Inferring intentions through state representations in cooperative human-robot environments

Craig Schlenoff

To cite this version:

Craig Schlenoff. Inferring intentions through state representations in cooperative human-robot environments. Other [cs.OH]. Université de Bourgogne, 2014. English. NNT : 2014DIJOS064 . tel-01172004

HAL Id: tel-01172004
https://theses.hal.science/tel-01172004
Submitted on 6 Jul 2015

UNIVERSITY OF BURGUNDY
U.F.R. SCIENCES ET TECHNIQUES

THESIS

presented by

Craig SCHLENOFF

to obtain the degree of

DOCTOR OF THE UNIVERSITY

INFERRING INTENTIONS THROUGH STATE REPRESENTATIONS IN COOPERATIVE HUMAN-ROBOT ENVIRONMENTS

Présentée le 30 juin 2014, devant le

Jury:

Prof. Benoît Eynard, Université de Technologie de Compiègne, Rapporteur

Prof. Bernard Grabot, Ecole Nationale D'Ingenieurs de Tarbes, Rapporteur

Prof. Salima Hassas, Université Claude Bernard Lyon 1, Examinatrice

Prof. Dominique Michelucci, Université de Bourgogne, Examinateur

Prof. Sebti Foufou, Université de Bourgogne, Directeur de Thèse

Abstract

Humans and robots working safely and seamlessly together in a cooperative environment is one of the future goals of the robotics community. When humans and robots can work together in the same space, a whole class of tasks becomes amenable to automation, ranging from collaborative assembly to parts and material handling to delivery. Proposed standards exist for collaborative human-robot safety, but they focus on limiting the approach distances and contact forces between the human and the robot. These standards focus on reactive processes based only on current sensor readings. They do not consider future states or task-relevant information. A key enabler for human-robot safety in cooperative environments involves the field of intention recognition, in which the robot attempts to understand the intention of an agent (the human) by recognizing some or all of their actions to help predict the human's future actions.

We present an approach to inferring the intention of an agent in the environment via the recognition and representation of state information. This approach to intention recognition is different than many ontology-based intention recognition approaches in the literature as they primarily focus on activity (as opposed to state) recognition and then use a form of abduction to provide explanations for observations. We infer detailed state relationships using observations based on Region Connection Calculus 8 (RCC-8) and then infer the overall state relationships that are true at a given time. Once a sequence of state relationships has been determined, we use a Bayesian approach to associate those states with likely overall intentions to determine the next possible action (and associated state) that is likely to occur.

We compare the output of the Intention Recognition Algorithm to those of an experiment involving human subjects attempting to recognize the same intentions in a manufacturing kitting domain. The results show that the Intention Recognition Algorithm, in almost every case, performed as good, if not better, than a human performing the same activity.

Résumé

Les humains et les robots travaillant en toute sécurité et en parfaite harmonie dans un environnement est l'un des objectifs futurs de la communauté robotique. Quand les humains et les robots peuvent travailler ensemble dans le même espace, toute une catégorie de tâches devient prête à l'automatisation, allant de la collaboration pour l'assemblage de pièces, à la manutention de pièces et de materiels ainsi qu'à leur livraison. Garantir la sûreté des humains nécessite que le robot puisse être capable de surveiller la zone de travail, déduire l'intention humaine, et être conscient suffisamment tôt des dangers potentiels afin de les éviter.

Des normes existent sur la collaboration entre robots et humains, cependant elles se focalisent à limiter les distances d'approche et les forces de contact entre l'humain et le robot. Ces approches s'appuient sur des processus qui se basent uniquement sur la lecture des capteurs, et ne tiennent pas compte des états futurs ou des informations sur les tâches en question. Un outil clé pour la sécurité entre des robots et des humains travaillant dans un environnement inclut la reconnaissance de l'intention dans lequel le robot tente de comprendre l'intention d'un agent (l'humain) en reconnaissant tout ou partie des actions de l'agent pour l'aider à prévoir les actions futures de cet agent. La connaissance de ces actions futures permettra au robot de planifier sa contribution aux tâches que l'humain doit exécuter ou au minimum, à ne pas se mettre dans une position dangereuse.

Dans cette thèse, nous présentons une approche qui est capable de déduire l'intention d'un agent grâce à la reconnaissance et à la représentation des informations de l'état. Cette approche est différente des nombreuses approches présentes dans la littérature qui se concentrent principalement sur la reconnaissance de l'activité (par opposition à la reconnaissance de l'état) et qui « devinent » des raisons pour expliquer les observations. Nous déduisons les relations détaillées de l'état à partir d'observations en utilisant Region Connection Calculus 8 (RCC-8) et ensuite nous déduisons les relations globales de l'état qui sont vraies à un moment donné. L'utilisation des informations sur l'état sert à apporter une contribution plus précise aux algorithmes de reconnaissance de l'intention et à générer des résultats qui sont equivalents, et dans certains cas, meilleurs qu'un être humain qui a accès aux mêmes informations.

Acknowledgments

There are many people that I would like to thank for their support and encouragement that has culminated in the publication of this thesis.

First and foremost, I would like to thank my wife, Rani. She has been incredibly supportive of my academic pursuits, and has spent many extra hours taking care of our family while I was hunched over a computer analyzing the experiments or writing up the thesis. Rani, thank you and I love you very much. I would also like to thank my children, Jake, Alec, and Tali, for giving me the time (and the quiet environment) to get through my research. Now that my work is complete, I can start getting through the to-do list of all of the places you want me to take you.

I would also like to thank my advisor, Professor Sebti Foufou. For many years, I have been contemplating going back for my PhD and it was conversations with him that gave me the push to do so. Having been over 15 years since my last degree, it took some adjustment to get back into the grind of being a student, and I thank him for helping me with the transition. I also owe him many thanks for guiding me through this process, offering his help and support during the research challenges, and helping with the language barrier between a French university and a non-French speaking student.

I would like to thank my thesis committee, made up of Prof. Benoit Eynard (rapparteur), Prof. Bernard Grabot (rapparteur), Prof. Salima Hassas (examinatrice), and Prof. Dominique Michelucci (examinateur) for serving on my committee and for the the excellent comments about my thesis.

In addition, I would like to thank my employer, the National Institute of Standards and Technology (NIST) for allowing me to leverage the strong work that NIST performs.

One of the reasons I was able to perform the significant amount of research described in this thesis is because of the emphasis that was placed on education as I was growing up and because of the encouragement I had as a child. My parents, Marvin and Harriet, created this very enriching environment and gave me a lot of love throughout the process. For all of this and much more, I am forever thankful.

Lastly, I would like to thank Zeid Kootbally for helping me understand the PhD process of the University of Burgundy (having completed it a few years earlier) and for helping me interpret and complete the numerous French forms that without him I would have had a very tough time understanding.

Table of Contents

Abstract 3
Résumé 4
Acknowledgments 5
Table of Contents 6
List of Tables 9
List of Figures 10
1. Introduction 13
1.1. Objective and Thesis Contribution 14
1.2. Organization of Thesis 17
2. Related Work 21
2.1. Intention Recognition 21
2.1.1. Overview 21
2.1.2. Classification of Intention Recognition 23
2.1.3. Belief-Desire-Intention Architectures 33
2.1.4. Applications Domains 35
2.1.5. Metrics for Intention Recognition 41
2.2. Activity Recognition vs. State Recognition 43
2.2.1. State of the Art in State Recognition 44
2.2.2. State of the Art in Activity Recognition 47
2.2.3. Comparison of State and Activity Recognition 50
2.3. State Representation 50
2.3.1. States and Spatial Knowledge in High-Level Ontologies 50
2.3.2. Approaches to Spaces Representation Within Detailed Ontologies 52
2.3.3. Different Dimensions of State (Spatial) Representation 54
2.3.4. Uses of Spatial Language in Various Applications 56
2.4. Conclusion 57
3. State Representation and Reasoning 60
3.1. The State Relation Ontology 60
3.1.1. RCC8 Approach 61
3.1.2. Defining More Complex Relations 63
3.1.3. State Relation Ontology Constructs 65
3.2. The State Relation Algorithms 66

3.2.1. Projecting the Objects onto the Planes and Determining the Contour ... 66
3.2.2. Determining When Object Projections Overlap ... 67
3.2.3. Inferring Intermediate State Relations from RCC8 Relations ... 68
3.3. The Manufacturing Kitting Ontology and Associated State Relations ... 70
3.3.1. Human-Robot Collaboration Manufacturing Scenarios ... 70
3.3.2. Manufacturing Kitting Domain Description ... 72
3.3.3. The Manufacturing Kitting Ontology ... 72
3.4. Conclusion ... 87
4. Intention Recognition ... 88
4.1. The Intention Ontology ... 88
4.1.1. Ordering Constructs ... 88
4.1.2. Intentions ... 90
4.1.3. A Kitting Example ... 91
4.1.4. Kitting Example Represented in the Intention Ontology ... 92
4.2. The Intention Recognition Algorithm ... 95
4.2.1. Tracking States and State Relations as Actions Occur ... 95
4.2.2. Matching Perceived Real-Time State Relations to Intention Templates ... 98
4.2.3. Associating Likelihoods to Intentions ... 100
4.2.4. Individual Intention Recognition Approaches Explored ... 101
5. System Architecture ... 119
5.1. USARSim ... 120
5.2. USARTruth ... 122
5.3. MySQL Database ... 123
5.3.1. MySQL Database Generation ... 124
5.3.2. C++ Class Generation and Usage ... 125
5.3.3. C++ Classes to Access Data from the MySQL Database ... 126
5.4. State Relation and Kitting Ontology ... 127
5.5. State Recognition Algorithms ... 129
5.6. State Recognition Output ... 131
5.7. Intention Ontology ... 133
5.8. Intention Recognition Algorithm ... 134
5.8.1. Loading and Reading an Ontology ... 134
5.8.2. Methodology to Compute Likelihoods ... 135
5.9. Intention Likelihoods ... 137
6. Intention Recognition Experiment ... 138
6.1. Experimental Procedures ... 138

6.2. Experimental Results (Overview) 142
6.3. Experimental Results (Details) 144
6.4. Experimental Results (Overall Observations) 171
6.5. Applying Addition Intention Recognition Approaches 172
7. Conclusion and Future Work 177
7.1. Achievement 177
7.2. Future Work 178
Bibliography 181
Appendix A: Kitting Workstation Ontology Details 192
Appendix B: Kit Plans for Experiment 210

List of Tables

Table 1: Actions Used for Activity Recognition 48
Table 2: Kitting Workstation Class Hierarchy 75
Table 3: Preconditions and Effects for the Action *take-kittray* 83
Table 4: Revised Definitions of Spatial Relationships 85
Table 5: Initial State Representation Ordering Constructs. 89
Table 6: Construct Details 90
Table 7: Kit 1 Intention Template 94
Table 8: Kit 2 Intention Template 95
Table 9: Real-Time State Relation Tracking Table 97
Table 10: Real-Time State Relation Tracking Table (Version 2) 99
Table 11: Kit 1 Intention Template (Version 2) 99
Table 12: Kit 2 Intention Template (Version 2) 100
Table 13: Intentions vs. Matching States Relations 102
Table 14: Summary of Intention Variables at State 8 103
Table 15: The Effect of Metrics on Likelihood 115
Table 16: Intentions vs. Matching States Relations 116
Table 17: Detailed Human Intention Recognition Data 140
Table 18. Comparison of Algorithm Output to Human Intention Recognition 144
Table 19: Impact of Weights on First Three Approaches - Coarse Level 173
Table 20: Impact of Weights on First Three Approaches: Fine-Grained Level 174

List of Figures

Figure 1: Six Possible Paths for Intention Recognition 14
Figure 2: Novel Path for Intention Recognition 16
Figure 3: Intention Recognition Process Flow 17
Figure 4: Abduction vs. Deduction vs. Induction 25
Figure 5: Sample Bayesian Belief Network 29
Figure 6: Case-Based Reasoning Approach [52] 31
Figure 7: An Intention Graph [13] 33
Figure 8: High-Level DBI Architecture 34
Figure 9: Graph-Based Approach to Intention Recognition [24] 40
Figure 10: Sample Machined NIST Artifacts Used in the 2011 SPC, Arranged Randomly. 45
Figure 11: The Ground Truth Fixture 45
Figure 12: Translation Scores Over All 399 Frames 46
Figure 13: Precision Results from Mind's Eye Evaluation 49
Figure 14: Results of MCC Metric 49
Figure 15: Sample Cyc Spatial Relations 51
Figure 16: Qualitative Entity Location [126] 55
Figure 17: RCC8 Relations (Credit: http://en.wikipedia.org/wiki/RCC8) 61
Figure 18: State Relation Ontology 66
Figure 19: Projection Objects onto Places and Determining Overlap 67
Figure 20: Pseudcode for Computing Overlaps 68
Figure 21: Predicate Evaluation Pseudo-code 69
Figure 22: Joint Human-Robot Assembly 71
Figure 23: Example Kit (courtesy of LittleMachineShop.com) 72
Figure 24: Kitting Workstation Model 77
Figure 25: End Effector Changing Station Model 77
Figure 26: Large BoxWith Empty Kit Trays Model 78
Figure 27: Large Box With Kits Model 78
Figure 28: Kit Model 79
Figure 29: Robot Model 79
Figure 30: Stock Keeping Unit Model 80
Figure 31: Work Table Model 80
Figure 32: Vacuum Effector Single Cup Model 80
Figure 33: Kit Design Model 81
Figure 34: Part Model 81
Figure 35: Pose Location Model 81

Figure 36: The Intention Ontology 91
Figure 37: Sample Kitting Scenario 92
Figure 38: Completed Kits 1 and 2 92
Figure 39: Diagram Explaining Overall Likelihood Equation 101
Figure 40: Kitting Intention Tool Interface 117
Figure 41: Output of the Intention Recognition Tool 118
Figure 42: Experimentation Architecture 120
Figure 43: USARSim Kitting Environment 120
Figure 44: USARTruth Channel 122
Figure 45: OWL to SQL Tab 125
Figure 46: Example Using the Generated C++ Classes 127
Figure 47: State Relation and Kitting Ontologies 127
Figure 48: Five Possible Kits (Intentions) 128
Figure 49: State Recognition Algorithms 129
Figure 50: Java Code for Evaluating Externally Connected RCC8 Relation 130
Figure 51: State Relation Output 131
Figure 52: Results of State Recognition Algorithms 131
Figure 53: Intention Ontology 133
Figure 54: Intention Recognition Algorithm 134
Figure 55: Methodology to Compute and Display Likelihoods 136
Figure 56: Intention Likelihoods 137
Figure 57: Likelihoods for Each Intention at Different States 137
Figure 58. Experiment User Interface 139
Figure 59: Human/Algorithm Comparison Data for Kit 1 Plan 1 146
Figure 60: Human/Algorithm Comparison Data for Kit 1 Plan 2 147
Figure 61: Human/Algorithm Comparison Data for Kit 1 Plan 3 148
Figure 62: Human/Algorithm Comparison Data for Kit 1 Plan 4 149
Figure 63: Human/Algorithm Comparison Data for Kit 1 Plan 5 150
Figure 64: Human/Algorithm Comparison Data for Kit 2 Plan 1 151
Figure 65: Human/Algorithm Comparison Data for Kit 2 Plan 2 152
Figure 66: Human/Algorithm Comparison Data for Kit 2 Plan 3 153
Figure 67: Human/Algorithm Comparison Data for Kit 2 Plan 4 154
Figure 68: Human/Algorithm Comparison Data for Kit 2 Plan 5 155
Figure 69: Human/Algorithm Comparison Data for Kit 3 Plan 1 156
Figure 70: Human/Algorithm Comparison Data for Kit 3 Plan 2 157
Figure 71: Human/Algorithm Comparison Data for Kit 3 Plan 3 158
Figure 72: Human/Algorithm Comparison Data for Kit 3 Plan 4 159

Figure 73: Human/Algorithm Comparison Data for Kit 3 Plan 5...160
Figure 74: Human/Algorithm Comparison Data for Kit 4 Plan 1...161
Figure 75: Human/Algorithm Comparison Data for Kit 4 Plan 2...162
Figure 76: Human/Algorithm Comparison Data for Kit 4 Plan 3...163
Figure 77: Human/Algorithm Comparison Data for Kit 4 Plan 4...164
Figure 78: Human/Algorithm Comparison Data for Kit 4 Plan 5...165
Figure 79: Human/Algorithm Comparison Data for Kit 5 Plan 1...166
Figure 80: Human/Algorithm Comparison Data for Kit 5 Plan 2...167
Figure 81: Human/Algorithm Comparison Data for Kit 5 Plan 3...168
Figure 82: Human/Algorithm Comparison Data for Kit 5 Plan 4...169
Figure 83: Human/Algorithm Comparison Data for Kit 5 Plan 5...170
Figure 84: Individual Approaches Weights Effect on Sum of Difference..174
Figure 85: Comparison of Human Bayesian and Additional Approaches for Kit 2, Plan 4 175

1. Introduction

Humans and robots working safely and seamlessly together in a cooperative environment is one of the future goals of the robotics community. When humans and robots can work together in the same space, a whole class of tasks becomes amenable to automation, ranging from collaborative assembly to parts and material handling and delivery. Keeping humans safe requires the ability of the robot to monitor the work area, infer human intention, and be aware of potential dangers soon enough to avoid them. Robots are under development throughout the world that will revolutionize manufacturing by allowing humans and robots to operate in close proximity while performing a variety of tasks [1].

Proposed standards exist for collaborative human-robot safety, but these focus on limiting approach distances and contact forces between the human and the robot [2, 3]. In essence, the robot attempts to minimize the likelihood and potential severity of collisions between the human and the robot. These approaches focus on reactive processes based only on current sensor readings, and do not consider future states or task-relevant information.

A key enabler for human-robot safety in cooperative environments involves the field of intention recognition, in which the robot attempts to understand the intention of an agent (the human) by recognizing some or all of their actions [4] to help predict the human's future actions. With the addition of reliable intention recognition, one might imagine the following:

A robot is placed in a manufacturing kitting environment with the goal of helping the humans in the factory complete their tasks. Orders change very frequently in the factory, and the humans often switch between activities to make sure that all orders are met. The robot is tasked with making sure that all necessary parts are available to the humans and that the kits and assemblies are produced correctly. This must all occur while insuring safe interaction with the humans. In addition, the robot may be asked to finish the creation of a kit when the human is pulled to another task.

There are many robot functionalities needed to allow this scenario to occur, including perception, manipulation, mobility, etc. In this thesis, we will focus on the ability for the robot to recognize the intention of the human. By recognizing the human's intention, the robot will be able to determine what overall goal the human is trying to accomplish (i.e. what kit is desired to be developed). Once the intention is determined, the robot can: 1) retrieve any additional parts necessary for the remainder of the kit-building process, but not immediately available to the human; 2) ensure that the reminder of the kit is developed to specification; and/or 3) determine where the human left off and complete the development of the kit when the human is pulled to another task. The robot may also ensure that it interacts with the human safely by accurately predicting which actions will occur next (based on the identification of the intention and its known subsequent steps) and ensuring that it stays away from the path of those actions.

