i, g)| >= 1;

factor[3]: for Int i, Action a
if action(i, _) & i >= 0:
|Goal g: goal(i, g)| <= 1;

/**** Soft Formulae ****/

/* (2) */

weight prior: Goal -> Double;
factor: for Int i, Goal g

```

```

if action(i, _) & i >= 0
add [goal(i, g)] * prior(g);

/* (3) */
weight cur_action: Action x Goal -> Double;
factor: for Int i, Action a, Goal g
if action(i, a) & i >= 0
add [goal(i, g)] * cur_action(a, g);
set collector.cutoff.cur_action = 10;

/* (4) */
weight cur_loc: Loc x Goal -> Double;
factor: for Int i, Loc l, Goal g
if location(i, l) & i >= 0
add [goal(i, g)] * cur_loc(l, g);
set collector.cutoff.cur_loc = 10;

/* (5) */
weight cur_state: State x Goal -> Double;
factor: for Int i, State s, Goal g
if state(i, s) & i >= 0
add [goal(i, g)] * cur_state(s, g);
set collector.cutoff.cur_state = 10;

/* (6) */

```

```

weight cur_action_state: Action x State x Goal -> Double;
factor: for Int i, Action a, State s, Goal g
if action(i, a) & i >= 0 & state(i, s)
add [goal(i, g)] * cur_action_state(a, s, g);
set collector.cutoff.cur_action_state = 10;

/* Additional Discovery Events */

/* Picked up flags */
weight pickup_flags: Goal -> Double;
factor: for Int i, Goal g
if action(i,_) & i >= 1
& (|Int j: (j <= i & action(j,"FlagManagerPickupFlag"))| >= 1)
add [goal(i,g)] * pickup_flags(g);
set collector.cutoff.pickup_flags = 10;

/* Drop flags */
weight drop_flags: Goal -> Double;
factor: for Int i, Goal g
if action(i,_) & i >= 1
& (|Int j: (j <= i & action(j,"FlagManagerDropFlag"))| >= 1)
add [goal(i,g)] * drop_flags(g);
set collector.cutoff.drop_flags = 10;

/* Take Notes */

```

```

weight notes: Goal -> Double;
factor: for Int i, Goal g
if action(i,_) & i >= 1
& (|Int j: (j <= i & action(j,"PhotoJournalApp"))
& argument(i,"AddNewNote"))|>=1)
add [goal(i,g)] * notes(g);
set collector.cutoff.notes = 10;

```

```

weight update_notes: Goal -> Double;
factor: for Int i, Goal g
if action(i,_) & i >= 1
& (|Int j: (j <= i & action(j,"PhotoJournalApp"))
& argument(i,"NoteUpdated"))|>=1)
add [goal(i,g)] * update_notes(g);
set collector.cutoff.update_notes = 10;

```

```

/* Pickup a Sign */
weight pickup_sign: Goal -> Double;
factor: for Int i, Goal g
if action(i,_) & i >= 1
& (|Int j: (j <= i & action(j,"SignManagerPickupSign"))|>= 1)
add [goal(i,g)] * pickup_sign(g);
set collector.cutoff.pickup_sign = 10;

```

```

/* Take a photo */

```

```

weight photo: Goal -> Double;
factor: for Int i, Goal g
if action(i,_) & i >= 1
& (|Int j: (j <= i & action(j,"CameraApp"))| >= 1)
add [goal(i,g)] * photo(g);
set collector.cutoff.photo = 10;

/* Use IslandPedia */
weight reference: Goal -> Double;
factor: for Int i, Goal g
if action(i,_) & i >= 1
& (|Int j : (j <= i & action(j,"ReferenceApp"))| >= 1)
add [goal(i,g)] * reference(g);
set collector.cutoff.reference = 10;

/* Start Quests */
weight start_navigation1: Goal -> Double;
factor: for Int i, Goal g
if action(i,_) & i >= 1
& (|Int j: (j <= i & action(j,"QuestBegin")
& argument(j,"Navigation1"))|>= 1)
add [goal(i,g)] * start_navigation1(g);
set collector.cutoff.start_navigation1 = 10;

weight start_navigation2: Goal -> Double;

```

```

factor: for Int i, Goal g
if action(i,_) & i >= 1
& (|Int j: (j <= i & action(j,"QuestBegin")
& argument(j,"Navigation2"))|>= 1)
add [goal(i,g)] * start_navigation2(g);
set collector.cutoff.start_navigation2 = 10;

```

```

weight start_landforms1: Goal -> Double;
factor: for Int i, Goal g
if action(i,_) & i >= 1
& (|Int j: (j <= i & action(j,"QuestBegin")
& argument(j,"Landforms1"))|>= 1)
add [goal(i,g)] * start_landforms1(g);
set collector.cutoff.start_landforms1 = 10;

```

```

weight start_landforms2: Goal -> Double;
factor: for Int i, Goal g
if action(i,_) & i >= 1
& (|Int j: (j <= i & action(j,"QuestBegin")
& argument(j,"Landforms2"))|>= 1)
add [goal(i,g)] * start_landforms2(g);
set collector.cutoff.start_landforms2 = 10;

```

Appendix C

Extra Model Files

C.1 Ablated Experiment

C.1.1 Ablated Only

```
/** Hard Formulae **/  
/* (1) */  
factor[2]: for Int i, Action a  
if action(i, _) & i >= 0:  
|Goal g: goal(i, g)| >= 1;  
  
factor[3]: for Int i, Action a  
if action(i, _) & i >= 0:  
|Goal g: goal(i, g)| <= 1;  
  
factor[4]: for Int i, Goal g
```

```
if action(i,_) & action(i-1,_) & action(i+1,_) & i >= 0:  
goal(i-1,g) & goal(i+1,g) => goal(i,g);
```

```
/** Soft Formulae **/
```

```
/* (2) */
```

```
weight prior: Goal -> Double;
```

```
factor: for Int i, Goal g
```

```
if action(i, _) & i >= 0
```

```
add [goal(i, g)] * prior(g);
```

```
/* (3) */
```

```
weight cur_action: Action x Goal -> Double;
```

```
factor: for Int i, Action a, Goal g
```

```
if action(i, a) & i >= 0
```

```
add [goal(i, g)] * cur_action(a, g);
```

```
set collector.cutoff.cur_action = 10;
```

```
/* (4) */
```

```
weight cur_loc: Loc x Goal -> Double;
```

```
factor: for Int i, Loc l, Goal g
```

```
if loc(i, l) & i >= 0
```

```
add [goal(i, g)] * cur_loc(l, g);
```

```
set collector.cutoff.cur_loc = 10;
```

```
/* (5) */
```

```

weight cur_state: State x Goal -> Double;
factor: for Int i, State s, Goal g
if state(i, s) & i >= 0
add [goal(i, g)] * cur_state(s, g);
set collector.cutoff.cur_state = 10;

```

C.1.2 Baseline with Discovery Events

```

/**** Hard Formulae ****/
/* (1) */
factor[2]: for Int i, Action a
if action(i, _) & i >= 0:
|Goal g: goal(i, g)| >= 1;

factor[3]: for Int i, Action a
if action(i, _) & i >= 0:
|Goal g: goal(i, g)| <= 1;

/**** Soft Formulae ****/
/* (2) */
weight prior: Goal -> Double;
factor: for Int i, Goal g
if action(i, _) & i >= 0
add [goal(i, g)] * prior(g);

```

```
/* (3) */  
weight cur_action: Action x Goal -> Double;  
factor: for Int i, Action a, Goal g  
if action(i, a) & i >= 0  
add [goal(i, g)] * cur_action(a, g);  
set collector.cutoff.cur_action = 10;
```

```
/* (4) */  
weight cur_loc: Loc x Goal -> Double;  
factor: for Int i, Loc l, Goal g  
if loc(i, l) & i >= 0  
add [goal(i, g)] * cur_loc(l, g);  
set collector.cutoff.cur_loc = 10;
```

```
/* (5) */  
weight cur_state: State x Goal -> Double;  
factor: for Int i, State s, Goal g  
if state(i, s) & i >= 0  
add [goal(i, g)] * cur_state(s, g);  
set collector.cutoff.cur_state = 10;
```

```
/* (6) */  
weight cur_action_state: Action x State x Goal -> Double;  
factor: for Int i, Action a, State s, Goal g  
if action(i, a) & i >= 0 & state(i, s)
```

```
add [goal(i, g)] * cur_action_state(a, s, g);
set collector.cutoff.cur_action_state = 10;
```

```
/* (7) */
```

```
weight prev_action: Action x Goal -> Double;
factor: for Int i, Action a, Goal g
if action(i-1, a) & i >= 0 & action(i, _)
add [goal(i, g)] * prev_action(a, g);
set collector.cutoff.prev_action = 10;
```