1.1. Objective and Thesis Contribution

In [5], Heinze proposed six possible paths for intention recognition that start with the intention of the intending agent (in this case the human) and end with the recognition of the intention by the recognizing agent (in this case the robot). These possible paths are shown in Figure 1 and are described below:

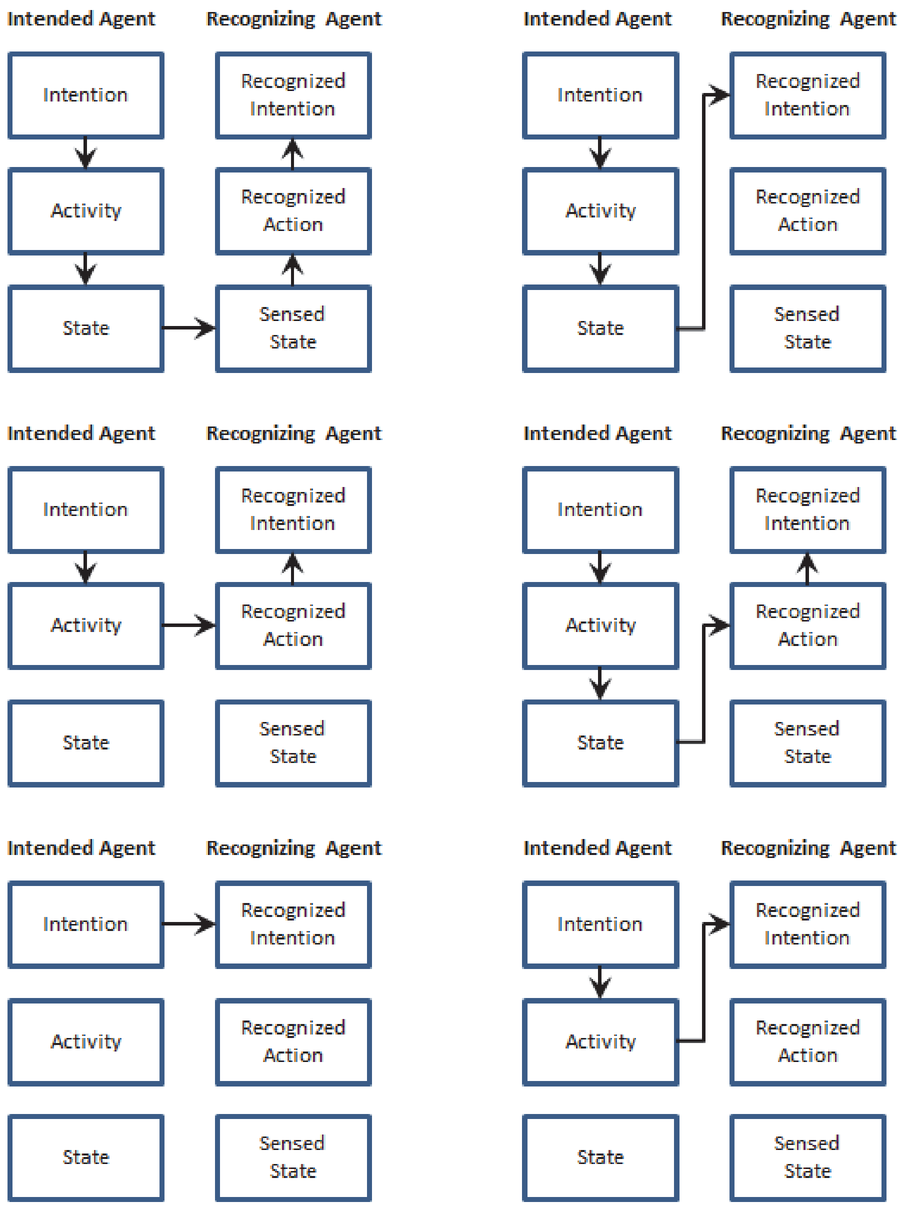


Figure 1: Six Possible Paths for Intention Recognition

- (Top Left) This case is most common in the literature. The intending agent executes his intentions, which is composed of actions, which causes states in the world to be true. The states are recognized by the recognizing agent which then infers the activities and finally infers the intention;

- (Middle Left) This case requires that the intending agent provides direct access to (and descriptions of) the actions that it is performing to the recognizing agent. The recognizing agent then uses these activity descriptions to infer the intending agent's intention. This requires that the intending agent is part of the system design process as its internal state must be made available to the recognizing agent;

- (Bottom left) This case requires that the intention of the intending agent is made directly available to the recognizing agent, so no reasoning is needed at all. This can be done through direct communication between the intending agent and the recognizing agent. In a highly dynamic environment, this is not always possible (or desired). It is envisioned that manufacturing facilities that use robots in the future will be highly dynamic and agile, with a large variety of orders that change quickly. We don't want the human to have to change his/her procedures to accommodate the robot; we want the robot to be able to figure this out on its own;

- (Top Right, Middle Right, and Bottom Right) These three cases all focus on the concept of "direct perception," where perception not only involves the recognition of objects and actions, but also incorporates the inference that is associated with them. In the top right case, the recognized intention is provided directly from the perception of the state which innately infers the intention. Likewise in the middle right case, the perception system innately infers which actions are being performed based on the perceived state of the environment. Inference then occurs to determine the intention from the recognized actions. In the bottom right case, the recognized intention is provided directly from the perception of the activities which innately infers the intention.

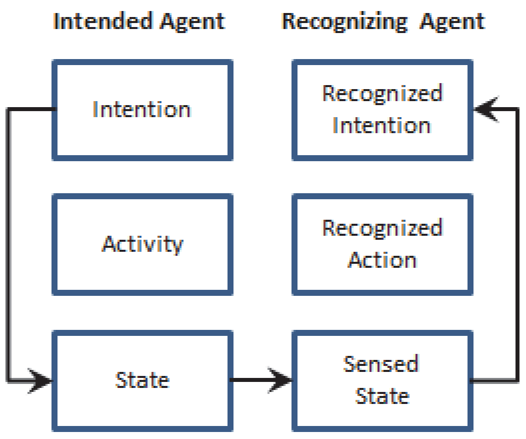


Figure 2: Novel Path for Intention Recognition

In this thesis, we propose a new structure for recognizing intentions, as shown in Figure 2. In this approach, we focus on the relationship between states and intentions, while bypassing the activity level all together. Direct state-based intention recognition offers some interesting advantages over activity-based recognition, including:

- States are often more easily recognizable by sensor systems than actions. As shown in [6] and discussed in the next chapter, existing algorithms that attempt to recognize the state of the environment (the identification, location, and orientation of objects) perform approximately 20% better than algorithms that attempt to identify activities that are being performed in the environment. As such, state information can be provided with higher accuracy as input into an Intention Recognition Algorithm, which in turn would allow for better algorithm output;

- Using activities, intention recognition is often limited to inferring the intention of a single person [4]. State-based intention recognition eliminates this shortfall, in that the existence of the state is independent of who created it. In the state-based approach multiple people can work on the same intention because the algorithms do not look at the activities that are being performed. They only focus on the state that is achieved once each activity is completed;

- State information tends to be more ubiquitous than activity information, thus allowing for reusability of the ontology. Spatial relations such as *On-Top-Of*, *Partially-In*, and *In-Contact-With* are generic across multiple domains, while “Place The Box in the Kit Tray” may only be specific to the manufacturing kitting domain.

We distinguish states from state relationships. In this context, a state is defined as a set of properties of one or more objects in an area of interest that consist of specific recognizable configuration(s) and or characteristic(s). A state relationship is a specific relation between two objects (e.g., Object 1 is on top of Object 2). A set of all relevant state relationships in an environment at a given time composes a state. This approach to intention recognition is different than many ontology-based intention recognition approaches in the literature (as described in the next chapter) as they primarily focus on activity (as opposed to state) recognition and then use a form of abduction to provide explanations for observations. We infer detailed state relationships using observations based on Region Connection Calculus 8 (RCC-8) [7] and then infer the overall state relationships that are true at a given time. Once a sequence of state relationships has been determined, we use a Bayesian approach to associate those states with likely overall intentions to determine the next possible action (and associated state) that is likely to occur.

1.2. Organization of Thesis

The overall flow of information that is presented in this thesis is shown in Figure 3. State change information if sent from the USARSim simulation engine (which is meant to simulate human actions) into a state relation algorithm. This algorithm analyzes the state changes and infers a set of state relations that are true within the environment (leveraging information that is present in the manufacturing kitting ontology and the state relation ontology). The resulting state relations are fed into an intention recognition algorithm which assigns likelihoods to possible intentions (leveraging information in an intention ontology). The areas that are shaded in blue represent the unique contribution of this work.

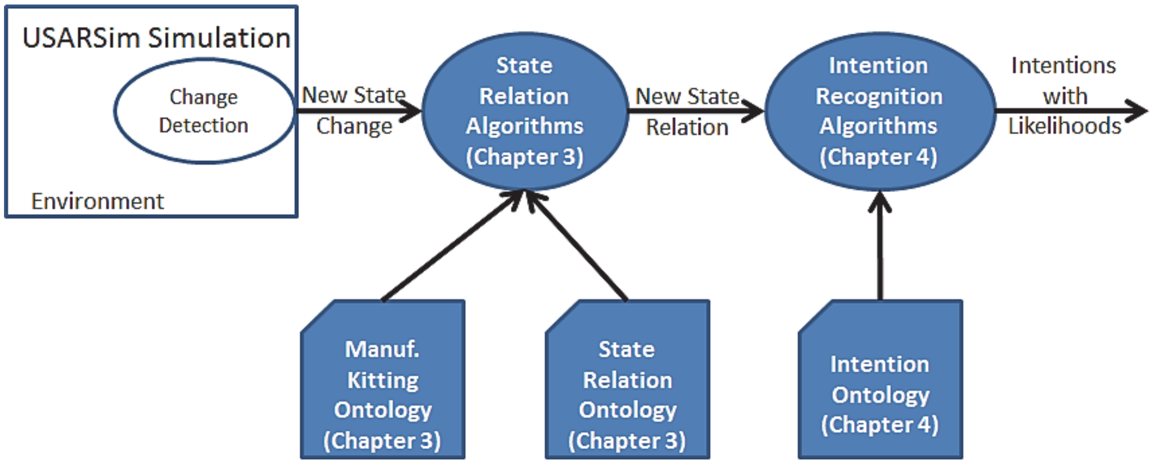

USARSim Simulation
Change Detection
Environment
New State Change
State Relation Algorithms (Chapter 3)
New State Relation
Intention Recognition Algorithms (Chapter 4)
Intentions with Likelihoods
Manuf. Kitting Ontology (Chapter 3)
State Relation Ontology (Chapter 3)
Intention Ontology (Chapter 4)

Figure 3: Intention Recognition Process Flow

Chapter 2 presents the state-of-the-art in the areas of intention recognition, state/activity recognition, and state representation. All three of these areas play a vital role in the novel research described in this thesis. We categorize the intention recognition field into logic-based approaches (which include deductive and abductive techniques), probabilistic approaches, and case-based approaches. Deductive approaches give definitive and provable results, but it is very rare in intention recognition that enough information (observations) is provided to be able to only use this approach. Abductive approaches provide reasonable explanations when the intention is not deductive, but mechanisms are needed to choose between the various possible explanations. Probabilistic approaches use well-tested theories to assign likelihoods to various intentions, but determining the input probabilities or belief to use these approaches can be more of an art than a science. Case-based approaches leverage previous instances of the same or similar intentions to try to match to what is being observed in the environment, but this starts to fall apart if there are no previous cases that match the current observations.

We then discuss the state of the art in activity and state recognition. A key component of this thesis is the fact that we use state information as the basis to recognition intentions as opposed to using activity information (as is predominantly performed in the literature). We examine two evaluation efforts, the Solutions in Perception Challenge and DARPA's Mind's Eye program, to show representative cases of the state-of-the-art in state and activity recognition, respectively. These two evaluations were purposely chosen because they are completely unbiased: an independent evaluation (not run by the algorithm developers) was performed to assess the performance of developed systems. Based on this analysis of these evaluations, it appears that state recognition systems are approximately 20% (80% vs. 60%) more accurate than the activity recognition systems. This shows the benefit of basing intention recognition on state information as opposed to activity information.

We conclude the chapter by showing the various ways that state and spatial information have been represented in ontologies. This ranges from very high-level abstract representations in foundational ontologies to more specific representations in domain-specific ontologies. We also make the case for using qualitative methods to capture a perspective of space which is more in line with human intuition. We look at various qualitative representation approaches, with an emphasis on RCC8, which serves as the basis for much of the work described in this thesis.

Chapter 3 describes an approach that uses RCC8 to model state relationships based on the relative positions of objects in the environment. We describe how we extend RCC8, which was initially developed for a two-dimensional space, to a three-dimensional space by applying it along all three planes (xy-plane, yz-plane, xz-plane). We then describe a set of high-level state relationships that were developed based on a set of logical rules that are grounded in the RCC8 relations. These RCC8 relations and corresponding high-level state relationships easily allow a sensor system to characterize the state of the environment, and these characterizations will serve as the basis for the Intention Recognition Algorithm. We then discuss how these state relationships can be represented in an ontology. We conclude this chapter by validating this approach using example scenarios from the manufacturing kitting domain, along with an associated manufacturing kitting ontology.

Chapter 4 describes how intentions are formed from states, and how these intentions are represented in an ontology. An ordering of state relationships represents an intention. We describe how we leverage OWL-S (Web Ontology Language – Services) [8] to form the ordering of the states. We then provide a kitting example to show examples of intentions based on states. Once intentions are defined, we describe a Bayesian-based approach to associate observations in the environment to known intentions.

We then discuss additional Intention Recognition Algorithm based on the following criteria: 1) the number of observed state relations that are true in an intention; 2) the percentage of an intention that is complete; and 3) the number of productive states that have occurred recently; These additional approaches will help to address situations (in the future) when an intention is not perceived from the very beginning, or when a new intention starts mid-way through the set of observations. We then propose an overall equation that can be used to bring all of these approaches together and assign weights to each.

Chapter 5 describes the overall architecture of how this approach was implemented. We describe the overall architecture that was used to recognize states in the environment and infer intentions. The architecture involves a Simulation System, a State Recognition Algorithm (with associated state relation and kitting ontologies), and an Intention Recognition Algorithm (with an associated intention ontology).

Chapter 6 focuses on an experiment used to assess the performance of the state-based Intention Recognition Algorithm described above. This was done by comparing the output of the algorithms to the performance of several humans watching agents performing the same intentions. We used the manufacturing kitting domain for the experiment. Five kits were created that each contained a unique combination of ten differently-shaped blocks. Using these kits, a set of simulations were developed capturing a set of steps in which one could assemble these kits. For this experiment, we randomly chose five set of steps for each kit, resulting in 25 total runs. At each state, the State Recognition Algorithms were run to identify the state relations in the environment and, based on these state relations, likelihoods were assigned to each intention. To determine how well these likelihoods compared to what a human would perceive in the same situation, we had 15 students serve as the human participants. One by one, the students were presented with a textual description of something that happened in the environment (a state), for example, “Red object added in the kit tray.” Based on this information, the students assigned likelihoods representing which kit they thought was being developed (i.e., what was the likely intention based on the observed states). Because there were exactly 10 objects in each kit, exactly ten states were presented per kit. After all 10 states were completed, the student moved on to the next intention until all 25 intentions were completed. The results of this experiment were compared to the output of the Intention Recognition Algorithm, and comparison charts are presented near the end of this chapter. We then introduce the additional three intention recognition approaches to see how they affect the performance of the overall system.

Chapter 7 is devoted to the conclusions of the thesis. We describe future efforts which include applying the approaches described in this thesis to areas outside of the manufacturing kitting domain, transitioning the algorithms to a real manufacturing environment, and performing additional experiments to further determine the optimal weights of the proposed intention recognition approaches.

2. Related Work

In this chapter, state-of-the-art technologies in the areas of intention recognition, state/activity recognition, and state representation are presented. All three of these areas are core to this thesis and thus a thorough literature review is provided.

2.1. Intention Recognition

Intention recognition traditionally involves recognizing the intent of an agent by analyzing some, or all, of the actions that the agent performs. Many of the recognition efforts in the literature are composed of at least three components: 1) identification and representation of a set of intentions that are relevant to the domain of interest; 2) representation of a set of actions that are expected to be performed in the domain of interest, and the association of these actions with the intentions; 3) recognition of a sequence of observed actions executed by the agent, and matching them to the actions in the representation. [4]

2.1.1. Overview

There have been many techniques in the literature applied to intention recognition that follow the three steps listed above, including ontology-based approaches [9], multiple probabilistic frameworks such as Hidden Markov Models [10], Dynamic Bayesian Networks [11], utility-based intention recognition [12], and graph-based intention recognition [13]. All of these approaches are described in more detail below. In this thesis, we will focus on ontology/logic-based approaches. An overview of how to assess ontologies, with a focus on intention recognition, can be found in [14].

In many of these efforts, abduction has been used to provide hypotheses about intentions. In abduction, the system "guesses" that a certain intention could be true based on the existence of a series of observed actions. For example, one could guess that someone may have watered the lawn if the lawn is wet. There may be other possible explanations for why the lawn is wet (e.g., it rained) but the fact that someone watered the lawn could be the most probable explanation given the circumstances. As more information is learned and activities are performed, probabilities of certain intentions can be refined to be consistent with the observations.

Intention recognition, and specifically the approach described in this thesis, builds upon research in the area of causal theory used for planning. In this approach, a background theory is provided, which contains intentions that are expected to be performed in the environment along with the relationships between the states and the intentions. There are also observations of the states that exist over time, along with the hypotheses (and probabilities) of the agent's intentions based on these states. Such an approach is conceptually related to causal theories used for planning [15], with the exception of states taking the place of activities.

As shown above, there are often many possible, equally-plausible hypotheses an intention of an observed agent's intentions or to describe the situation in an environment. Choosing which hypothesis to believe is one challenge. Another challenge is that the adversarial nature of the observed agent may limit what actions are able to be observed. In other words, the scene may be occluded by other objects so that it cannot be seen in its entirety, or the agent performing the action may purposefully hide their actions or intentions from the observer. Furthermore, agents may even deliberatively execute misleading actions to throw off the intention recognition system.

Another challenge are situations where the observed agent may be performing multiple intentions at the same time and may intersperse the execution of their actions, or the case where the agent is trying different plans for achieving a single intention. In addition, intention recognition becomes very challenging when attempting to reason about the actions of cognitively impaired individuals. They may be executing the actions erroneously and with confusion, as in the case of Alzheimer patients [16]. Additional complications arise when trying to analyze the actions of many cooperating agents [17]

Much research in the intention recognition field focuses on pruning the space of hypotheses. In a given domain, there could be many possible intentions. Based on the observed actions, various techniques have been used to eliminate improbable intentions and assign appropriate probabilities to intentions that are consistent with the actions performed. Some of the efforts have weighted conditional rules used for intention recognition as a function of the likelihood that those conditions are true [18]. For example, it may be unlikely that a person is using an umbrella outside unless it is raining or very sunny, thus the cost associated with this condition would be very high since the probability of this happening would be low.

Once observations of actions have been made, different approaches exist to match those observations to an overall intention or goal. For example, in [19], the authors use existentially quantified observations (not fully grounded observations) to match actions to plan libraries. In this instance, they can handle situations when they see some action occur (e.g., opening a door) or without seeing or knowing who performed that action. This is different than many other approaches in the literature that require fully grounded activities. Other approaches have focused on building plans with frequency information, to represent how often an activity occurs [20]. The rationale behind this approach is that there are some activities that occur very frequently and are often not relevant to the recognition process (e.g., a person cleaning their hands). When these activities occur, they can be mostly ignored and only activities that are less commonly performed can be considered. In [21], the authors combine probabilities and situation calculus-like formalization of actions. In particular, they not only define the actions and sequences of actions that constitute an intention, they also state which activities cannot occur for the intention to be valid. For example, if the intention was to drive a car, the activity may be to open the door, get into the car, turn on the engine, release the emergency brake, and take the car out of park. They may also include that an activity cannot be to turn the car off after it is turned on and before the car is taken out of park.

All of these approaches have focused on the activity being performed as the primary basis for observation and the building block for intention recognition. However, as noted in [4], activity recognition is a very hard problem and one that is far from being solved. There has been limited success in using Radio Frequency Identification (RFID) readers and tags attached to objects of interest to track their movement with the goal of associating their movement with known activities. Case in point, in [22], the authors describe the process of making tea as a three step process involving using a kettle, getting a box of tea bags, and adding some combination of milk, sugar or lemon. Each of these activities is identified by a user wearing a special set of gloves that can read RFID tags on objects of interest. However, this additional hardware can be inhibiting and unnatural. Recognizing and representing states as opposed to actions can help to address some of the issues involved in activity recognition and this will be the focus of the thesis.

The rest of this chapter describes 1) classification of intention recognition; 2) applications to which intention recognition has historically been applied; 3) approaches to activity and state recognition and their performance; and 4) approaches to state representation.

2.1.2. Classification of Intention Recognition

There are two main dimensions of intention recognition in the literature. The first deals with the interaction and cooperativeness of the agent that is being observed. The second is the mechanism by which the intention of the agent is determined. Each is discussed below.

2.1.2.1. Interaction and Cooperativeness of Agent Being Observed

In [23], the authors propose two classifications of intention recognition: intended or keyhole. In intended intention recognition, the observed agent allows his intentions to be identified and purposefully and openly provides signals to allow the observed agent to sense his actions. One example of this could be cooperative assembly, where the agent openly conveys his intentions. In keyhole intention recognition, the observed agent doesn’t expect his intentions to be identified; he is only focused on his own activities, which may limit how much an observing agent can view his activities. This is referred to as keyhole because it can be considered a situation where an observing agent is watching a performing agent through the keyhole of a door, where the performing agent cannot see the observing agent nor does the performing agent know that he is being observed. An example of this may be a help system that provides guidance, as in a home ambient intelligence system.

A third class, proposed by [24], is adversarial, in which the agent purposefully does not want his actions being observed, for example, in a war situation where the enemy is purposefully trying to deceive the observing agent. Almost all of the intention recognition work in the literature focuses on the intended and keyhole cases. Recent work in adversarial intention recognition can be found in [25].

2.1.2.2. Approaches To Perform Intention Recognition

In addition to classifying the types of intention recognition with respect to the agent that is performing the activities, one can also classify intention recognition by the approaches that are used to perform them. Intention recognition can roughly be broken down into three categories: logic-based approaches (using abduction and deduction); probabilistic approaches; and case-based approaches. Each of these are defined and described below.

2.1.2.2.1. Logic-Based Approaches

As the name implies, logic-based approaches use some form of logical reasoning to deduce the intentions of agents in the environment. These approaches can be broken down into abductive, deductive, and inductive approaches. Abduction [26] is a type of defensible reasoning, primarily used to explain observations. Abduction does not provide a provably correct answer; instead it provides one reasonable explanation for an observation, though there may be others.

Conversely, deduction is the process of reasoning from one or more general observations (premises) to reach a logically certain conclusion. In this case, the observations guarantee the truth of the conclusion.[1]

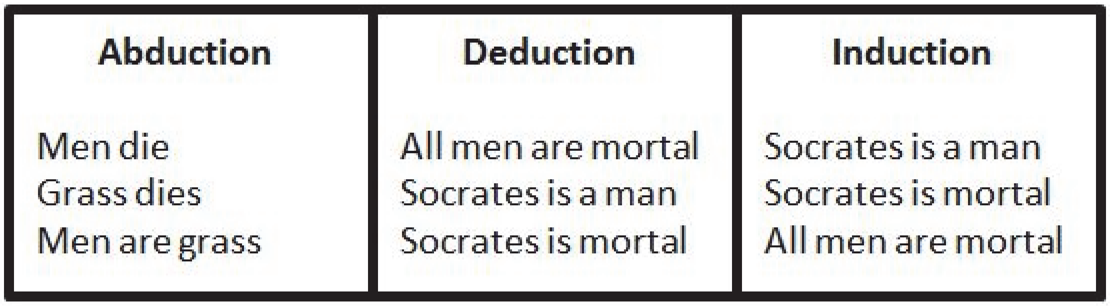
[4] gives the following rule to explain deduction and abduction:

$$\textit{room-is-hot} \leftarrow \textit{heating-is-on} \quad (1)$$

Deduction allows one to derive the fact that the *room-is-hot* from knowing that the *heating-is-on*. In other words, if the heating is on, the room must be hot. Abduction allows one to infer that *heating-is-on* to explain the fact that the *room-is-hot*. The room could be hot for other reasons, (e.g., the weather is hot outside, a fireplace is burning in the room).

[1] http://en.wikipedia.org/wiki/Deductive_reasoning

Induction is a weak prediction based on regularity of past and present observations, true only until a contrary case is found. This is not often used in intention recognition, but included here for completion.



Abduction	Deduction	Induction
Men die	All men are mortal	Socrates is a man
Grass dies	Socrates is a man	Socrates is mortal
Men are grass	Socrates is mortal	All men are mortal

Figure 4: Abduction vs. Deduction vs. Induction

Figure 4 shows all three logic approaches. The left part of the figure shows abduction, where the agreement is of the predicates (in this case, 'die'). It is not true that men are grass, but one may guess that based on the two observations. The more observations that are made, the stronger or weaker the conclusion will become that men are grass. The middle part of the figure is deduction – the fact that Socrates is mortal is provable based on the two preceding facts. The right part of the figure shows induction - a weak prediction based on regularity of past and present observations, true only until a contrary case is found. It is true that all men are mortal, but this cannot be proven from the two observations.

2.1.2.2.1.1. Abductive Approaches

Abductive reasoning allows one to infer causes of a particular observation. For instance, if one observes that there is a light in a bedroom that will not turn on, one might infer that the light bulb burnt out. This provides a reasonable explanation of the observation, since having a burnt-out light bulb is a viable reason why the light will not turn on in the bedroom. It may be the case that this inference is not warranted. The power may be interrupted in the house or the light switch may be broken. In any case, there are multiple potential causes for the bedroom light not to have turned on in the bedroom. Abduction allows one to reason backwards from the observation to the possible cause. [27]

In the specific case of intention recognition one is reasoning not about cause and effect, but about the behaviors that are being performed and the overall goal or intention that are attempting to be produced. One reasons how potential intentions would motivate the observed behavior. When viewed from this perspective, abductive reasoning is a key component of intention recognition. Abduction provides a direct link from the observed actions to possible reasons for that behavior. While other reasoning processes play a role in intention recognition (e.g., deduction), abduction plays a critical role because it is driven by observations with an unknown goal being performed. While one could conceivably reason strictly deductively from intentions to expected actions and eventually find an expected action that matched the observation, this would be an extremely inefficient way to achieve the effect of abduction (especially with the possibility of a large number of intentions). Abduction limits the scope to only those intentions that could explain the observation.

Charniak & McDermott [15] were one of the first authors to publish that intention recognition could be seen as a problem of abduction. Their focus was on *motivation analysis*, which is now referred to as intention recognition, in the area of story comprehension. They described the approach as the opposite of planning. In planning, when provided a task (intention), reasoning approaches are used to determine what actions would be utilized to achieve it. In motivation analysis, given an action, reasoning is used to determine what intentions it could help to achieve. This opposite of the reasoning provides the concept of abduction. In their work, plan schemas were represented in the form of *todo(G, A)* which meant that goal G was achieved by Action A. In this approach, if one observed an Action A instance, they could hypothesize goal G as a possible intention.

Abduction can often result in multiple hypotheses to explain an observation. [15] proposed a set of criteria for choosing between multiple hypotheses to determine which is most probable. The first criterion focuses on the preference of the hypothesis that utilizes the most specific characteristics of the observed action. As an example, if we see that John picks up a magazine, there might be two explanations, he wants to read it or he wants to throw it away. Based on the preference of using the hypothesis with the most specific characteristics, the first hypothesis is chosen because it uses the magazine characteristic of being an object that is readable, unlike the second one that treats the magazine as any other object.

A second suggested criterion is to prefer a hypothesis that requires the fewest additional assumptions. For example, using the example from above, the hypothesis of throwing out the magazine needs an second assumption that a trash can exists, and therefore may be less preferred to the hypothesis of reading because, in this case, no additional assumptions are needed.

Intention recognition has been used in dialogue understanding [28] which has itself been cast as an abductive process. Charniak represented semantic content of a sentence as a conjunctive clause in first order logic. Interpretation is taken to be the derivation of this clause from the current knowledge base, using abduction as required to explain literals that cannot be otherwise derived. Such assumed literals become the new information that is provided to the sentence. More information about how this approach has been applied to dialogue understanding can be found in Section 2.1.4.1.

2.1.2.2.1.2. Deductive Approaches

Deductive reasoning is a sound form of inference that allows one to reason from causes to effects. This type of inference plays a major role in our everyday reasoning, including our intention recognition activities. As discussed earlier, deduction is the process of reasoning from one or more general observations (premises) to reach a logically certain conclusion. In this case, the observations guarantee the truth of the conclusion.

As an example, consider the example provided in [27] where a computer user corresponds with a technical staff member in order to retrieve a file that was accidentally deleted. In response to a staff member's question "When do you need the file restored?" the user might respond "The project is due on Thursday." In order to properly interpret the user's response, the staff member would need to use abduction to conclude that the file is related to the project, and deduction to conclude that the file should be restored before the due date since it cannot be submitted before it is restored.

In general, deductive reasoning will help abductive reasoning by deriving new facts that allow abductive chains to be extended or to tie them together. In the previous example, the answer to the question cannot be answered by abductive reasoning alone. Deductive reasoning is required for the staff member to translate the project constraints into requirements on when the file must be restored.

2.1.2.2.2. Probabilistic Approaches

Probabilistic approaches to intention recognition have been implemented to address plan inference uncertainty (where multiple intentions can correspond to a set of perceived actions). Probabilistic approaches are often based on Bayesian network and (Hidden) Markov models [29]. Because the probabilistic approach is most similar to the work presented in this thesis, more attention will be given to it as compared to the other types of intention recognition approaches.

There are numerous papers in the literature that have addressed intention recognition in a formal model of argumentation [30] or abduction [31, 32]. Appelt used weights in abduction where the weights were assigned to each rule's premises. The cost that was associated with proving a conclusion was equivalent to the sum of the costs of associated premises. The cost associated with a premise was a function of three things: if it was inherently true, assumed, or proven from other rules. The best hypothesis about the agent's plan was the one given by the proof with the smallest cost. Weighted abduction includes domain-dependent knowledge about the probability of the truth a premise. The disadvantage of this approach is that, for the most part, the costs are very subjective and difference in cost values can greatly affect the overall cost of the intention and therefore make a specific intention "rise to the top" when it may not be the most appropriate one.

Formal models of probability approaches have become more popular recently. One example includes Bauer [33] work in using Dempster-Shafer theory for rating hypotheses about an agent's plan. The Dempster–Shafer theory is "a mathematical theory of evidence. It allows one to combine evidence from different sources and arrive at a degree of belief (represented by a belief function) that takes into account all the available evidence."[2] One of the arguments given by people in favor of Dempster-Shafer is that it accurately distinguishes between the lack of evidence for a proposition as opposed to the evidence against the proposition. Others have used Bayesian reasoning [34]. For example, Raskutti and Zukerman [35] provides an example of applying Bayes Rule to determine the probabilities of different hypotheses.

In the context of this thesis, Bayesian belief networks are a model for capturing belief about the probability of an action or state occurring and how that propagates to higher-level intentions being true. They are stochastic, which means that they use probability theory to capture and handle uncertainty by explicitly representing the conditional dependencies between the different information components. The network is modeled with a directed acyclic graph of dependence structure between multiple interacting quantities where the nodes represent random variables and the edges indicate conditional dependencies.

One needs to represent the belief probability distribution at each graph node. If they are discrete variables, they can be represented in tabular format which shows the belief probability of each of the child nodes based on the combination of values of its parents. In Figure 5, we show a simple example of binary nodes, i.e., they can have two values, which are true (T) and false (F).

[2] http://en.wikipedia.org/wiki/Dempster-Shafer_theory

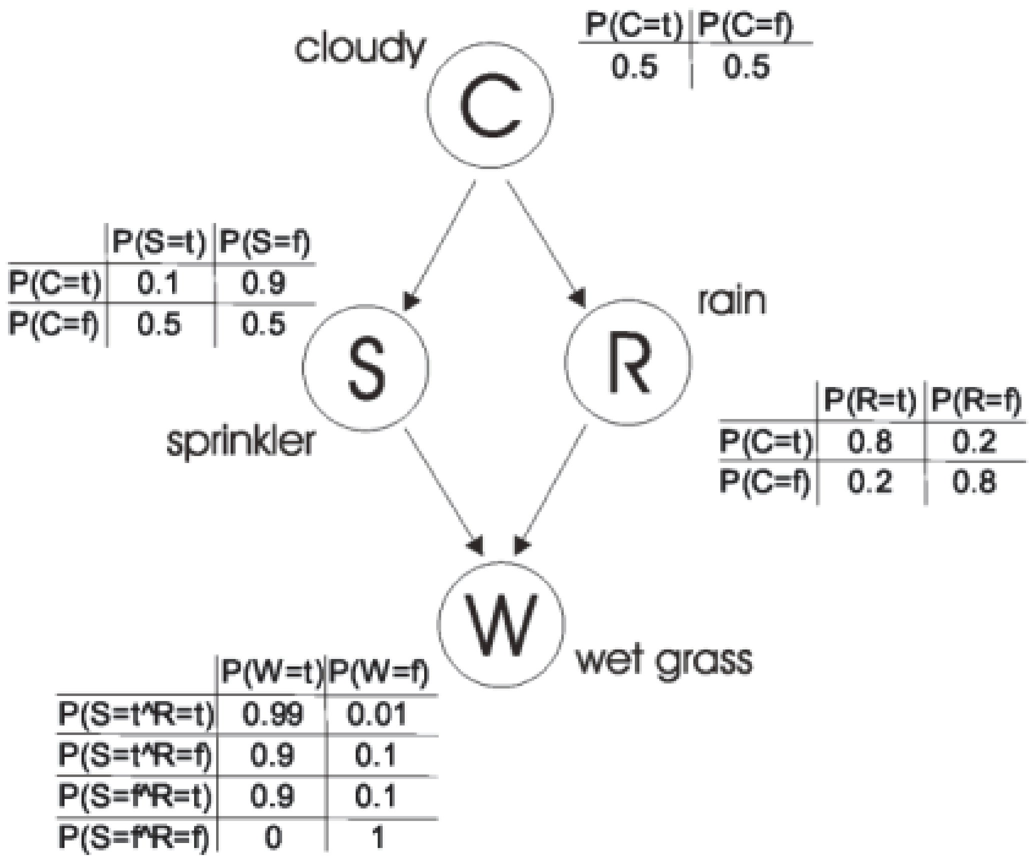


Figure 5: Sample Bayesian Belief Network[3]

Figure 5 shows the "grass is wet" event (W) has two possible causes: the water sprinkler has been turned on (S) or the fact that it is raining (R). The probability of these relationships is shown in the corresponding tables. We see that Prob(W=true given S=true, R=false) = 0.9, and, Prob(W=false given S=true, R=false) = 1 - 0.9 = 0.1, because each of the rows have to sum to one.

Charniak and Goldman [28] created the first Bayesian-based intention inference system. Their system passed markers (a way to spread activation in a network) to determine possible explanations for actions and to determine nodes to insert into a Bayesian belief network. In a Bayesian belief network, nodes stand for random variables; and the arcs connecting nodes capture the causal dependencies, which are represented by conditional probability distributions. In these conditional probability distributions, the parent node value dictates the probability of each of the possible child node values. When used for intention inference, the random variables represent proposition, the root nodes are hypotheses about an agent's intention, and the probability associated to a node captures the probability of a proposition given the observation. Bayes Rule is

[3] http://www.ra.cs.uni-tuebingen.de/software/JCell/tutorial/ch03s03.html

used to compute the probability of each proposition from the evidence (observations). As new evidence is added to the network, the node probabilities at recomputed, which propagates the evidence throughout all of the nodes. Bayesian network systems, in general, require a lot of prior, conditional probabilities. These types of systems are most applicable to domains that lend well to reliably estimating these probabilities.

Albrecht et al. [36] created an Dynamic Belief Network-based intention inference system. Dynamic Belief Networks [37] capture the influence of time by using many nodes to capture the status of variables at different instances of time. In essence, a variable can have different values as a function of time in a dynamic environment. This intention inference system was used to infer an agent's intention during a computer adventure game. The designers investigated four networks of different complexity to determine which was most appropriate for this approach, which resulted in a detailed evaluation of the impact of the different networks on the quality of intention recognition.

The results of [36] show that Dynamic Belief Networks may provide a good approach to intention recognition in cases where sufficient training data can be gathered and the network’s causal structure can be clearly determined. They did not apply it to either intended or adversarial plan recognition, as described in Section 2.1.2.1. Other Dynamic Belief Networks applications in intention recognition can be found in [38] and [39].

Brown [40] presented a multi-agent architecture providing a dynamic, uncertainty-based knowledge representation for capturing the ambiguity in uncovering a user’s intention. The knowledge representation, which is a Bayesian network, provides a formalism for determining the probability that a user is performing a specific intention. Goldman, Geib and Miller [41] presented an abductive, probabilistic theory of plan recognition. This is different than other theories in that it focuses on a plan execution model. Many other methods have represented plans as formal objects or as rules capturing the process of recognition. Their model does address aspects omitted from many other intention recognition theories, such as the cumulative effect of an observation sequence of partially-ordered, interleaved plans. The model allows for inferences regarding the plan execution evolution in cases where a different agent becomes involved in the intention execution.

Other probabilistic intention recognition approaches in the literature include [5, 42-47] [48-50] and [51].

2.1.2.2.3. Case-Based Reasoning Approaches

Using abstraction, an observing agent using a case-based reasoning approach can predict the observed agent’s behavior by mapping the observed situation to an abstract state that tracks all previous cases that have the same abstract relationship. It then uses the most closely-matched previous case as the basis for predicting and interpreting the current case. An example of this is shown in Figure 6. In the figure, abstract states (as_k) refers to ovals (bins) with disjoint equivalence classes, containing concrete past situations ($s_{i,j}$) that point to the past plans (P_i) in which they are contained through state pointers (dashed lines into the library). [52]

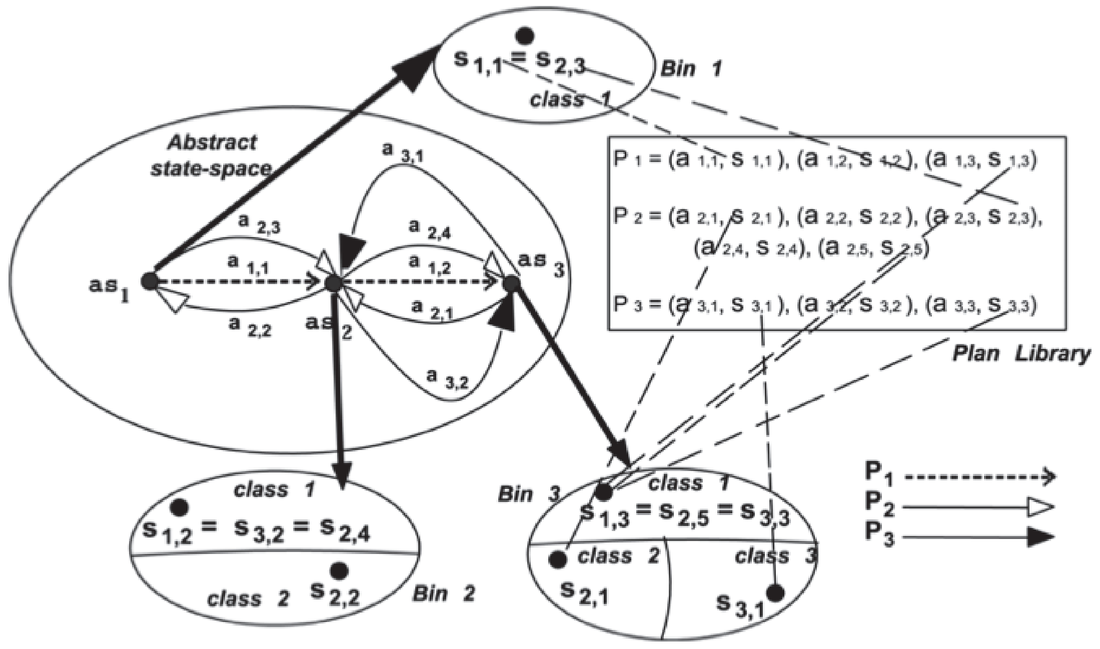

Bin 1
class 1
$s_{1,1} = s_{2,3}$
Abstract state-space
$a_{3,1}$
$a_{2,3}$
$a_{2,4}$
$a_{1,1}$
$a_{1,2}$
as_1
as_2
as_3
$a_{2,2}$
$a_{2,1}$
$a_{3,2}$
$P_1 = (a_{1,1}, s_{1,1}), (a_{1,2}, s_{1,2}), (a_{1,3}, s_{1,3})$
$P_2 = (a_{2,1}, s_{2,1}), (a_{2,2}, s_{2,2}), (a_{2,3}, s_{2,3}), (a_{2,4}, s_{2,4}), (a_{2,5}, s_{2,5})$
$P_3 = (a_{3,1}, s_{3,1}), (a_{3,2}, s_{3,2}), (a_{3,3}, s_{3,3})$
Plan Library
class 1
$s_{1,2} = s_{3,2} = s_{2,4}$
class 2 $s_{2,2}$
Bin 2
Bin 3
class 1
$s_{1,3} = s_{2,5} = s_{3,3}$
class 2
class 3
$s_{2,1}$
$s_{3,1}$
P_1
P_2
P_3

Figure 6: Case-Based Reasoning Approach [52]

This approach helps to ensure robustness when confronted with new actions and new states. The ability to do this relies on the old plan containing a state (or series of states) that map to the same abstract condition in the current state. When this is not the case, the agent is can’t accurately predict the next action the observed agent will perform. [53]

Rather than inferring every new intention from scratch, case-based reasoning (CBR) systems use can leverage previous experience represented by previously-solved problems to address new problems that contain comparable states. The typical CBR process is to find an old, comparable case, refine the old solution to address the new situation (set of observations), apply and assess the solution, and then save the new solution if it was sufficiently distinct from the previous solutions. This stored new solution can be used for subsequent executions of the CBR system. These plans are traditionally action sequences captured as operators whose executions will go from an initial state to a goal state.

Kerkez and Cox [54] developed a CBR method that interprets observations of plan behavior using an incrementally constructed case library of past observations. In each domain, the system begins with an empty case knowledge base that expands to capture thousands of past observations. It is unique in that it combines case-based reasoning and plan recognition to leverage both strengths. The plan representation is a ordered set of action-state pairs, thus different than traditional approaches that only represent actions. The technique addresses the complexity of representing both states and actions by using a form of abstraction including similarity relations to capture indices into the set of old cases in the knowledge base. As with traditional case-based approaches, past cases are leveraged to predict future actions by using old actions. In addition, the approach is able to make predictions even in case of observations of unknown actions by using the plan recognition aspects of the system. Kerkez employed evaluation criteria by measuring the accuracy of the prediction at a concrete and abstract level, and across multiple domains.

Riesbeck and Schank [55] introduced issues in case-based reasoning through a comprehensive book that presented detailed explanations of four programming efforts. Each chapter contains the program version of the information. Leake [56] wrote a book which provides detailed examples of how fundamental issues, including indexing and retrieval, case adaptation, evaluation, and application of CBR methods, are being addressed in the context of a range of tasks and domains.

Somewhat related to case-based reasoning approaches are intention graph approaches. Youn and Oh [13] described intention graphs as consisting of state, action, goal, and intention nodes and edges. It is represented as IntentionGraph = <S, A, G, I, E> where S is a set of state node, A is a set of action node, G is a set of goal node, I is a set of intention node, and E is a set of edges. The general structure of an intention graph is shown in Figure 7.