```
/* (8) */
```

```
weight prev_loc: Loc x Goal -> Double;
factor: for Int i, Loc l, Goal g
if loc(i-1, l) & i >= 0 & loc(i, _)
add [goal(i, g)] * prev_loc(l, g);
set collector.cutoff.prev_loc = 10;
```

```
/* (9) */
```

```
weight prev_state: State x Goal -> Double;
factor: for Int i, State s, Goal g
if state(i-1, s) & i >= 0 & state(i, _)
add [goal(i, g)] * prev_state(s, g);
set collector.cutoff.prev_state = 10;
```

```
/* (10) */
```

```

weight prev_action_state: Action x State x Goal -> Double;
factor: for Int i, Action a, State s, Goal g
if action(i-1, a) & i >= 0 & state(i-1, s) & action(i, _)
add [goal(i, g)] * prev_action_state(a, s, g);
set collector.cutoff.prev_action_state = 10;

/* (11) */
weight prev_cur_action: Action x Action x Goal -> Double;
factor: for Int i, Action a, Action a_prev, Goal g
if action(i, a) & i >= 0 & action(i-1, a_prev)
add [goal(i, g)] * prev_cur_action(a_prev, a, g);
set collector.cutoff.prev_cur_action = 10;

/* (12) */
weight prev_cur_goal: Goal -> Double+;
factor[1]: for Int i, Action a1, Action a2, Goal g1, Goal g2
if action(i-1, a1) & action(i, a2) & i >= 0 & a1 != a2 & g1 == g2
add [goal(i-1, g1) => goal(i, g2)] * prev_cur_goal(g1);

/* (13) */
weight preva_cura_prevg_curg: Action x Action x Goal -> Double+;
factor[1]: for Int i, Action a1, Action a2, Goal g1, Goal g2
if action(i-1, a1) & action(i, a2) & i >= 0 & a1 != a2 & g1 == g2
add [goal(i-1, g1) => goal(i, g2)] * preva_cura_prevg_curg(a1, a2, g1);

```

```

weight worksheet: Goal -> Double;
factor: for Int i, Goal g
if action(i,_) & i >= 1
& (|Int j: (j <= i & action(j,"WORKSHEET")) | >= 1)
add [goal(i,g)] * worksheet(g);
set collector.cutoff.worksheet = 10;

weight tested_objects: Goal -> Double;
factor: for Int i, Goal g
if action(i,_) & i >= 1
& (|Int j: (j <= i & action(j,"TESTCOMPUTER")) | >= 1)
add [goal(i,g)] * tested_objects(g);
set collector.cutoff.tested_objects = 10;

weight eating_habits: Goal -> Double;
factor: for Int i, Goal g
if action(i,_) & i >= 1
& (|Int j : (j <= i & arg(j,_, "eating-habbits")) | >= 1)
add [goal(i,g)] * eating_habits(g);
set collector.cutoff.eating_habits = 10;

weight read_salmonellosis: Goal -> Double;
factor: for Int i, Goal g
if action(i,_) & i >= 1
& (|Int j: (j <= i & action(j,"READ"))

```

```

& arg(j,"obj","salmonellosis"))| >= 1)
add [goal(i,g)] * read_salmonellosis(g);
set collector.cutoff.read_salmonellosis = 10;

weight observed_symptoms: Goal -> Double;
factor: for Int i, Goal g
if action(i,_) & i >= 1
& (|Int j: (j <= i & action(j, "DIALOG")
& arg(j, "dialog", "symptom"))| >= 1)
add [goal(i,g)] * observed_symptoms(g);
set collector.cutoff.observed_symptoms = 10;

weight learned_bacteria: Goal -> Double;
factor: for Int i, Goal g
if action(i,_) & i >= 1 & (|Int j: (j <= i & arg(j,_"bacteria"))| <= 1)
add [goal(i,g)] * learned_bacteria(g);
set collector.cutoff.learned_bacteria = 10;

```

C.2 Representation Experiment

C.2.1 Crystal Island: Uncharted Discovery

with Milestone States

```
/** Hard Formulae */
/* (1) */
factor[2]: for Int i, Action a
if action(i, _) & i >= 0:
|Goal g: goal(i, g)| >= 1;

factor[3]: for Int i, Action a
if action(i, _) & i >= 0:
|Goal g: goal(i, g)| <= 1;