S_t represents a set of states at time step *t*. Each state node stands for a ground literal which is True. This assumes a closed-world assumption, implying that any condition which is not explicitly mentioned in the state is considered to be false. S_0 represents the initial state of set *S*, which is assumed to be completely provided.

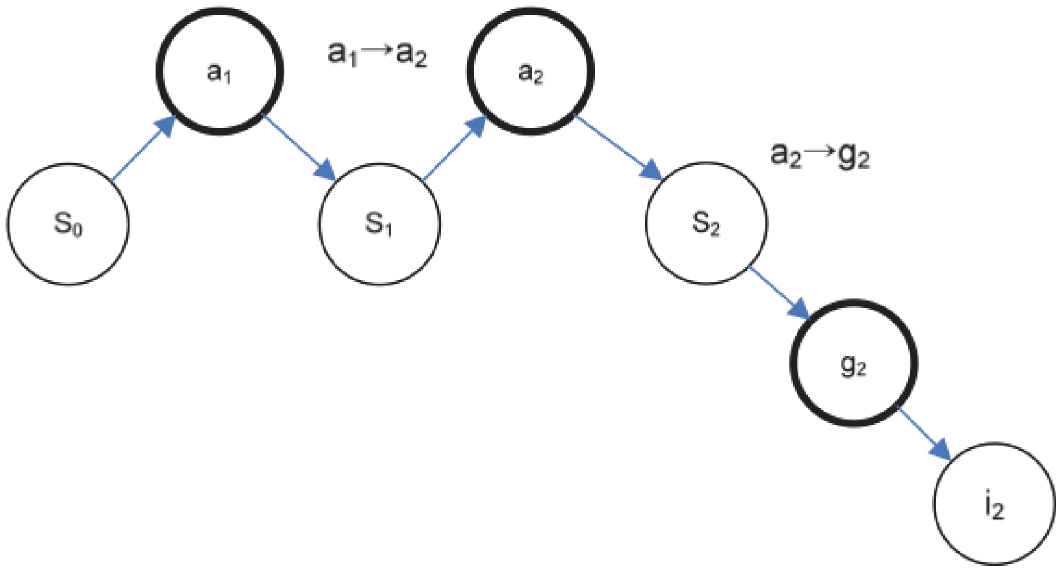


Figure 7: An Intention Graph [13]

An action schema is composed of preconditions and effects sets. A precondition set describes what has to be true in a state before the action can be performed. An effect set is describes how the state is affected when the action is performed.

2.1.3. Belief-Desire-Intention Architectures

Belief–Desire–Intention (BDI) is not an approach as described in the previous section; it is more of a software model developed for programming intelligent agents. As such, it is discussed in this section but treated differently than the previously described approaches.

BDI describes on a procedure for separating the plan selection activity (from a plan library) from the plan execution. It focuses on the definition and implementation of an agent's beliefs, desires, and intentions. This permits BDI agents to appropriately allocate resources between deliberating about plans and executing them.[4]

The idealized architectural components of a BDI system include:

- **A Beliefs Component**- Beliefs represent an agent's understanding about the world. They can include inference rules, which allows forward chaining to lead to the determination (deduction) of new beliefs;

[4]http://en.wikipedia.org/wiki/Belief%E2%80%93desire%E2%80%93intention_software_model

- **A Desires Component**- These are situations that the agent would like to accomplish. An example might be to assemble a part or to gathering information about a product;
    - **A Goals Component**- A goal is a desire that the agent is actively pursuing. This implies that the active desires must be consistent with the goal.

- **An Intentions Component**- Intentions represent what the agent has chosen to do. In other words, they are desires to which the agent has committed. This means the agent has begun to execute a plan;
    - **A Plans Component**- Plans are sequences of actions to achieve at least one intention. Plans can be represented at different levels of abstraction and may therefore contain other plans. For example, a plan to cook dinner may include a plan to make pasta and a plan to steam vegetables.

- **An Events Component**- Events are triggers that cause an agent to perform a reactive response. An event may update the state of a belief, it could trigger a plan to start, or it could change the goals. Events may occur in the external world and are perceived by sensors or they may occur internally to trigger updates or initiate activity plans.

The relationships between some of these concepts are shown in Figure 8.

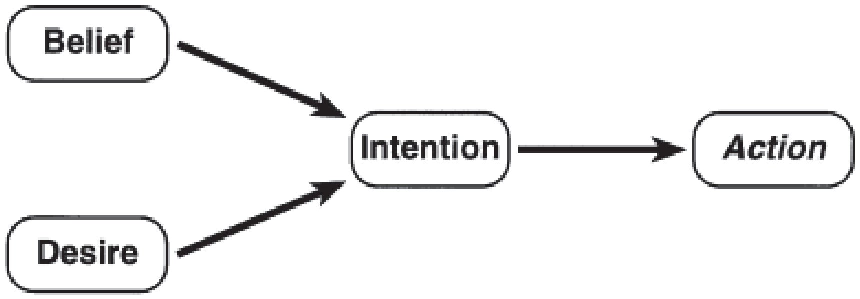


Figure 8: High-Level DBI Architecture

BDI approaches have been applied to a wide variety of applications. Taillandier, Therond and Gaudou [57] applied the BDI architecture in a simulation context to the problem of agricultural cropping planning and decision-making. They propose a new architecture based on the BDI paradigm that copes with the challenges that the BDI paradigm is often complex to understand by non-computer-scientists and they are often very time-consuming in terms of computation.

Dignum, Morley, Sonenberg and Cavedon [58] described an approach to social reasoning that integrates prior work on norms and obligations with the BDI approach to agent architectures. Pokahr, Braubach and Lamersdorf [59] described Jadex, a software framework for the creation of goal-oriented agents following the BDI model. The objective is to build up a rational agent layer that sits on top of a middleware agent infrastructure and allows for intelligent agent construction using sound software engineering foundations.

While BDI architectures have shown promise and have been used extensively in the literature, they do have some limitations. They include:

- BDI agents to not have mechanisms to perform machine learning from past behavior. [60, 61];
- Some classical decision theorists question the need for all three types (beliefs, desires, and intentions), while some AI researchers questions whether the three are enough [62];
- Some believe that the multi-modal logics that underlie BDI have little relevance in practice [61, 62];
- The BDI model doesn't include guidance for interacting with other agents [63];
- Many BDI implementations do not represent goals explicitly [64];
- The architecture cannot perform look-ahead deliberation or forward planning [65].

In the next section, we will explore how these various approaches to intention recognition have been applied to different application domains.

2.1.4. Applications Domains

In this section, we will describe domains (in no particular order) in which intention recognition systems have been historically applied, along with a brief overview of how researchers in the field have applied them.

2.1.4.1. Language and Story Understanding

In story understanding, intention recognition systems have been used to recognize the plans of a character based on the described actions in a story in order to answer questions based on the story. One can liken this to a test that a student would take after they read a story to determine if they had a deep understanding of the characters' goals. Story understanding is challenging, in that the actions might not be described in the story in the actual order that they occurred. In this and other domains, the system must allow actions to occur simultaneously with each other, or allow the temporal ordering not to be known at all. Also, one must allow the possibility that an action may be executed as part of two independent plans.

In [66], Kautz addressed this challenge by proposing that intention recognition be considered as deductive inference based on an observation set, a taxonomy of actions, and one or more simplicity constraints. In their work, the taxonomy of actions is a complete description of how actions can be executed and how any sequence of actions can be mapped to a complex action.

An action taxonomy is obtained by applying two closed-world assumptions. The first assumption states that the described ways of executing an action are the sole ways of performing it. Each time an abstract action is specialized, more is known about how to perform it. For example, because the action type "throw" specializes the action type "transfer location", we can think of throwing as a way of transferring location of an object.

The second assumption states that all actions done for a purpose, and that all the possible reasons for executing an action are known. This assumption is realized by stating that if an action A occurs and P is the set of more complex actions in which A occurs as a sub-step, then some member of P also occurs.

2.1.4.2. Interactive Storytelling

A primary research area in interactive storytelling is how to create stories that are interesting and coherent. It is useful to allow for means to allow the user to play a part in the story while guaranteeing that user intervention won't allow for events that go against the bounds of the intended genre. Plan recognition and generation tools have been developed to allow a system to infer the intention of the person while interactively developing a story so that the next logical steps in the story can be automatically determined.

Karlsson, Ciarlini, Feijo and Furtado [67] described the usage of an intention recognition paradigm in LOGTELL, which is a logic-based tool for the interactive creation of stories. The basis behind this work is two-fold: 1) the description of a logic model to describe the behavior of characters and events; and 2) a tools that aids the interactive composition of story plots by leveraging fully or partially generated plots. The user can interact with the system at different abstraction levels, generating a variety of stories in line with an individual's tastes, and within the imposed understandability criteria. The system toggles between stages of goal inference, planning, plan recognition, user intervention, and 3D visualization.

2.1.4.3. Interface Design and Implementation

Intention recognition can help in the design and implementation of user interfaces by providing a model containing possible user intention to make interfaces more intelligent and interactive. By trying to infer the intention of the user, the system can dynamically adjust the user interface to better match the possible next steps that a user may take based on their perceived intention..

Goodman and Litman [68] showed how interface tasks impose constraints that have to be satisfied for any intention recognizer to create a plan that efficiently support the development of an intelligent assistant. They specifically looked at two questions: 1) how can information about a plan be leveraged to design interface tasks to support different communication types?; and 2) how can the tasks constrain creation of intention recognition algorithms? Their research was developed using CHECS, which is a plan-based design interface.

2.1.4.4. Collaborative Problem Solving

The field of human-computer interaction treats the human and computer and collaborators. In order to achieve successful collaboration, the collaborators must have a common understanding of their shared goals and the actions needed to accomplish them. Verbal communication is often used to achieve this mutual understanding. However, it is usually more natural to convey intentions by doing actions and allowing the other agent to observe these actions and infer the intention [69]. For example, if it is dinner-time and one person starts looking up restaurant phone numbers online, the other person may infer that the proposed plan is to find a restaurant, place an order for delivery, and eat at home.

Lesh [69] described how to focus on aspects of a collaborative environment to make intention recognition more practical. The aspects they describe on are: the focus of attention; partially elaborated hierarchical plans; and the possibility of requesting clarification. They demonstrated their approach in the via a collaborative email system. They how plan recognition reduced the communication burden of the user. To implement their system, they used an approach based on the SharedPlan theory [70] of task-oriented collaborative discourse. They formalized the task structure in terms of high-level goals, such as “checking email”, lower-level goals, such as “writing a message,” and individual actions such as single clicks on the interface.

2.1.4.5. Assisted Technologies and the Care of Elderly and Disabled At Home

Over the past 20 years, there has been a large jump of the average age of people in most western countries and it is expected that this number will be constantly growing [18]. To address this, there has been a significant focus on supportive technology for older people living by themselves in their own homes.

Pereira and Ahn [18] described a system they created for elder care applications in order to provide help for elders. Their system recognizes the intentions of elders and then provides recommendations on how to accomplish the perceived intentions. They employed Causal Bayes Networks and plan generation techniques. They used the Evolution Prospection Agent (EPA) system to provide suggestions for realizing the recognized intention. This system prospectively looks ahead to choose the best set of actions to realize the recognized intention, while being cognizant of the environment, of the person's desires, and of planned future events.

Smart homes could help to improve the autonomy of cognitively impaired people, and thus alleviate the burden placed on caregivers [71]. Research in the literature strives to turn the home into a "cognitive prosthetic." In this process, behavior tracking and intention recognition are fundamental pieces of the vision of a smart home environment.

Giroux [71] presented a cognitive assistant to show how intention recognition can help to address initiation, attention, planning, and memory cognitive defects in people with disabilities. An experiment was conducted using a cognitive assistant (Archipel) involving 12 people with mild intellectual disabilities. During the first day, a smart apartment and the Archipel assistance were provided to the subject. The researcher and the subject created a recipe together monitored by Archipel to let the subject to get familiar with the task and the environment. In the next two days, participants were asked to create a recipe with and without Archipel. Archipel was used to monitor the actions of the participant and to infer what intentions were being performed to allow Archipel to guide the participant through the process. The count of human interventions needed by the participants to complete both recipes was tracked and used as an indicator of behavioral autonomy. Archipel dropped human assistance by 50 percent.

Roy [16] used intention recognition to predict Alzheimer patients' behaviors to identify techniques to support them in performing their daily activities. Because the subjects were Alzheimer patients, this situation raised the dilemma that an observation of a perceived novel action, different from an expected action, may not be interpreted as a mistake. This instead could represent the start of a second plan, when the first plan was aborted or forgotten. To address this, they created a hybrid recognition model based on probabilistic description logic. Using the hybrid model, they performed an experiment on 106 Alzheimer's patients. They analyzed the cognitive performance of each patient based on the Kitchen Task Assessment (KTA) [72], involving 17 common errors of patients, each modeled in distinct scenarios. A knowledge database was created composed of 40 primitive actions and 10 activities of necessary for daily living. These activities consist of kitchen task (cooking cake, cooking pasta, making tea, etc.). The results showed that their model recognized almost 100% of the errors performed by the Alzheimer patients.

2.1.4.6. Recognition of Intention Behind Bar Chart Graphs

A rather unique use of plan and intention recognition is its application towards inferring the intention of designers of non-pictorial graphs such as bar charts and line graphs. Many information graphics that appear in popular media have a message they are trying to convey. Applying intention recognition systems to this domain would allow one to assess the impact of different communication approaches on an information graphic's success at conveying a message.

Carberry and Elzer [73] created an approach to plan inference of information graphics. In their work, the graphic designer is treated as the user whose intention is being modeled, and the intention inference system tries to infer the intention that is intended by the user. This is similar to intention recognition in language understanding, where the speaker wants the listener to hypothesize the speaker's intention. One interesting application of this is in the area of assistive technology, where the intention recognition system infers the graphic's intended message and speaks it to an individual with sight-impairments.

2.1.4.7. Computer Security Intrusion Detection

Intrusion detection systems (IDSs) currently describe actions that have already happened and do not predict what future actions may occur. For IDSs to be successful, they must be able to analyze a hacker's actions, infer the goals of the hacker, and make predict the hacker's actions in the future [24]. Intention recognition plays a vital role in inferring future intentions.

Geib and Goldman [24] built a set of Intention Recognition Algorithms for computer security intrusion detection and prediction based on probability distribution over the combination of all probable actions which could occur next. For cooperative agents with complete and accurate observations, this is acceptable. However, for hostile agents (adversarial intention recognition as described earlier), they cannot assume that all actions are observed. To complicate for this, they extend the observed actions with possible unobserved actions that are consistent with the observed actions, state changes, and the plan graph. They then determine a set of plausible execution traces, thus attempting to infer the intention of the hacker. These execution traces enable them to create the possible sets and then use the probability distribution over the sets of hypotheses of goals and plans implicated by each of the traces and pending sets to infer the hacker's intention. An example of this is shown in Figure 9, where various actions were represented in a plan graph. The challenge of plan recognition was therefore a problem of covering aspects of the graph. They focused on computing minimal explanations, as represented by vertex covers of the plan graph.

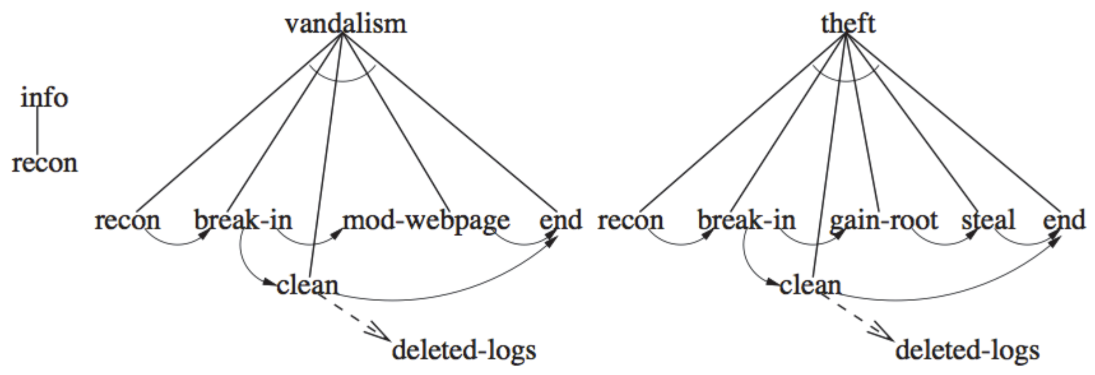


Figure 9: Graph-Based Approach to Intention Recognition [24]

2.1.4.8. Human-Robot Interaction

There is a significant amount of literature in the area of intention recognition for human-robot collaboration, though all of these approaches use activity recognition as the primary input for the intention recognition system. Schrempf, Albrecht and Hanebeck [48] used a form of Bayesian networks with a reduced state space to allow for tractable on-line evaluation. They utilized a combination of telepresence techniques and virtual environments in a kitchen-like environment. In this environment, they were able to reduce the state space from 2^{16} to 17. Wang et. al. [74] proposed the Intention-Driven Dynamic Model (IDDM), which is a latent variable model for inferring unknown human intentions. They applied this model to two human-robot scenarios including robot table tennis and action recognition for interactive robots. They showed that modeling the intention-driven dynamics can achieve better prediction than algorithms without modeling dynamics. Goto [75] explored human-robot collaboration in assembly tasks. Focusing on the assembly of a table, they used finite state machines and a set of visual recognition routines for state transition detection to recognize the human's intention. They also incorporated verbal messages as cues during various stages of the process. Tahboub [47] used Dynamic Bayesian Networks with a focus on eliminating cycles by time delay. In essence, they feed back sensed states from previous time slices instead of the current one. This paper stressed the need to handle qualitative spatial data, though they did not go into detail on how to address this. This need will be focused on in this thesis. In all of these cases, activity recognition is the primary input into the intention recognition system, with state information being of secondary importance.

2.1.4.9. Possible Future Application: Computer Games

The current gaming industry around the world is one of the fastest growing industries [76]. One very popular gaming genre is real-time strategy games, where gamers play against other gamers in real-time. However, traditional implementations of games when played against computer agents apply extensive usage of finite state machines that makes them very predictable (since subsequent actions are only a function of the state that exists at a given time) and provide less unique replayability.

Cheng and Thowanmas [76] presented a thought paper describing some of the areas that intention recognition approaches can be applied, building off of work by Buro and Furtak [77]:

- **Adversarial Real-time Planning** – Planning can take place in three levels:
    - Strategic planning focuses on what should be done;
    - Tactical focuses on how to carry out plans;
    - Operational focuses on specific actions for each tactical decision;
- **Decision-making Under Uncertainty** – Humans decide on specific strategies even with lack of information. They are can also proactively determine the necessity to look for additional information to gain an advantage;
- **Opponent Learning and Modeling** – The ability to infer a player's strategies and find ways to react to them. This has been sought after for many years, but not yet achieved. Most current games still follow a pre-determined plan;
- **Spatial and Temporal Reasoning** – Strategies and plans have to be constantly reassessed for applicability, especially in an environment that constantly changes;
- **Path Finding** –The ability to quickly determine a path in a 2D terrain with mobile objects and ever-changing environments.

Cheng Towanmas stated that the concept of opponent learning and modeling is ripe for intention recognition approaches, but to date have not been successfully applied.

The next section will describe some of the metrics that have been used to assess intention recognition systems.

2.1.5. Metrics for Intention Recognition

Surprisingly, the literature is very light on metrics that have been applied to measure the performance of intention recognition systems, in a general sense. The one exception was work that was performed by James Mayfield [78]. In this work, he proposed three criteria including, applicability, grounding, and completeness for assessing intention recognition systems. He applied these metrics to dialogue systems, but the metrics seem to be ubiquitous enough to be applied outside of this domain. These criteria are described below.

2.1.5.1. Applicability

The principle of applicability states that a good explanation of an intention is applicable to the needs of the system that will use it [78]. That is, no matter how good an explanation might be in other respects, if it doesn't provide the information that the system needs to do its job, then it is not a good explanation.

An assumption that underlies the principle of applicability is that a system that is trying to explain an intention has interests of its own, which understanding the intention might help to further. For example, on a collaborative human-robot manufacturing facility, a robot might be trying to help a human to accomplish an assembly task. By having the robot infer the intention of a human at a given time, it may be able to deduce what actions it can best take to either help advance the assembly process or at a minimum, make sure that it stays at a safe distance away.

According to Mayfield, the principle of applicability implies that the operation of an intention recognizer cannot be independent of the system in which it is embedded, and therefore the concept of a domain-independent intention recognizer is impractical.

The three dimensions of a description along which applicability may be assessed include content, composition, and granularity. These three dimensions are described below, and have been modified slightly to show their applicability outside of the dialogue system domain.

Content applicability deals with the particular choice of elements that compose an intention description. The principle of content applicability holds that each of the components of a particular description should be applicable to the needs of the system. Not all such intentions will be applicable to a given system though. A good description of an intention will not include all of these goals, but will include only those that are applicable to the purposes of the system.

Composition applicability concerns the type of concepts out of which intention descriptions should be constructed. Concepts could include the objects that are used to describe the intention along with how they are combined in the description. The principle of composition applicability holds that the concept types that compose an intention description must be applicable to the needs of the system. The composition of a good description is largely a reflection of how that description will be put to use. Different tasks require different types of descriptions.

Granularity applicability covers the level of generality of the elements of a particular description. The principle of granularity applicability holds that a good description contains the optimal amount of detail. It is often possible to divide a single action into a number of sub-actions. The extent to which an explanation is divided in this way is its level of granularity. Different levels of granularity may be more or less applicable to a given system.

2.1.5.2. Grounding

The second criterion for evaluating a description is the principle of grounding. This principle states that a good description relates what is inferred to what is already known about the agent and the environment. Typically, before perceiving a new activity or state, a system will already have some knowledge of previous actions or states that is applicable to the processing of it.

2.1.5.3. Completeness

The principle of completeness states that a good description of an intention covers every aspect of the activity or state; it leaves no relevant portion of the activity or state, or of the description itself, unexplained. Two kinds of completeness include depth completeness and breadth completeness.

Depth completeness is comprehensiveness of an individual aspect of the state of the environment. A description of a state exhibiting depth completeness would consist of the motivating goal of that state and a goal to explain each previously inferred goal. Breadth completeness is coverage of all parts of the environment. A description that consists of breadth completeness includes every goal that contributed to the existance of the state.

With this background on ways of performing intention recognition, the domains in which they have been used, and how they has been measured, the next section will start to explore some the aspects that make this thesis unique. As mentioned earlier, this work is taking a novel approach to intention recognition by using state information to identify intentions as opposed to using activities. The next section will also explore the state-of-the-art in both activity and state recognition to show the advantages of using a state-based approach.

2.2. Activity Recognition vs. State Recognition

One of the core assumptions of this work is that existing technology can more accurately identify and model information about the state of the environment as compared to information about activities that are being performed in the environment. Based on this assumption, the output of an intention recognition system that uses state information as input should therefore be better than one that uses activity information as input. While there have been many efforts that have attempted evaluate the performance of state and activity recognition systems, they have primarily been self-evaluated in configurations and environments that have been most conducive to their approaches. There have, however, been some open, impartial competitions/evaluations in these areas [79, 80]. Below we describe a state recognition competition and an activity recognition evaluation performed by external, impartial parties. We will use the results of these competitions as benchmarks to characterize the state–of-the-art in these two fields.

State recognition can encompass many things, including recognizing objects' color, size, shape, location, and pose, as well as identifying the object itself. Later in this thesis, we describe an intention recognition approach that relies on the identification of an object as well as determining its location and pose to characterize its spatial relationships with other objects. As such, we will use the competition described in Section 2.2.1 as the basis for the performance of state recognition technology since it is well aligned with the focus of this thesis.

2.2.1. State of the Art in State Recognition

In 2011, a Solutions in Perception Challenge (SPC) was held as part of the International Conference on Robotics and Automation (ICRA) in Shanghai, China. [81]. The purpose of this event was to determine the state of the art of robotic perception algorithms. There are many algorithms that exist world-wide for object identification and pose determination (position and orientation), yet it is difficult to determine if an algorithm is applicable to a given task and to know its robustness. Also, these algorithm development efforts are occurring all around the world, but there is no easy way of knowing which algorithms have already solved particular perception challenges [80]. The SPC sought to identify the best available perception algorithms for specific challenges. They were documented as open source software to address duplicate development efforts and accelerate the development of advanced perception algorithms.

The focus of the 2011 Challenge was Single and Multiple Rigid Object Identification and Six Degrees of Freedom (6DOF) Pose Estimation in Structured Scenes. Teams developed algorithms that could "learn" a specified number objects from the provided 3D point cloud data. This data was enhanced with corresponding red-green-blue (RGB) point color. The teams were then tested on how well they could correctly locate and identify the same objects in a scene.

The training and evaluation data sets were assembled using 16 machined aluminum artifacts representing commonly-encountered features of manufacturing parts (Figure 10). Each artifact was created from a unique computer-aided design (CAD) model. The artifacts were grouped into three classifications based on the perceivable features. Group 1 consisted of objects with maximal heights greater than two inches. Group 2 objects were shorter than two inches, but had raised features that gave them non-level surfaces. And Group 3 objects were shorter than two inches, and had level surfaces. The artifacts were designed to be comparable with industrial assembly parts such as automotive or aircraft parts (see Figure 11).

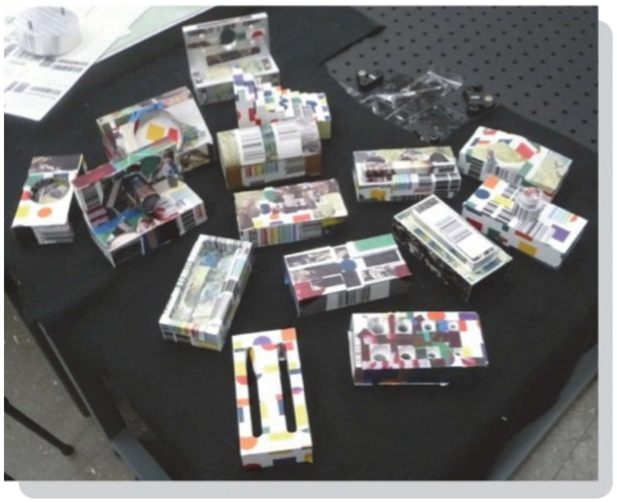


Figure 10: Sample Machined NIST Artifacts Used in the 2011 SPC, Arranged Randomly.

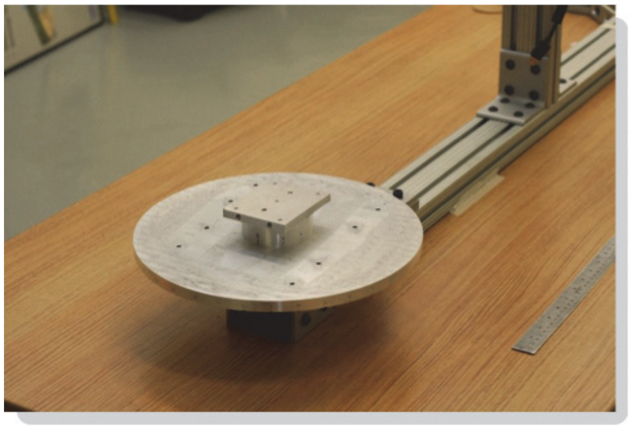


Figure 11: The Ground Truth Fixture

The submitted algorithms were evaluated based on two criteria (rounds). Round 1 consisted of image frames with only one artifact. Round 2 consisted of three artifacts each. Both rounds were contained several sub-runs which varied the objects' translation and rotation. Run 1 included of only object translations; Run 2 included only object rotations; and Run 3 had a combination of the two.

For each run in Round 1, a single artifact was randomly selected from the three classification groups. The pose of the artifact was set that was consistent with the run-based transformation restrictions for each run, and the artifacts were each applied to the same subset of transformations. For Round 2, a single artifact from each of the three classification group was randomly selected. Random poses were generated for each run, with each artifact assigned an independent location on the ground truth fixture. There were 399 frames used as input for the participants.

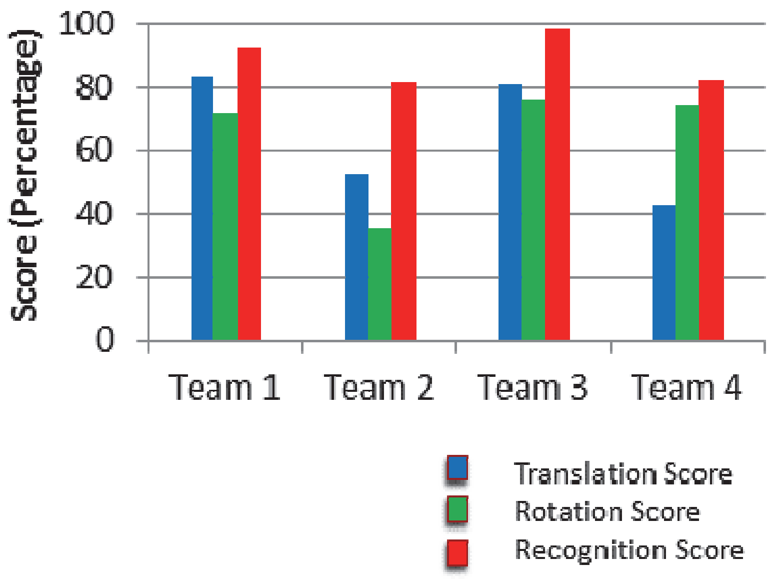


Figure 12: Translation Scores Over All 399 Frames

For each data frame, the ground truth was composed of one or more objects. A true positive count (*hits*, c_h) shows that the algorithm properly determined an object within a scene. A non-zero false positive count (*noise*, c_n) shows that an algorithm identified objects that were not present within a scene, and a non-zero false negative count (*misses*, c_m) shows that the algorithm couldn't accurately identify objects in the scene.

The true positive counts were counted over all of the frames. The participants correctly identified over 80% of all objects over all frames (at least 665 artifacts of the 831 present in all 399 frames).

Although seven teams originally participated in the event, only four were able to finish. The other three were excluded due to hardware and software issues. The remaining four teams scored an average of 65% on the translation test (within the allowable tolerance) and 65% on the rotation test (see Figure 12).

2.2.2. State of the Art in Activity Recognition

Activity recognition can be roughly divided into two camps. The first focuses on sensor-based activity recognition which involves the person in the environment wearing some type of sensor (which could be a smartphone, accelerometer, or other tracking device) that provides data as to where the person is and the motion s/he is taking. Examples of this are included in [82] and [83]. The second is vision-based activity recognition, in which the behaviors of agents are characterized using videos taken by various cameras. Examples of this type of work can be found in [84] and [85].

For the work described in this chapter, the focus is on approaches that do not require the agent to wear any external or internal device, as this is not reasonable or practical in an industrial setting. Therefore we will only consider vision-based activity recognition. In addition, as mentioned earlier in this section, we will limit our analysis to evaluations that were performed by an external evaluator (i.e., not the developers of the systems) to remove any conflict of interest and ensure an unbiased evaluation.

According to the "Defense Advanced Research Projects Agency (DARPA)" web site, the Mind's Eye Program[5] "*seeks to develop machine-based visual intelligence by automating the ability to learn generally applicable and generative representations of actions between objects in a scene directly from visual inputs, and then reason over those learned representations. The focus of this project is on the military domain, where Army scouts are commonly tasked with covertly entering uncontrolled areas, setting up a temporary observation post, and then performing persistent surveillance for 24 hours or longer. A truly "smart" camera would describe with words everything it sees and reason about what it cannot see. These devices could be instructed to report only on activities of interest, which would increase the relevancy of incoming data to users. Thus, smart cameras could permit a single scout to monitor multiple observation posts from a safe location*."

To evaluate the activity recognition, the DARPA program identified 2,588 short vignettes (approximately 15 seconds to 20 seconds each) of 48 different activities. Each activity was represented by approximately 54 of the short vignettes. The activities that were explored are shown in Table 1.

[5] http://www.darpa.mil/Our_Work/I2O/Programs/Minds_Eye.aspx (August 14, 2012)

Table 1: Actions Used for Activity Recognition

Approach	Close	Flee	Have	Open	Run
Arrive	Collide	Fly	Hit	Pass	Snatch
Attach	Dig	Follow	Hold	Pick up	Stop
Bounce	Drop	Get	Kick	Push	Take
Bury	Enter	Give	Jump	Put down	Throw
Carry	Exchange	Go	Leave	Raise	Touch
Catch	Exit	Hand	Lift	Receive	Turn
Chase	Fall	Haul	Move	Replace	Walk

The eight systems under evaluation had to determine if one or more of the verbs listed above were present in the vignettes. There were two metrics used. The first was precision which is defined as:

$$\boldsymbol{Precision} = \frac{VerbMatches}{VerbMatches + FalsePositives} \tag{1}$$

The precision results from each team are shown in Figure 13. Team scores ranged from 21% to 60%.

For the second metric, the program used the Matthews Correlation Coefficient (MCC), which is a balanced measure of correlation which can be used even if the classes are of different sizes (in lieu of accuracy). The formula is:

$$\boldsymbol{MCC} = \frac{TP * TN - FP * FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \tag{2}$$

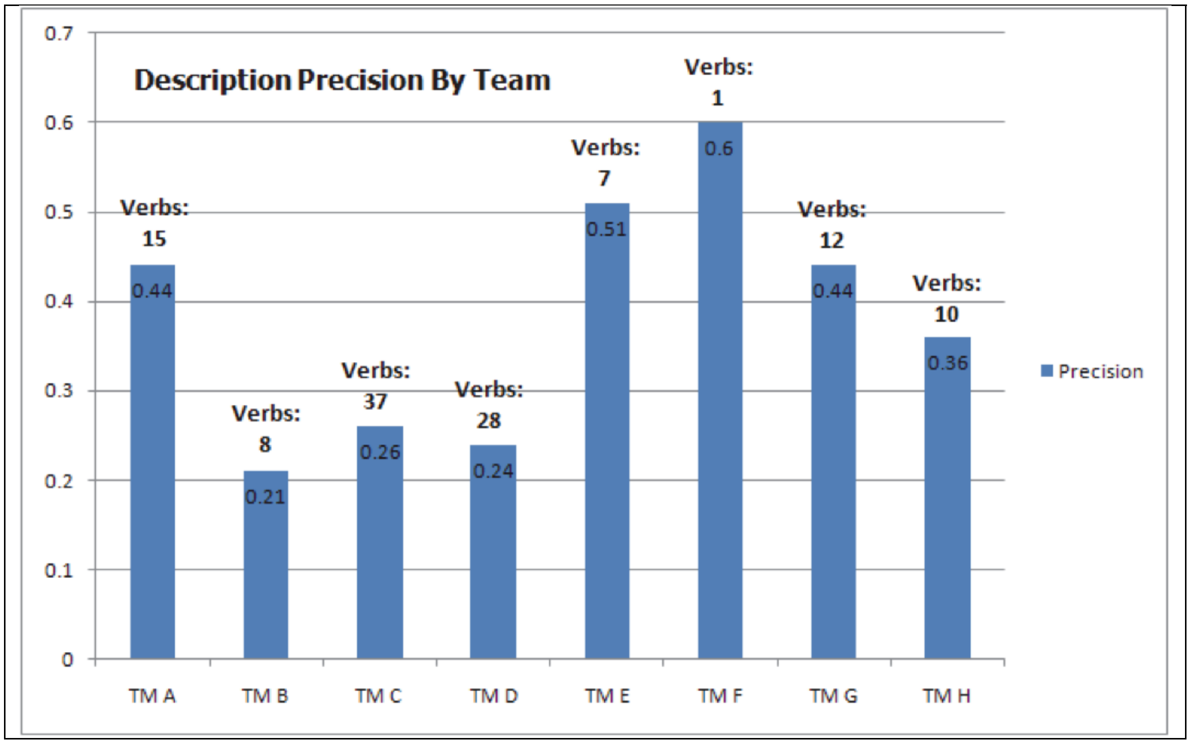


Figure 13: Precision Results from Mind's Eye Evaluation

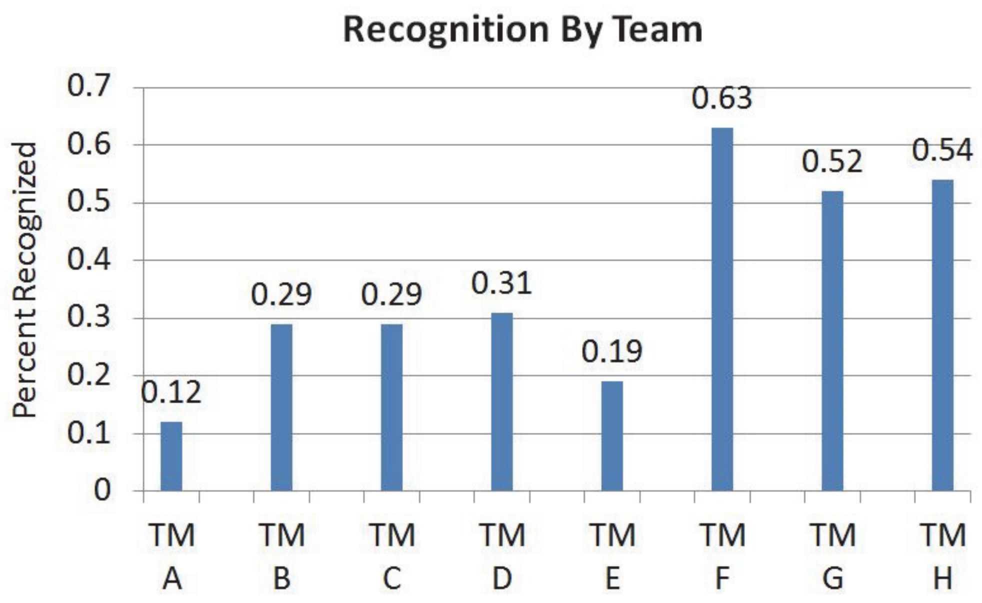


Figure 14: Results of MCC Metric

TP are true positives, TN are true negatives, FP are false positives, and FN are false negatives. The results of applying this metric are shown in Figure 14 and are very similar to the precision metrics, where systems scored between 12% - 63% when using this metric, with an average of approximately 36%.

2.2.3. Comparison of State and Activity Recognition

In both cases, the system was given a scene or video and asked to characterize it by stating either the type/pose/orientation of an object or the existence of an activity. While it is impossible to perform a direct comparison between these two approaches, these examples provide a sampling of the type of performance that can be expected by each type of technology. Based on this analysis, it appears that state recognition systems are approximately 40% (75% vs. 35%) more accurate than the activity recognition systems. This shows the benefit in using state information as the basis for performing intention recognition as opposed to using activity information.

The next section will describe various approaches to state representation in the literature, with an emphasis on ontology-based state representation as this will be the focus of this thesis.

2.3. State Representation

In this section, we will explore approaches to representing spatial and state information, which an emphasis on ontology-based approaches. [86] gives an excellent overview of this field; we will summarize the key points below as it relates to this thesis. In this thesis, spatial information refers to the relative and absolute physical location of objects in the environment. State information is more general, and can also include information such as color, shape, size, etc.

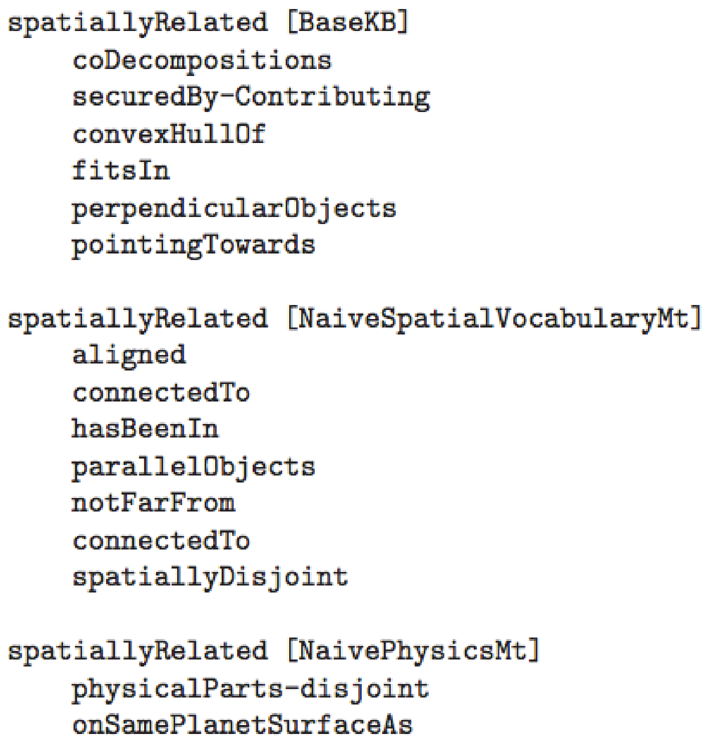
2.3.1. States and Spatial Knowledge in High-Level Ontologies

Foundational ontologies, such as Basic Formal Ontology (BFO) [87], Descriptive Ontology for Linguistic and Cognitive Engineering (DOLCE) [88] and General Ontological Language (GOL) [89] provide the fundamental concepts needed in ontologies. These include the high-level concepts and the attributes or characteristics that pertain to them, all stitched together with axioms which provide their relationship to each other. Typical types of information that are captured in these ontologies are attributes of physical objects and how those attributes change over time. One such type of attribute is the spatial relations between and among these physical objects. However, in these ontologies, the level of abstraction is so great that they are not directly useful to detailed applications [90].

At a level lower, one can refer to general ontologies. Although often overlapping with foundational ontologies, general ontologies specialize the categories needed to represent detailed spatial relations. General ontologies, such as Cyc [91], Omega [92], SUMO [93], and SWIntO [94], typically encompass a broader set of information about specific concepts similar to what one would find in an encyclopedic. Ontologies can also contain abstract axioms about space (such as in SUMO and Cyc).