/** Soft Formulae */
/* (2) */
weight prior: Goal -> Double;
factor: for Int i, Goal g
if action(i, _) & i >= 0
add [goal(i, g)] * prior(g);

/* (3) */
weight cur_action: Action x Goal -> Double;
factor: for Int i, Action a, Goal g
```

```

if action(i, a) & i >= 0
add [goal(i, g)] * cur_action(a, g);
set collector.cutoff.cur_action = 10;

/* (4) */
weight cur_loc:  Loc x Goal -> Double;
factor:  for Int i, Loc l, Goal g
if location(i, l) & i >= 0
add [goal(i, g)] * cur_loc(l, g);
set collector.cutoff.cur_loc = 10;

/* (7) */
weight prev_action:  Action x Goal -> Double;
factor:  for Int i, Action a, Goal g
if action(i-1, a) & i >= 1 & action(i, _)
add [goal(i, g)] * prev_action(a, g);
set collector.cutoff.prev_action = 10;

/* (8) */
weight prev_loc:  Loc x Goal -> Double;
factor:  for Int i, Loc l, Goal g
if location(i-1, l) & i >= 1 & location(i, _)
add [goal(i, g)] * prev_loc(l, g);
set collector.cutoff.prev_loc = 10;

```

```

/* (11) */
weight prev_cur_action: Action x Action x Goal -> Double;
factor: for Int i, Action a, Action a_prev, Goal g
if action(i, a) & i >= 1 & action(i-1, a_prev)
add [goal(i, g)] * prev_cur_action(a_prev, a, g);
set collector.cutoff. prev_cur_action = 10;

/* (12) */
weight prev_cur_goal: Goal -> Double+;
factor[1]: for Int i, Action a1, Action a2, Goal g1, Goal g2
if action(i-1, a1) & action(i, a2) & i >= 1 & a1 != a2 & g1 == g2
add [goal(i-1, g1) => goal(i, g2)] * prev_cur_goal(g1);

/* (13) */
weight preva_cura_prevg_curg: Action x Action x Goal -> Double+;
factor[1]: for Int i, Action a1, Action a2, Goal g1, Goal g2
if action(i-1, a1) & action(i, a2) & i >= 1 & a1 != a2 & g1 == g2
add [goal(i-1, g1) => goal(i, g2)] * preva_cura_prevg_curg(a1, a2, g1);

/* Narrative State */

/* Stage 1: Completing the Tutorial */
weight complete_tutorial: Goal -> Double;
factor: for Int i, Goal g
if action(i,_) & i >= 1

```

```

& (|Int j : (j <= i & action(j,"TutorialCompleted"))| >= 1)
add [goal(i,g)] * complete_tutorial(g);
set collector.cutoff.complete_tutorial = 10;

/* Stage 2: Landform Quests */
weight complete_landform1: Goal -> Double;
factor: for Int i, Goal g
if action(i,_) & i >= 1
& (|Int j : (j <= i & action(j,"QuestComplete")
& argument(i,"Landforms1"))| >= 1)
add [goal(i,g)] * complete_landform1(g);
set collector.cutoff.complete_landform1 = 10;

weight complete_landform2: Goal -> Double;
factor: for Int i, Goal g
if action(i,_) & i >= 1
& (|Int j : (j <= i & action(j,"QuestComplete")
& argument(i,"Landforms2"))| >= 1)
add [goal(i,g)] * complete_landform2(g);
set collector.cutoff.complete_landform2 = 10;

/* Stage 3: Navigation Quests */

weight complete_navigation1: Goal -> Double;
factor: for Int i, Goal g

```

```

if action(i,_) & i >= 1
& (|Int j : (j <= i & action(j,"QuestComplete")
& argument(i,"Navigation1"))| >= 1)
add [goal(i,g)] * complete_navigation1(g);
set collector.cutoff.complete_navigation1 = 10;

factor: for Int i, Goal g
if action(i,_) & i >= 1
& (|Int j : (j <= i & action(j,"QuestComplete")
& argument(i,"Navigation2"))| >= 1)
add [goal(i,g)] * complete_navigation2(g);
set collector.cutoff.complete_navigation2 = 10;

```

C.2.2 Crystal Island: Uncharted Discovery with Milestone States and Discovery Events

```

/**** Hard Formulae ****/
/* (1) */
factor[2]: for Int i, Action a
if action(i, _) & i >= 0:
|Goal g: goal(i, g)| >= 1;

factor[3]: for Int i, Action a
if action(i, _) & i >= 0:

```

```

|Goal g: goal(i, g)| <= 1;

/**** Soft Formulae ****/

/* (2) */
weight prior: Goal -> Double;
factor: for Int i, Goal g
if action(i, _) & i >= 0
add [goal(i, g)] * prior(g);

/* (3) */
weight cur_action: Action x Goal -> Double;
factor: for Int i, Action a, Goal g
if action(i, a) & i >= 0
add [goal(i, g)] * cur_action(a, g);
set collector.cutoff.cur_action = 10;

/* (4) */
weight cur_loc: Loc x Goal -> Double;
factor: for Int i, Loc l, Goal g
if location(i, l) & i >= 0
add [goal(i, g)] * cur_loc(l, g);
set collector.cutoff.cur_loc = 10;

/* Stage 1: Completing the Tutorial */
weight complete_tutorial: Goal -> Double;

```

```

factor: for Int i, Goal g
if action(i,_) & i >= 1
& (|Int j : (j <= i & action(j,"TutorialCompleted"))| >= 1)
add [goal(i,g)] * complete_tutorial(g);
set collector.cutoff.complete_tutorial = 10;

/* Stage 2: Landform Quests */
weight complete_landform1: Goal -> Double;
factor: for Int i, Goal g
if action(i,_) & i >= 1
& (|Int j : (j <= i & action(j,"QuestComplete"))
& argument(i,"Landforms1"))| >= 1)
add [goal(i,g)] * complete_landform1(g);
set collector.cutoff.complete_landform1 = 10;

weight complete_landform2: Goal -> Double;
factor: for Int i, Goal g
if action(i,_) & i >= 1
& (|Int j : (j <= i & action(j,"QuestComplete"))
& argument(i,"Landforms2"))| >= 1)
add [goal(i,g)] * complete_landform2(g);
set collector.cutoff.complete_landform2 = 10;

/* Stage 3: Navigation Quests */
weight complete_navigation1: Goal -> Double;

```

```

factor: for Int i, Goal g
if action(i,_) & i >= 1
& (|Int j : (j <= i & action(j,"QuestComplete"))
& argument(i,"Navigation1"))| >= 1)
add [goal(i,g)] * complete_navigation1(g);
set collector.cutoff.complete_navigation1 = 10;

weight complete_navigation2: Goal -> Double;
factor: for Int i, Goal g
if action(i,_) & i >= 1
& (|Int j : (j <= i & action(j,"QuestComplete"))
& argument(i,"Navigation2"))| >= 1)
add [goal(i,g)] * complete_navigation2(g);
set collector.cutoff.complete_navigation2 = 10;

/* Additional Discovery Events */

/* Picked up flags */
weight pickup_flags: Goal -> Double;
factor: for Int i, Goal g
if action(i,_) & i >= 1
& (|Int j: (j <= i & action(j,"FlagManagerPickupFlag"))| >= 1)
add [goal(i,g)] * pickup_flags(g);
set collector.cutoff.pickup_flags = 10;

```

```

/* Drop flags */
weight drop_flags: Goal -> Double-;
factor: for Int i, Goal g
if action(i,_) & i >= 1
& (|Int j: (j <= i & action(j,"FlagManagerDropFlag"))| >= 1)
add [goal(i,g)] * drop_flags(g);
set collector.cutoff.drop_flags = 10;

/* Take Notes */
weight notes: Goal -> Double;
factor: for Int i, Goal g
if action(i,_) & i >= 1
& (|Int j: (j <= i & action(j,"PhotoJournalApp")
& argument(i,"AddNewNote"))|>=1)
add [goal(i,g)] * notes(g);
set collector.cutoff.notes = 10;

weight update_notes: Goal -> Double;
factor: for Int i, Goal g
if action(i,_) & i >= 1
& (|Int j: (j <= i & action(j,"PhotoJournalApp")
& argument(i,"NoteUpdated"))|>=1)
add [goal(i,g)] * update_notes(g);
set collector.cutoff.update_notes = 10;

```

```

/* Pickup a Sign */
weight pickup_sign: Goal -> Double;
factor: for Int i, Goal g
if action(i,_) & i >= 1
& (|Int j: (j <= i & action(j,"SignManagerPickupSign"))|>= 1)
add [goal(i,g)] * pickup_sign(g);
set collector.cutoff.pickup_sign = 10;

```