SUMO [93] combines various approaches to space and spatial relations to provide wider coverage of the domain. It unites theory of holes [95] with aspects of mereotopology and specifies a set of axioms capturing basic and fundamental concepts, such as that physical objects can be situated in space and time. Locations are further characterized as regions, which allow provide for the specialization of specific spatial relations such as exactlyLocated, partlyLocated, among others.

The Cyc ontology [91] leverages various sources but does so using modularization by decomposing its description into a set of microtheories. In the area of spatial theory, some of the microtheories include naive geometry, naive physics, and naive spatial, which was developed to capture the natural way humans perform spatial relation reasoning [96]. The high-level entity for space is called SpatialThing, which possesses a diverse collection of subtypes, as shown in Figure 15. An overview of Cyc and its representation of space can be found in [97].



```
spatiallyRelated [BaseKB]
    coDecompositions
    securedBy-Contributing
    convexHullOf
    fitsIn
    perpendicularObjects
    pointingTowards

spatiallyRelated [NaiveSpatialVocabularyMt]
    aligned
    connectedTo
    hasBeenIn
    parallelObjects
    notFarFrom
    connectedTo
    spatiallyDisjoint

spatiallyRelated [NaivePhysicsMt]
    physicalParts-disjoint
    onSamePlanetSurfaceAs
```

Figure 15: Sample Cyc Spatial Relations

Not surprisingly, in less complex ontologies, less useful details are often provided. Many of them simply have an inheritance hierarchy plus scattered attributes and axioms. These types of ontologies focus on large semantic lexicon initiatives [98] – and often only provide thesaurus-type descriptions for structuring concepts. These ontologies provide a basis for knowledge related to spatial relations to be inferred, but often little else. For example, some information pieces that could be included are geographic entities, building types, types of spatial relations, etc., but with little detail about their meaning. There is a large variation in types of ontologies, with some containing loose definition and some containing formal axioms. In addition, whenever axiomatization does not exist in an ontology, the designers frequently do not follow basic principles of ontological modeling, such as those described in [99, 100].

There are also application-specific ontologies. These often limit axiomatization and definitions to only what is necessary for their specific application needs. Although not considered to be ontologies, this information can often be very valuable for creating more formal ontologies for those domains. Example domains that contain a rich amount of state and spatial relations include mapping and geographic information standards, including the ones being developed in the International Organization for Standardization (ISO) technical committee on geographic information (ISO/TC 211, Geographic Information/Geomatics: http://www.isotc211.org/), the Open Geospatial Consortium (OGC: http://www.opengeospatial.org/), the spatial reference model of the Synthetic Environment Data Representation and Interchange Specification (SEDRIS: http://www.sedris.org/), the Building Information Model (BIM), the W3C Geospatial ontology incubator group (http://www.w3.org/2005/Incubator/geo/XGR-geo-ont/), and national mapping agencies such as the U.K. Ordnance Survey or the U.S. Geological Survey [101].

Well-known structures (though perhaps not ontologies in the formal sense) such as EuroWordNet [102], FrameNet [103], WordNet [104], and VerbNet [105] all including a large amount of terms related to space– including concept such as place, North, South, locations, parts, regions, along with specific spatial objects– yet, as discussed above, many of them are little more than a hierarchy of terms with few definitions. Some of the spatial relations that do occur are modeled using case roles, but these roles do not provide formal semantics.

2.3.2. Approaches to Spaces Representation Within Detailed Ontologies

Unlike the foundational and general ontologies discussed above, there are a large set of detailed ontologies that offer representations of space and spatial semantics. Here, it is beneficial to focus on the areas of qualitative spatial representation, which will be the focus of the state representation approach present in this thesis. In qualitative representations, a situation is characterized by variables that can only take a small, predetermined number of values [106]. Qualitative descriptions can include concepts such as *in_front_of*, *behind*, *on_top_of*, *next_to*, etc. This makes the specifics of precise metrical information more abstract.

It is well cited in the literature that qualitative descriptions are more representative of human perception, as this is how human typically refer to the spatial relations between objects. The argument against this focuses on the intuitive level – for instance, the concept of *geometric points* is often rejected as an adequate abstractions because zero-dimensional objects cannot be seen [107]. Mark [108] wrote about the relation between human judgments of relations between regions and lines, including particular spatial calculi distinctions. Rauh, Renz and Knauff [109, 110] explored the degree of fit between qualitative spatial descriptions and preferred mental model building during the task of problem solving. Klippel, Kai-Florian, Barkowsky and Freksa [111] argued that qualitatively constructed maps lend themselves well to interactions with Geographic Information Systems. Kuehne and Forbus[112] suggested that qualitative representations are needed for natural language semantics in various domains. Knauff [113] provides a review of the literature focusing on cognitive adequacy.

As shown above, using qualitative methods to capture space and spatial relations within ontologies align well for capturing a perspective on space in line with human intuition. However, in specific domains, such as ontologies for geographic observation data, the need for metrical representations of space is necessary and implemented [86].

One of the more well-known implementations of a qualitative representation and reasoning is the Region Connection Calculus (RCC) [113]. RCC is a form of characterization of the mereotopology often used in foundational ontologies. When applied to two-dimensional areas, this calculus represents spatial configurations by describing convex regions and a small set of spatial relations which can be applied to those regions.

Region Connection Calculus 8 (RCC-8) [114] is a well-known and cited approach for representing the relationship between two regions in Euclidean space or in a topological space. There are eight possible relations, including disconnected, externally connected, partially overlapping, etc. However, RCC8 only addresses these relationships in two-dimensional space. There have been approaches that have tried to extend this into a region connected calculus in three-dimensional space while addressing occlusions [115]. There have also been approaches to develop calculi for spatial relations. FlipFlop calculus [116] describes the position of one point (the referent) in a plane with respect to two other points (the origin and the relatum). Single Cross Calculus (SCC) [117] is a ternary calculus that describes the direction of a point (C - the referent) with respect to a second point (B - the relatum) as seen from a third point (A - the origin) in a plane. Double Cross Calculus (DCC) [117] extends SCC by allowing one to also determine the relative location of point A with respect to point B (in addition to point B with respect to point A as in SCC). Coarse-grained Dipole Relation Algebra [118] describes the orientation relation between two dipoles (an oriented line segment as determined by a start and end point). Oriented Point Relation Algebra (OPRA) [119] relates two oriented points (a point in a plane with an additional direction parameter) and describes their relative orientation towards each other. All of these approaches, apart from RCC8, focus on points and lines as opposed to regions. Also, despite the large variety of qualitative spatial calculi, the amount of applications employing qualitative spatial reasoning techniques is comparatively small [120]. The combination of RCC constructs with other relations, such as proximity or time-related relationships, have also been explored– though many of these approaches have led to much computational complexity [121].

There are many other families of spatial calculi which address other conceptualizations of space that differ from the RCC approach. These include exploration of orientation and direction [117, 119] and of relative movement [122]. All propose reasoning approaches similar to RCC, but don't involve geometric calculations. A qualitative spatial reasoning summary is given by [123], while [97] describes spatial information approaches found in foundational ontologies.

In the next section, we will explore some ways that spatial information can be characterized.

2.3.3. Different Dimensions of State (Spatial) Representation

Moratz [124] identified three reference systems for spatial knowledge; intrinsic, relative, and absolute. Each occurs from three different perspectives depending on whether the agent is performing the actions, the agent is observing the actions, or a third entity is observing the action. This approach, though created for dialogue systems, will be explained in terms of the manufacturing domain, as this is the initial focus of this thesis.

In an *intrinsic reference systems*, the relative position of one object (which is referred to as the *referent*) to a second object (which is referred to as the *relatum*) is characterized by referencing the relatum's intrinsic properties, which could include its *front* or *back*. In a situation where a part (the referent) is placed in the front of a machine (the relatum), the part could be clearly identified by referencing the machine's front as the reference system's origin. In such a situation, the performer's or observer's positions are not relevant to identify the object. Although, the performer's or observer's left or right, might also represent the origins in intrinsic reference systems in statements such as, "The part is to the left of you." In such cases, no further entity is needed.

Humans employing *relative reference systems* use the third party's position as origin. In this case, the part (the referent) might be placed to the right of the machine (the relatum) from the performer's, the observer's, or a different entity's point of view (origin). All three may result in different "rights." Here, the machine's front and back are not relevant.

In *absolute reference systems*, the Earth's cardinal directions, such as *east* and *west,* serve as directions. A third entity and intrinsic features are not used as reference. Thus, the part might be situated to the west of the performer, the observer, or the machine.

As another perspective, Bateman and Farrar [125] described four high-level requirements that are needed for any spatial representation. These high-level requirements include:

- **Requirement 1:** A selection of an appropriate granular partition of the world that located entities in the world compared to other entities (e.g., identifying a chair, driveway);
- **Requirement 2:** A selection of a space region formalization that makes possible relevant spatial relationships;
- **Requirement 3:** A selection of an appropriate partition over the space region (e.g., RCC8, qualitative distance, cardinal direction);
- **Requirement 4:** The identification of an entity's location with respect to the selected space region description.

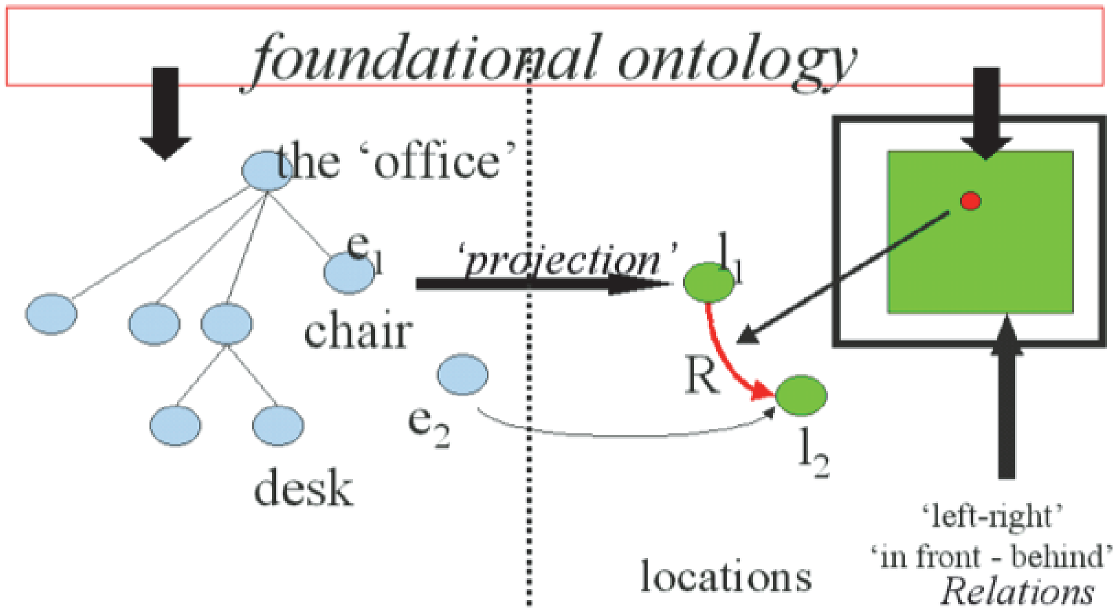


Figure 16: Qualitative Entity Location [126]

Figure 16 shows this graphically [126]. In this example, there is an office environment where we are characterizing the location of objects in various ways, as may be needed by a robotic system to identify the location of the object.

We could represent the office environment at different levels of granularity (Requirement #1 above). We could refer to the office as a single entity or by the objects that are in the office (e.g., chairs, desk), or at an extremely, by the molecules that make up the items in the office. The granularity that is appropriate here is the specification of the objects in the office, which is decomposed as blue circles shown in Figure 16.

The space region is decomposed according to a selected specification involving spatial relations (e.g., in front of, behind, left, right). This, represented by the green inner box, is a parameterized set of relations (R) for describing a particular location scheme (Requirement #2 above).

We can then map the objects in the world to locations in the world, as shown by the arrow from the chair (e_1) to the location in space (l_1) where the chair is located (requirement #3 above). If we then want to locate some additional object that was not in our original hierarchy (e_2), we can select a relation (R_1) from the space region in order to capture that new object's location (l_2). We can then use descriptions such as "e_2 is to the right of the chair" or "e_2 is near the chair," depending on how the space region is constructed (requirement #4 above).

Bateman ended up using a variation of the DOLCE ontology, but there is no mention in the literature about the detailed spatial relations that were developed as part of this effort apart from the high-level requirements.

2.3.4. Uses of Spatial Language in Various Applications

There have been applications of spatial knowledge representation is a number of fields, but most of them focus on dialogue-based approaches. The fields of situated robotics and geographic information science have achieved significant results. In situated robotics tasks, which require interaction with humans, the people that are interacting must have the ability to communicate about spatial actions and relationships. It is not as simple as translating a spatial relation into an abstract semantic knowledge base because the syntax and semantics must be related to the robot's sensor system and actuators. The issue of tying the representation to components and modules of the kind described in [127] therefore play a pivotal role.

The discussion above that representations be tied to robotic sensory algorithms and to actuator movement has led to the focus on numerically-based and approximate spatial representations. There has been a lot of interest in the literature in the area of field potentials. These are abstractions that capture the probability that some specified qualitative spatial representation is deemed to be appropriate at a given point in the potential field. As an example, in the expression "the block is to the left of the device" might result in a field potential indicating a very high confidence for a block being along a 90 degree clockwise axis to a specified orientation, with gradually lessening values as the point moves further away from this axis. A large number of spatial expressions have focused on this approach [128, 129] [130]. Robotics work has also addressed the challenge of using embodied spatial language more effectively primarily because the robots themselves are embodied agents with direct environmental sensory input [131-133].

In GIS systems, when the need arises to enable human–computer interaction, one must find correspondences between the system's internal representation of space and human-understandable representations. The mappings between abstract, internal spatial representations, and natural language expressions have been well explored in the literature [134-137] to provide GIS-compatible semantics for those expressions.

2.4. Conclusion

In this chapter, we have presented the state-of-the-art research in the areas of intention recognition, state/activity recognition, and state representation. All three areas play a vital role in the novel research described in the remainder of this thesis.

We have categorized intention recognition into logic-based approaches (which include deductive and abductive techniques), probabilistic approaches, and case-based approaches. Deductive approaches give definitive, provable results, but it is very rare in intention recognition that enough information (observations) is provided to be able to only use this approach. Abductive approaches provide reasonable explanations when the intention is not deductive, but mechanisms are need to choose between the various possible explanations.

Probabilistic approaches use well-tested theories to assign likelihoods to various intentions, but determining the input probabilities or belief to use these approaches can be more of an art than a science. Case-based approaches leverage previous instances of the same or similar intentions to try to match to what is being observed in the environment, but this starts to fall apart if there are no previous cases that match the current observations. We have also explored BDI approaches, which provide a mechanism for separating the activity of selecting a plan from the execution of currently active plans. However, BDI approaches lack mechanisms to learn from past behavior and the architecture doesn't have any forward planning.

We have explored metrics to measure the performance of intention recognition systems. Surprisingly, the literature is very light on these metrics. The one set of metrics that were found, namely applicability, grounding, and completeness, was described in terms of manufacturing-based intention recognition although it was originally applied to dialogue understanding systems.

We have then discussed the state-of-the-art research in activity and state recognition. A key component of this thesis is that fact that we use state information as the basis for recognizing intentions as opposed to using activity information as is predominantly performed in the literature. We showed two evaluation efforts, the Solutions in Perception Challenge and DARPA's Mind's Eye program, to represent the state-of-the-art performance in state and activity recognition, respectively. These two evaluations were purposely chosen because they are unbiased, independent evaluation which occurred to assess the performance of the developed systems. In many other evaluations, the developers evaluated themselves, often with situations and scenarios that favored their systems.

While it is impossible to perform a direct comparison between these two evaluations, these examples provide a sampling of the type of performance that can be expected by each type of technology. Based on this analysis, it appears that state recognition systems are approximately 20% (80% vs. 60%) more accurate than the activity recognition systems. This shows the benefit in using state information as the input for performing intention recognition as opposed to using activity information.

Finally, we show the various ways that state and spatial information have been represented in ontologies. This ranges from very high-level abstract representations in foundational ontologies to more specific representations in domain-specific ontologies. We also make the case for using qualitative methods for capturing a perspective on space more in line with human intuition. We look at various qualitative representation approaches, with an emphasis on RCC8, which serves as the basis for much of the work described in this thesis.

The approach described in this thesis uses what we believe to be the best aspects of all of the intention recognition approaches. It is primarily an abduction-based approach since it is trying to determine intentions based on a series of observations that don't definitively point to a single intention. In addition, it uses deduction to determine spatial relations that are true in the world based on primitive RCC8 relations. It applies a probabilistic approach by applying Bayesian techniques to assign probabilities to possible intentions. It also uses case-based reasoning approaches by predefining, in an ontology, the possible intentions that can occur in the world and the series of states that compose them, and then matches the observed states to those in the ontology.

The primary contribution of this work is the paradigm of using state information, and the sequence of the observed states, to infer intentions. For this we use qualitative representations, specifically RCC8 plus some cardinality relations, to represent the state of the world. We then map these sequences of states to those that are predefined in the ontology to infer intentions. As shown in this section, because state detection has shown to be more accurate that activity detection, this would therefore imply that the results of the intention recognition that use states as opposed to activities should be correspondingly more accurate.

The rest of this thesis is organized as follows:

- The component of this research applied to state representation is discussed in Chapter 3;
- The component of this research applied to intention recognition based on state representations is discussed in Chapter 4;
- The system architecture that was used for the experiment is described in Chapter 5;
- The performance of these approaches applied to the manufacturing kitting domain is discussed in Chapter 6.

3. State Representation and Reasoning

In this chapter, we describe an approach that uses RCC8 relations to model state information based on the relative position of objects in the environment [138]. We extend RCC8, which was initially developed for a two-dimensional space, into a three-dimensional space by applying it along all three axial planes (xy-plane, yz-plane, xz-plane) [139]. The frame of reference will be with respect to the reference object (e.g., a worktable), with the z-dimension pointing straight upwards and the primary axes of the object extending in the x- and y- dimension (with detailed orientation specific to the application). Each of the high-level state relationships (to be discusses later in the thesis) will have a set of logical rules that that are based upon the truth values of the RCC8 relations. By basing the state information on easily observable spatial relations, sensor systems will be able to more easily characterize the environment.

The state of objects in the real world can be determined by using different types of sensors, including cameras, laser range finders, and depth sensors. For the purpose of this work, we assume a Kinect-like sensor[6], which includes a 3-D depth sensor and a Red-Green-Blue (RGB) camera. In the experiments described in [81] showing the accuracy of state detection algorithms, a Kinect sensor was used. The placement of the Kinect sensor(s) to ensure maximum coverage of the work area is very important, and this placement will be specific to the individual application. This thesis does not focus on the types or placements of the sensors. Instead, it focuses on how intentions can be inferred from the output of the sensors.

3.1. The State Relation Ontology

In this section, describe our approach to representing state relations and how this information is captured in an ontology. An ontology-based approach was used through this thesis for the following reasons:

- Ontologies tend to be very expressive in what they can capture (including detailed descriptions of manufacturing objects and states);
- There are existing manufacturing ontologies that could be leveraged for this work;
- An IEEE Ontologies for Robotics and Automation Standards group was forming in which this work could be proposed.

[6] http://support.xbox.com/en-US/xbox-360/kinect/kinect-sensor-components

3.1.1. RCC8 Approach

Based upon the fact that RCC8 was the most mature spatial reasoning approach available in the literature that could handle solid region, it was chosen for this work.

As mentioned in the previous chapter, RCC8 abstractly describes regions in Euclidean or topological space by their relations to each other. RCC8 consists of eight basic relations that are possible between any two regions:

- Disconnected (DC);
- Externally Connected (EC);
- Equal (EQ);
- Partially Overlapping (PO);
- Tangential Proper Part (TPP);
- Non-Tangential Proper Part (NTPP);
- Tangential Proper Part Inverse (TPPi);
- Non-Tangential Proper Part Inverse (NTPPi).

These are shown pictorially in Figure 17.