```

/* Take a photo */
weight photo: Goal -> Double;
factor: for Int i, Goal g
if action(i,_) & i >= 1
& (|Int j: (j <= i & action(j,"CameraApp"))| >= 1)
add [goal(i,g)] * photo(g);
set collector.cutoff.photo = 10;

```

```

/* Use IslandPedia */
weight reference: Goal -> Double;
factor: for Int i, Goal g
if action(i,_) & i >= 1
& (|Int j : (j <= i & action(j,"ReferenceApp"))| >= 1)
add [goal(i,g)] * reference(g);
set collector.cutoff.reference = 10;

```

```

/* Start Quests */

```

```

weight start_navigation1: Goal -> Double;
factor: for Int i, Goal g
if action(i,_) & i >= 1
& (|Int j: (j <= i & action(j,"QuestBegin")
& argument(j,"Navigation1"))|>= 1)
add [goal(i,g)] * start_navigation1(g);
set collector.cutoff.start_navigation1 = 10;

```

```

weight start_navigation2: Goal -> Double;
factor: for Int i, Goal g
if action(i,_) & i >= 1
& (|Int j: (j <= i & action(j,"QuestBegin")
& argument(j,"Navigation2"))|>= 1)
add [goal(i,g)] * start_navigation2(g);
set collector.cutoff.start_navigation2 = 10;

```

```

weight start_landforms1: Goal -> Double;
factor: for Int i, Goal g
if action(i,_) & i >= 1
& (|Int j: (j <= i & action(j,"QuestBegin")
& argument(j,"Landforms1"))|>= 1)
add [goal(i,g)] * start_landforms1(g);
set collector.cutoff.start_landforms1 = 10;

```

```

weight start_landforms2: Goal -> Double;

```

```

factor: for Int i, Goal g
if action(i,_) & i >= 1
& (!Int j: (j <= i & action(j,"QuestBegin")
& argument(j,"Landforms2"))|>= 1)
add [goal(i,g)] * start_landforms2(g);
set collector.cutoff.start_landforms2 = 10;

```

C.3 Schema Experiment

C.3.1 Data Definitions

```

/*
MLN Definition for our Plan Reco work, which includes
1. Predicates (observed, hidden, and global)
2. Formulae
3. Declaration of observed, hidden, global for the predicates
*/

/* 1. Define predicates */
// (1) Observed predicates
predicate userId: Id;
predicate action: Int x Action;
predicate loc: Int x Loc;
predicate state: Int x State;

```

```

predicate arg:  Int x ArgType x ArgValue;
predicate prevschema:  Int x Schema;
predicate prevarg:  Int x ArgValue;

// (2) Hidden predicates
predicate goal:  Int x Goal;

include "model.pml";

/* 3. Define which predicates are hidden, observed and global. */
observed:  userId, action, loc, state,prevschema,arg,prevarg;
hidden:  goal;

```

C.3.2 Schema Model

```

/** Hard Formulae - 41 minutes for 10 folds */
/* (1) */
factor[2]:  for Int i, Action a
if action(i, _) & i >= 0:
|Goal g:  goal(i, g)| >= 1;

factor[3]:  for Int i, Action a
if action(i, _) & i >= 0:
|Goal g:  goal(i, g)| <= 1;

```

```

/**** Soft Formulae ****/
/* (2) */
weight prior:  Goal -> Double;
factor:  for Int i, Goal g
if action(i,_) & i > 0
add [goal(i,g)] * prior(g);
set collector.cutoff.prior = 10;

weight prev_schema:  Schema x Goal -> Double;
factor:  for Int i, Schema s, Goal g
if prevschema(i,s) & i > 0
add [goal(i,g)] * prev_schema(s,g);
set collector.cutoff.prev_schema = 10;

weight cur_action:  Action x Goal -> Double;
factor:  for Int i, Action a, Goal g
if action(i,a) & i > 0
add [goal(i,g)] * cur_action(a,g);
set collector.cutoff.cur_action = 10;

weight cur_location:  Loc x Goal -> Double;
factor:  for Int i, Loc l, Goal g
if loc(i,l) & i > 0
add [goal(i,g)] * cur_location(l,g);
set collector.cutoff.cur_loctaion = 10;

```

```
weight cur_state: State x Goal -> Double;
factor: for Int i, State s, Goal g
if state(i,s) & i > 0
add [goal(i,g)] * cur_state(s,g);
set collector.cutoff.cur_state = 10;
```