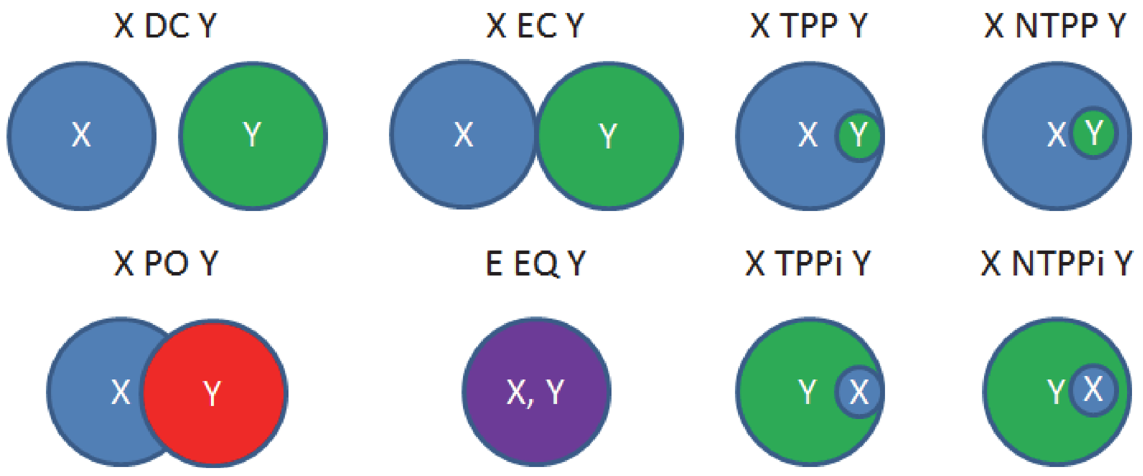


Figure 17: RCC8 Relations (Credit: http://en.wikipedia.org/wiki/RCC8)

RCC8 was created to model the relationships between two regions in two dimensions. In many domains, these relations need to be modeled in all three dimensions. As such, every pair of objects has a RCC8 relation in all three dimensions. To address this, we are prepending an xy-, yz- or xz- before each of the RCC8 relations to denote axial planes. For example, to represent the RCC8 relations in the xy-plane, the nomenclature would be:

- xy-DC;
- xy-EC;
- xy-EQ;
- xy-PO;
- xy-TPP;
- xy-NTPP;
- xy-TPPi;
- xy-NTPP.

Similar nomenclature would be used in the yz- and xz- dimensions.

The combination of all 24 RCC relations (eight per plane) starts to describe the spatial relations between any two objects in the scene. However, more information is needed to represent the cardinal direction between any two objects. For example, to state that a worktable is empty (worktable-empty(wtable)), one needs to state that there is nothing on top of it. For this approach, we use the reference frame attached to a fixed point of reference (in this case, the worktable) with the positive z-direction pointing away from the gravitational center. Using this approach, we can start to model this state by saying that:

$$\text{yz-EC}(\textit{wtable, obj1}) \tag{3}$$

This intuitively means *obj1* is externally connected to the worktable in the z-dimension. However, this is not sufficient because *obj1* could be either on top of or below the worktable, both which would make Equation 3 evaluate to true. To address this, we need to represent directionality. We do this using the following Boolean operators:

$$\text{greater-x}(A,B) \tag{4}$$

$$\text{smaller-x}(A,B) \tag{5}$$

$$\text{greater-y}(A,B) \tag{6}$$

$$\text{smaller-y}(A,B) \tag{7}$$

$$\text{greater-z}(A,B) \tag{8}$$

$$\text{smaller-z}(A,B) \tag{9}$$

This intuitively means, in Equation 5, that the edge of the bounding box of Object A is greater than (in the x-dimension in the defined frame of reference) the edge of the bounding box of Object B with respect to the separation plane..

3.1.2. Defining More Complex Relations

There are undoubtedly many other relationships that may be needed in the future to fully describe a scene of interest. These could include absolute locations [140] and orientations of objects (x, y, z, roll, pitch, yaw) and relative distance (closer, farther, etc.). However, the spatial relations defined in this chapter are sufficient for describing the manufacturing kitting example used later in this thesis.

From these RCC8 spatial relations, we can define more complex spatial relations such as the ones below:

- **Contained-In**- an object is enclosed in a second object from all sides;
- **Not-Contained-In**- an object is not enclosed in a second object from all sides;
- **Partially-In**: an object is inside of another object in at least two dimensions but not fully contained in;
- **In–Contact-With**- an object is touching at least one side of another object, but not contained within it (i.e., touching outer edges);
- **On-Top-Of**- an object is in the same x-y region as second object and is greater (in the z-dimension) than that of a second object;
- **Under**- an object is in the same x-y region as second object and is less than (in the z-dimension) than that of a second object.

And from these, we can define composite spatial relationships such as:

- **Under-And-In-Contact-With**- an object is both under and in contact with a second object;
- **Partially-In-And-In-Contact-With**- an object is inside of another object in at least two dimensions and touching the object in at least one dimension.

Below we formalize these spatial relationships by defining them using the RCC8 state representation [6]. In natural language, Equation 10 below states that Object 1 (*obj1*) is contained in object 2 (*obj2*) if obj1 is tangentially or non-tangentially a proper part of Object 2 in the x, y, and z-dimension. One can logically envision this by drawing two convex figures, and the first convex figure is completely inside of the second convex figure in all three dimensions, with it touching or not touching the second convex hull in any of the three dimensions.

Contained-In(*obj1*, *obj2*) $\rightarrow$
(xy-TPP(*obj1*, *obj2*) ∨ xy-NTPP(*obj1*, *obj2*)) ∧
(yz-TPP(*obj1*, *obj2*) ∨ yz-NTPP(*obj1*, *obj2*)) ∧
(xz-TPP(*obj1*, *obj2*) ∨ xz-NTPP(*obj1*, *obj2*)) (10)

Not-Contained-In(obj1, obj2) $\rightarrow$
¬Contained-In(obj1, obj2) (11)

Partially-In(obj1, obj2) $\rightarrow$
((xy-TPP(obj1, obj2) ∨ xy-NTPP(obj1, obj2)) ∧ (yz-PO (obj1, obj2) ∧ xz-PO(obj1, obj2)) ∨ ((xz-TPP(obj1, obj2) ∨ xz-NTPP(obj1, obj2)) ∧ (yz-PO (obj1, obj2) ∧ xy-PO(obj1, obj2)) ∨ ((yz-TPP(obj1, obj2) ∨ yz-NTPP(obj1, obj2)) ∧ (xz-PO (obj1, obj2) ∧ xy-PO(obj1, obj2)) (12)

In-Contact-With(obj1, obj2) $\rightarrow$
xy-EC(obj1, obj2) ∨ yz-EC(obj1, obj2) ∨ xz-EC(obj1, obj2) (13)

On-Top-Of(obj1, obj2) $\rightarrow$
greater-z(obj1, obj2) ∧ (xy-EQ(obj1, obj2) ∨ xy-NTPP(obj1, obj2) ∨ xy-TPP(obj1, obj2) ∨ xy-PO(obj1, obj2) ∨ xy-NTPPi(obj1, obj2) ∨ xy-TPPi(obj1, obj2)) (14)

Under(obj1, obj2) $\rightarrow$
smaller-z(obj1, obj2) ∧ (xy-EC(obj1, obj2) ∨ xy-NTPP(obj1, obj2) ∨ xy-TPP(obj1, obj2) ∨ xy-PO(obj1, obj2) ∨ xy-NTPPi(obj1, obj2) ∨ xy-TPPi(obj1, obj2)) (15)

Under-And-In-Contact-With(obj1, obj2) $\rightarrow$
Under(obj1, obj2) ∧ In-Contact-With (obj1, obj2) (16)

Partially-In-And-In-Contact-With(obj1, obj2) $\rightarrow$
Partially-In(obj1, obj2) ∧ In-With-Contact(obj1, obj2) (17)

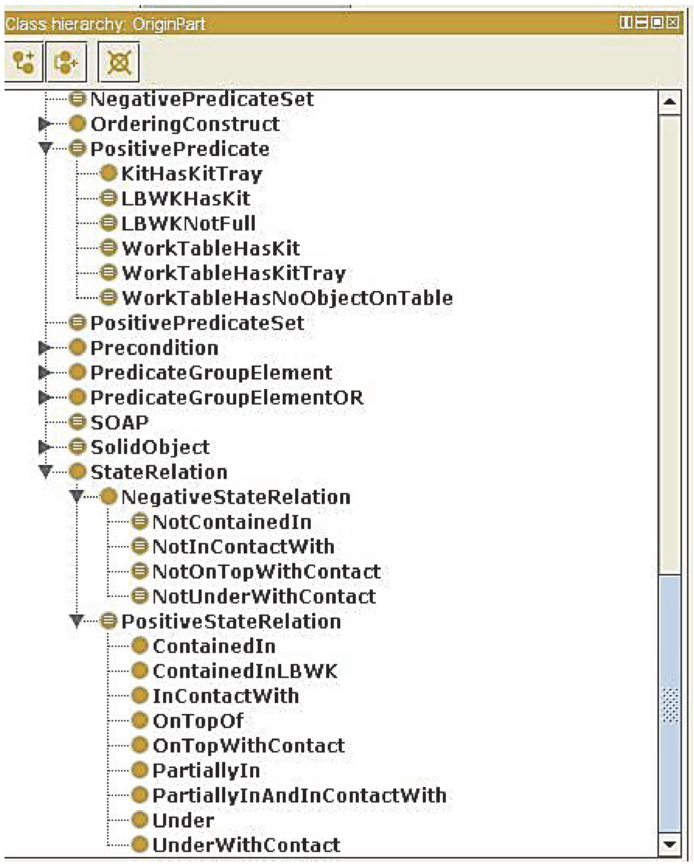
These spatial relationships will be used later in the thesis to define two manufacturing kitting intentions.

3.1.3. State Relation Ontology Constructs

The spatial relations above are represented as subclasses of the RelativeLocation class which is a subtype of the PhysicalLocation class which is a subtype of the DataThing class in the ontology (to be discussed in more detail later in the thesis). DataThings are abstract, non-tangible things. There are three types of spatial relations, each described below:

- **RCC8_Relations**- These are the 24 RCC8 relations and the six cardinality direction operators described earlier in this chapter. These classes are not any further defined but can be instantiated as occurrences of them are found in the environment;
- **Intermediate_State_Relations**- These are intermediate level state relations that can be inferred from the combination of RCC8 and cardinal direction relations. The examples above, such as *Under* and *In-Contact-With*, are examples of intermediate state relations. The logical expression based on the RCC8 and cardinal direction relations (as shown in Equations 10-17) which are evaluated to determine that truth-value of the state relations are represented within the Equivalent Classes in the ontology. The information is exported from the ontology during run-time and converted into code that is evaluated as new perception data is presented to the system;
- **Predicates**- These are domain-specific states that are of interest to the current intention (or set of intentions) being evaluated. For example, in the manufacturing example to be discussed later in the thesis, one state of interest is that the worktable is empty. This is true if the worktable is not *Under-And-In-Contact-With* any object in the environment. The truth-value of predicates can be determined through the logical combination of intermediate state relations. As with the intermediate state relations, these are captured using the equivalent classes in the ontology.

A screen shot of the class hierarchy of the state relations in the ontology can be seen in Figure 18.


Class hierarchy: OriginPart

```
NegativePredicateSet
OrderingConstruct
PositivePredicate
    KitHasKitTray
    LBWKHasKit
    LBWKNotFull
    WorkTableHasKit
    WorkTableHasKitTray
    WorkTableHasNoObjectOnTable
PositivePredicateSet
Precondition
PredicateGroupElement
PredicateGroupElementOR
SOAP
SolidObject
StateRelation
    NegativeStateRelation
        NotContainedIn
        NotInContactWith
        NotOnTopWithContact
        NotUnderWithContact
    PositiveStateRelation
        ContainedIn
        ContainedInLBWK
        InContactWith
        OnTopOf
        OnTopWithContact
        PartiallyIn
        PartiallyInAndInContactWith
        Under
        UnderWithContact
```

Figure 18: State Relation Ontology

3.2. The State Relation Algorithms

Based on the ontological constructs described in the previous section, the state relation algorithms first project objects in the environment onto the three axial planes, then determine if the projections overlap to establish the RCC8 relations in each plane, and finally determine the intermediate state relations based on the RCC8 relations. This process is described in this section.

3.2.1. Projecting the Objects onto the Planes and Determining the Contour

The RCC-8 spatial relations in each axial plane can be determined by projecting the point cloud representing the three-dimension solid object onto each of the axial planes. The first step in this process is to determine the contour of each shape projected onto each of the three axial planes. We use as input into this process the set of 2D vertices that are projected on each plane as well as the list of nearest neighbor vertices. Figure 19a shows a schematic drawing explaining the first part of algorithm. Open circles represent the starting point (k_0) and the last point (k_2) of a closed boundary loop organized in the clockwise direction. Black dots correspond to vertices on a boundary while gray dot represent vertices from inside a 2D region limited by a loop. The vertex k_0 is directly connected with four vertices (nearest neighbors): k_1, ..., k_4. Vertex k_1 is set as ktry, which is an initial candidate for the next point on the loop. Then, remaining nearest neighbors are consecutively checked. Since the vector product of two vectors, ($\mathbf{P}(k_{try}) - \mathbf{P}(k_0)$) x ($\mathbf{P}(k_2)$ - $\mathbf{P}(k_0)$), points behind the 2D plane, k_2 is rejected. The next candidate for k_{try} is k_3 and since this time the cross product is directed upward, this point is accepted. The last vertex k_4 is rejected in the same way as k_2 before. Once this process is complete, the complete contour of the object on the specified projection is determined.

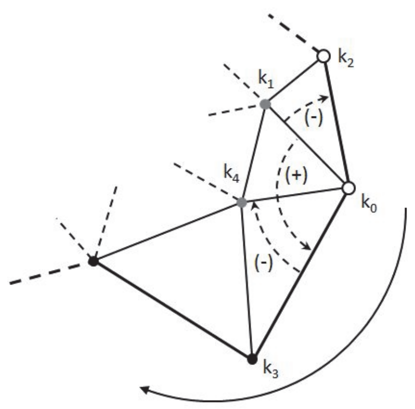


Figure 18a: Build Projection Contour

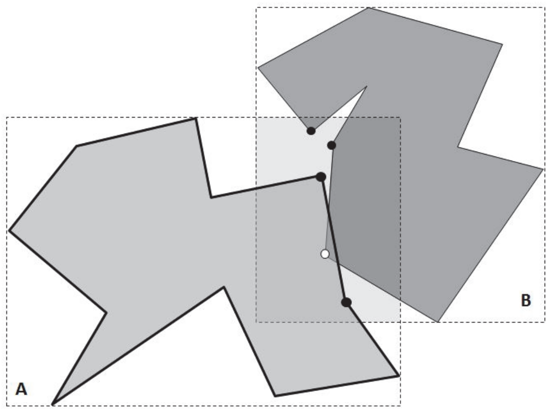
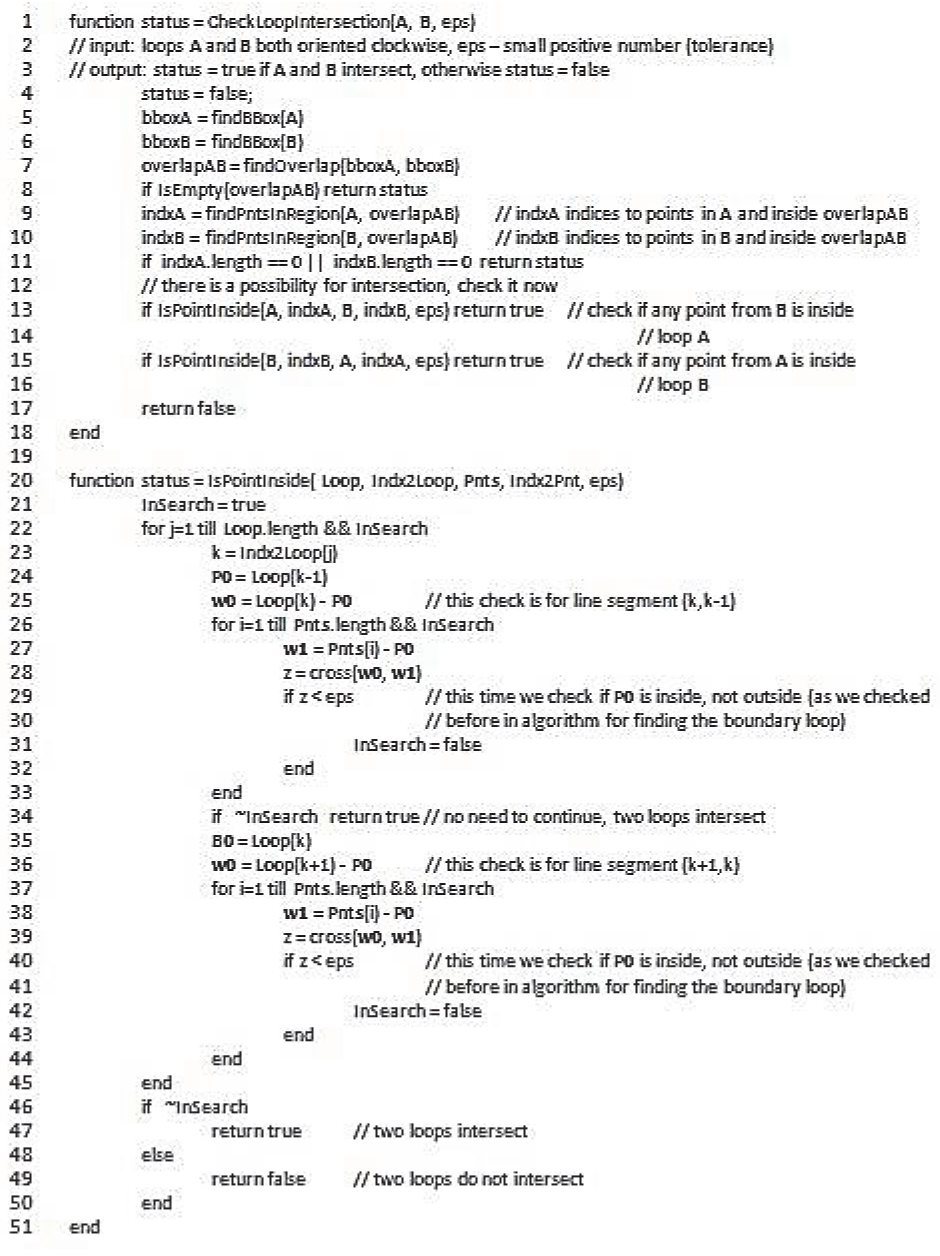


Figure 18b: Determine Contour Overlap

Figure 19: Projection Objects onto Places and Determining Overlap

3.2.2. Determining When Object Projections Overlap

The second step of the process is to determine if two projections are touching or overlapping each other, as shown in Figure 19b. Two contours (**A** and **B**) intersect each other only when there is at least one line segment in **A** and at least one line segment in **B** which intersect. Necessary but not sufficient condition for intersection is that bounding boxes of both loops overlap and each loop has at least one vertex in overlapping region. In the drawing, open circle indicates the vertex belonging to loop B which protrudes into 2D area limited by loop A. Pseudo-code for this process is shown in Figure 20. Using this information, we can determine the overlap between the projections of two objects, allowing us to determine the RCC relations.



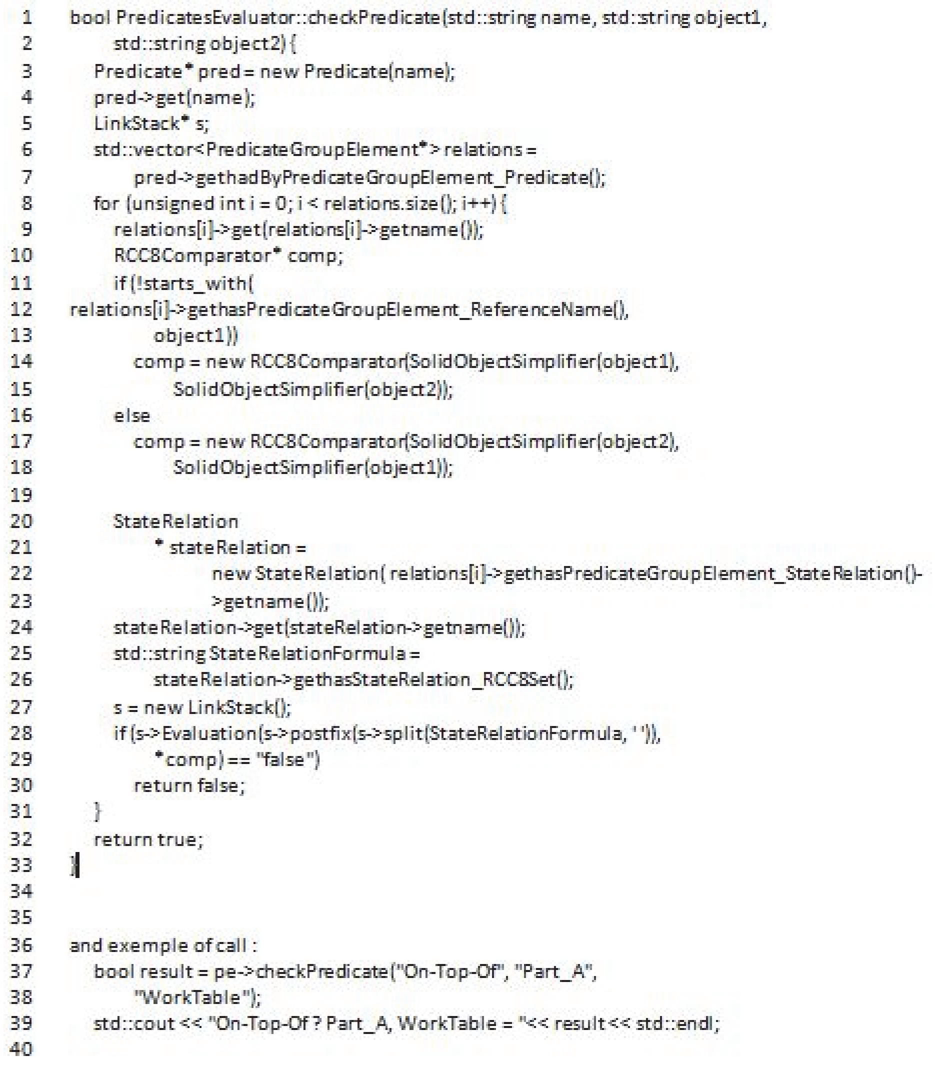
```
function status = CheckLoopIntersection(A, B, eps)
// input: loops A and B both oriented clockwise, eps – small positive number (tolerance)
// output: status = true if A and B intersect, otherwise status = false
        status = false;
        bboxA = findBBox(A)
        bboxB = findBBox(B)
        overlapAB = findOverlap(bboxA, bboxB)
        if IsEmpty(overlapAB) return status
        indxA = findPntsInRegion(A, overlapAB)      // indxA indices to points in A and inside overlapAB
        indxB = findPntsInRegion(B, overlapAB)      // indxB indices to points in B and inside overlapAB
        if indxA.length == 0 || indxB.length == 0 return status
        // there is a possibility for intersection, check it now
        if IsPointInside(A, indxA, B, indxB, eps) return true    // check if any point from B is inside
                                                                        // loop A
        if IsPointInside(B, indxB, A, indxA, eps) return true    // check if any point from A is inside
                                                                        // loop B
        return false
end

function status = IsPointInside( Loop, Indx2Loop, Pnts, Indx2Pnt, eps)
        InSearch = true
        for j=1 till Loop.length && InSearch
                k = Indx2Loop(j)
                P0 = Loop(k-1)
                w0 = Loop(k) - P0           // this check is for line segment (k,k-1)
                for i=1 till Pnts.length && InSearch
                        w1 = Pnts(i) - P0
                        z = cross(w0, w1)
                        if z < eps           // this time we check if P0 is inside, not outside (as we checked
                                             // before in algorithm for finding the boundary loop)
                                InSearch = false
                        end
                end
                if ~InSearch return true // no need to continue, two loops intersect
                B0 = Loop(k)
                w0 = Loop(k+1) - P0         // this check is for line segment (k+1,k)
                for i=1 till Pnts.length && InSearch
                        w1 = Pnts(i) - P0
                        z = cross(w0, w1)
                        if z < eps           // this time we check if P0 is inside, not outside (as we checked
                                             // before in algorithm for finding the boundary loop)
                                InSearch = false
                        end
                end
        end
        if ~InSearch
                return true          // two loops intersect
        else
                return false         // two loops do not intersect
        end
end
```

Figure 20: Pseudcode for Computing Overlaps

3.2.3. Inferring Intermediate State Relations from RCC8 Relations

Once the RCC8 relations are determined, a separate set of code is used to determine if any intermediate spatial relation can be evaluated as true. At this point, all pairwise relations between objects in the environment have been determined using the RCC8 relations. Intermediate state relations are based upon the RCC8 relations, and this relationship is captured explicitly in the ontology. Examples of these relationships were shown in Equations 10-17.

The code presented in Figure 21 shows an example of the On-Top-Of intermediate state relation being evaluated. In lines 36-39, the On-Top-Of state relation is being passed to the checkPredicate function along with two parameters: Part_A and Worktable. The checkPredicate function in lines 1-33 take the name of the predicate and search the ontology for the logical expression (composed of RCC8 relations) that represents the intermediate state relation. Once this is found, it uses to two object parameters to retrieved the truth value of the respective RCC8 relations and plugs them into the intermediate state relation expression to determine the truth value of the overall intermediate state relation.



```
1   bool PredicatesEvaluator::checkPredicate(std::string name, std::string object1,
2         std::string object2) {
3     Predicate* pred = new Predicate(name);
4     pred->get(name);
5     LinkStack* s;
6     std::vector<PredicateGroupElement*> relations =
7           pred->gethadByPredicateGroupElement_Predicate();
8     for (unsigned int i = 0; i < relations.size(); i++) {
9         relations[i]->get(relations[i]->getname());
10        RCC8Comparator* comp;
11        if (!starts_with(
12  relations[i]->gethasPredicateGroupElement_ReferenceName(),
13              object1))
14            comp = new RCC8Comparator(SolidObjectSimplifier(object1),
15                SolidObjectSimplifier(object2));
16        else
17            comp = new RCC8Comparator(SolidObjectSimplifier(object2),
18                SolidObjectSimplifier(object1));
19
20        StateRelation
21              * stateRelation =
22                    new StateRelation( relations[i]->gethasPredicateGroupElement_StateRelation()-
23                    >getname());
24        stateRelation->get(stateRelation->getname());
25        std::string StateRelationFormula =
26              stateRelation->gethasStateRelation_RCC8Set();
27        s = new LinkStack();
28        if (s->Evaluation(s->postfix(s->split(StateRelationFormula, ' ')),
29              *comp) == "false")
30            return false;
31    }
32    return true;
33  }
34
35
36  and exemple of call :
37    bool result = pe->checkPredicate("On-Top-Of", "Part_A",
38          "WorkTable");
39    std::cout << "On-Top-Of ? Part_A, WorkTable = "<< result << std::endl;
40
```

Figure 21: Predicate Evaluation Pseudo-code

The algorithms presented in this section have an execution time proportional to the number of points in the object projection's contour times the total number of vertices in the object's projection. This is a lower bound for the best 2D convex hull algorithms (such as Chan's algorithm, http://en.wikipedia.org/wiki/Convex_hull_algorithms). The execution time for the experiment presented later in this thesis is approximately 0.048 seconds for the object of interest.

3.3. The Manufacturing Kitting Ontology and Associated State Relations

Though we expect the approaches described in this chapter to be generic, we are initially applying them to the manufacturing domain to show their feasibility. In this section, we describe the manufacturing kitting domain, its associated ontology, the relevant manufacturing state relations.

3.3.1. Human-Robot Collaboration Manufacturing Scenarios

The manufacturing domain is rich for human-robot interaction, which would be greatly facilitated by an Intention Recognition Algorithm. In this section, we describe some scenarios in which humans and robot may collaborate to accomplish a task. Some of the these scenarios are adapted from other models [141] [142]. We will be focusing on one of these scenarios later in the thesis, but this section is provided to show the breadth of scenarios that would be applicable to approaches described in this thesis:

- **Pick-up and Delivery of Parts -** Consider a workcell during normal operation where a person carrying or pushing something, or a mobile robot/Automated Guided Vehicle (AGV), enters the workcell. These external agent(s) may be passing through, dropping off or retrieving parts, arriving to perform maintenance, or arriving to perform work at a station. The perception system must detect and track all moving objects in the open space, with the ability to resolve an accurate bounding shape but not necessarily to an articulated model. The perception system must determine the positions of the AGVs, the robots and the humans as well as identify their next steps and predict their intentions;
- **Bin Picking** – A robot arm, working independently while processing parts from a bin, has its bin changed by a person. The arrangement is fenceless, so a person can come close to the robot. The object could be a kit rather than a bin, or any part container. The perception system needs to perceive the person coming close, recognize the intent, stop the robot during the change, perceive that the change has taken place correctly, and restart the robot when conditions allow;
- **An AGV/Mobile Robot Working with Human(s) -** The USCAR study [142] briefly explains a current manufacturing scenario during automobile powertrain assembly where humans and AGVs share the same space. In the example, the workcell is "located adjacent to an aisle that is shared between pedestrian and

powered material handling vehicle traffic." AGVs are not typically behind fences and cordoned-off areas and must therefore be able to operate safely near humans and other equipment. The AGV may be used to carry the item or pick up the item and carry it or place it on top of another AGV. The AGV would follow the person to a new location, recognizing gestures or verbal command along the way, and determine the human's intent. At the new location, the AGV would either stop to allow the person to take the object or unload the item;

- **Human-Robot Assembly** - As shown in Figure 22, a robot and person are on either side of a table and collaborating on an assembly. In this case, the robot may pick up a part and hand it to the person. The perception system needs to track the part, the person's gesture, and the intentions. It needs to follow them through maintaining awareness of the changing space for safety, detect if the part is properly handled by the person and the robot, compute the new profile the interaction between person and robot, and identify when it is in the directed final configuration;

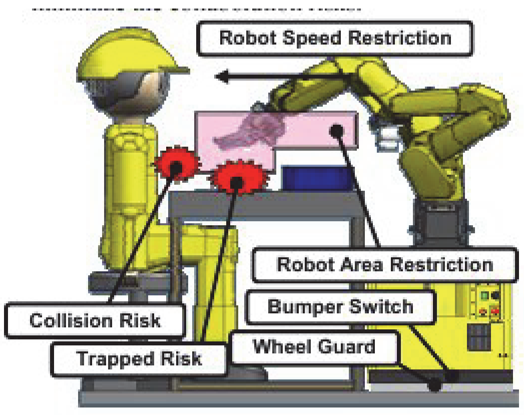


Figure 22: Joint Human-Robot Assembly

- **Kit Building:** The human takes empty kit trays out of a box containing empty kit trays. They take parts out of parts trays or parts bins and put them into kit trays. They take finished kits (kit trays with parts in them) and put them in a box of kits. A robot may be in charge of monitoring the action to determine the intention that the human is intending to perform, which could include which kit they are currently building. The robot may take steps to help the human, including replenishing parts that may not be available in the parts trays to complete the kit, or simply to stay a safe distance away from the likely subsequent actions that the human will perform. The robot may also track the actions and when the intention is determined, make sure the subsequent steps are sufficient for completing the kit (i.e., may sure that the human did not forget to put in a part).

3.3.2. Manufacturing Kitting Domain Description

We will initially focus on manufacturing kitting operations as described in the last scenario above and in further elaborated in [143]. Kitting is the process in which several different, but related items are placed into a container and supplied together as a single unit (kit) as shown in Figure 23. Kitting is often performed prior to final assembly in industrial assembly of manufactured products so all of the necessary parts are gathered in one location. Manufacturers utilize kitting due to its ability to provide cost savings including saving manufacturing or assembly space, reducing assembly workers' walking and searching times, and increasing line flexibility and balance.

In batch kitting, the kit's component parts may be staged in containers positioned in the workstation or may arrive on a conveyor. Component parts may be fixture (e.g., placed in compartments on trays) or may be in random orientations (e.g., placed in a large bin). In addition to the kit's component parts, the workstation usually contains a storage area for empty kit boxes as well as completed kits.

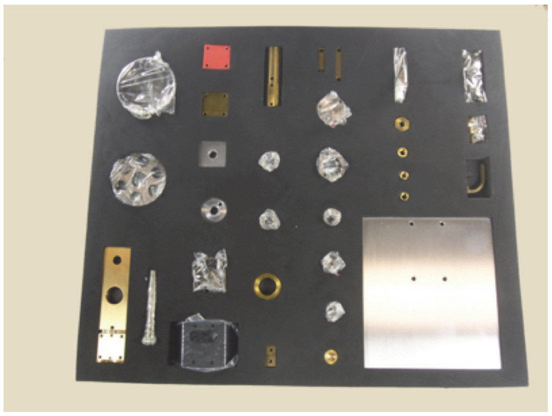


Figure 23: Example Kit (courtesy of LittleMachineShop.com)

Kitting has not yet been automated in many industries where automation may be feasible. Consequently, the cost of building kits is higher than it could be [143].

3.3.3. The Manufacturing Kitting Ontology

An industrial kitting ontology has been developed [144] which will serve as the basis for the Industrial Robotics Ontology as part of the IEEE Robotics and Automation Society's (RAS) Ontologies for Robotics and Automation (ORA) Standard Working Group[7]. The industrial kitting objects ontology is written in Web Ontology Language (OWL) [145]. It is included in Appendix A. Conceptually, the model is an object model. That is:

[7] https://ieee-sa.centraldesktop.com/p1872public/

- the model consists primarily of class definitions;
- a class defines a type of thing;
- classes have attributes ("elements" in XML schema language);
- the class definition gives the class (or data type for individual variables) of each attribute;
- some attributes may occur optionally or multiple times;
- some classes are derived from others; thus, there is a derivation hierarchy;
- a derived class has all the attributes of its parent plus, possibly, some of its own;
- if class B is derived from class A, then if the type of an attribute is class A, an instance of class B may be used as the value of the attribute;
- the model also uses primitive data types such as numbers and strings, and provides for defining specialized data types by putting constraints on primitive data types.

As shown in Table 1 below, the model has two top-level classes, **SolidObject** and **DataThing**, from which all other classes are derived. **SolidObject** models solid objects which are made of matter. **DataThing** models data. The level of indentation in the table indicates subclassing. For example, **KitTray** is derived from **SkuObject**, and **SkuObject** is derived from **SolidObject**. Items in *italics* following classes are names of data members of the class. Derived types inherit the data members of the parent. Each data member has a specific type not shown in the table. If the type of a data member has derived types, any of the derived types may be used.

The names of the OWL properties that give the data members shown in the table are formed from the data member name by adding the prefix *has*class_ where class is the class name. For example, the name of the ObjectProperty for the *SolidObjects* data member of a **WorkTable** is *hasWorkTable_SolidObjects*. For the XML representation, the prefixes are unnecessary and are not utilized.

Inverse properties are defined in the OWL version of the kitting workstation ontology for all of the **ObjectProperties**. The names of the inverse object properties are formed by changing the *has* at the beginning of the name to *hadBy* and reversing the order of the other two components of the name. For example, the inverse of *hasKit_KitTray* is *hadByKitTray_Kit*. The fact that two **ObjectProperties** are inverses is indicated by putting an InverseObjectProperties statement in the ontology.

In the table, (0) means zero or one of the data member may appear in an instance file, [0] means zero to many may appear, and [ ] means one to many may appear.

Each **SolidObject** has a native coordinate system conceptually fixed to the object. The native coordinate system of a **SolidObject** with a **BoxyShape**, for example, has its origin at the middle of the bottom of the object, its Z axis perpendicular to the bottom, and the X axis parallel to the longer horizontal edges of the object.

Each **SolidObject** *A* has at least one *PhysicalLocation* (the primary location). A *PhysicalLocation* is defined by giving a reference **SolidObject** *B* and information saying how the position of *A* is related to *B*. Two types of location are required for the operation of the kitting workstation. Relative locations, specifically the knowledge that one **SolidObject** is in or on another, are needed to support making logical plans for building kits. Mathematically precise locations are needed to support robot motion. The mathematical location, **PoseLocation**, gives the pose of the coordinate system of *A* in the coordinate system of *B*. The mathematical information consists of the location of the origin of *A*'s coordinate system and the directions of its Z and X axes. The mathematical location variety has subclasses representing that, in addition, *A* is in *B* (**PoseLocationIn**) or on *B* (**PoseLocationOn**). The subclasses of **RelativeLocation** are needed not only for logical planning, but also for cases when the relative location is known, but the mathematical information is not available. This occurs, for example when a **PartsBin** is being used, since by definition, the **Parts** in a **PartsBin** are located randomly.

All chains of location from **SolidObjects** to reference **SolidObjects** must end at a **KittingWorkstation** (which is the only class of **SolidObject** allowed to be located relative to itself).

For planning, it is assumed that **SolidObjects** do not move unless a command moves them. Also, if **SolidObject** *A* is in or on **SolidObject** *B* (so that the reference object for *A* is *B*), then if *B* is moved, the position of *A* relative to *B* is unchanged.

A **SolidObject** may be given multiple locations by using its **SecondaryLocation** data member. If multiple locations are used, they are expected to be logically and mathematically consistent.

The kitting ontology includes several subclasses of **SolidObject** that are formed from components that are **SolidObjects**. These are: **Kit**, **LargeBoxWithEmptyKitTrays**, and **LargeBoxWithKits**. Combined objects may come into existence or go out of existence dynamically when a kitting workstation is operating. For example, when all the kit trays in a **LargeBoxWithEmptyKitTrays** have been removed and put into kits, the **LargeBoxWithEmptyKitTrays** should go out of existence and the **LargeContainer** that was holding kit trays should have its location switched from its location relative to the **LargeBoxWithEmptyKitTrays** to the former location of the **LargeBoxWithEmptyKitTrays**.

Table 2: Kitting Workstation Class Hierarchy

SolidObject *PrimaryLocation SecondaryLocation[0]*
NoSkuObject *InternalShape(0) ExternalShape(0)*
EndEffector *Description Weight MaximumLoadWeight HeldObject(0)*
GripperEffector
VacuumEffector *CupDiameter Length*
VacuumEffectorMultiCup *ArrayNumber ArrayRadius*
VacuumEffectorSingleCup
EndEffectorChangingStation *Base EndEffectorHolder[ ]*
EndEffectorHolder *EndEffector(0)*
Human
Kit *DesignName KitTray Part[0] Slot[0] Finished*
KittingWorkstation *AngleUnit ChangingStation KitDesign[ ] LengthUnit OtherObstacle[0] Robot Sku[ ] WeightUnit*
LargeBoxWithEmptyKitTrays *LargeContainer KitTray[0]*
LargeBoxWithKits *LargeContainer Kit[0] KitDesign Capacity*
MechanicalComponent
Robot *Description EndEffector(0) MaximumLoadWeight WorkVolume*
WorkTable *ObjectOnTable[0]*
SkuObject *Sku*
KitTray *SerialNumber*
LargeContainer *SerialNumber*
Part *SerialNumber*
PartsVessel *SerialNumber PartSku PartQuantity Part[0]*
PartsBin
PartsTray
DataThing
BoxVolume *MaximumPoint MinimumPoint*
KitDesign *KitTraySku PartRefAndPose[ ]*
PartRefAndPose *Sku Point XAxis ZAxis*
PhysicalLocation *RefObject Timestamp(0)*
PoseLocation *Point ZAxis XAxis PositionStandardDeviation(0) OrientationStandardDeviation(0)*
PoseLocationIn
PoseLocationOn
PoseOnlyLocation
RelativeLocation *Description*
RelativeLocationIn
RelativeLocationOn
Point *X Y Z*
ShapeDesign *Description GraspPose*
ExternalShape *ModelFormatName ModelFileName ModelName(0)*
InternalShape
BoxyShape *Length Width Height HasTop*
CylindricalShape *Diameter Height HasTop*
Slot *PartRefAndPose Part*
StockKeepingUnit *Description InternalShape(0) ExternalShape(0) Weight EndEffector[0]*
Vector *I J K*

The kitting workstation ontology provides two methods of describing shape: use a shape defined in the ontology (an **InternalShape**), or use a shape described in a separate file (an **ExternalShape**). Currently, there are only two types of **InternalShape**: **BoxyShape** and **CylindricalShape**. An **ExternalShape** identifies the format and name of the file containing the shape. If that file contains more than one shape, the *ModelName* data member may be used to identify a particular one. Each **SolidObject** must have at least one of the two types of shape and may have both. If both types of shape are assigned, they should describe the same shape at different levels of detail.

In the kitting workstation ontology, there two principal subclasses of **SolidObject**: **SkuObject** and **NoSkuObject**. If the **SolidObject** is a **SkuObject**, it gets its shape from its SKU. If it is a **NoSkuObject**, it gets its shape directly.

3.3.3.1. Class Structure Diagram

This section provides figures containing diagrams of the structure of the principal classes in the kitting workstation ontology. All figures were generated by XMLSpy from the XML schema. In the figures:

- Each box contains a data member of the class;
- A dotted line around a box means the data member is optional (may occur zero times);
- A *0..∞* underneath a box means it may occur zero or more times, with no upper limit on the number of occurrences;
- A *1..∞* underneath a box means it must occur at least once, with no upper limit on the number of occurrences;
- The irregular octagons with a line and three dots indicate sets of data members added at different levels of the inheritance hierarchy.

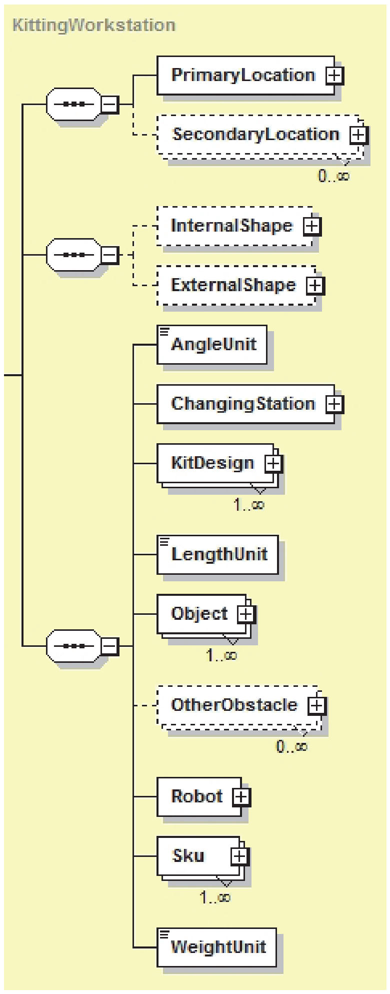


Figure 24: Kitting Workstation Model

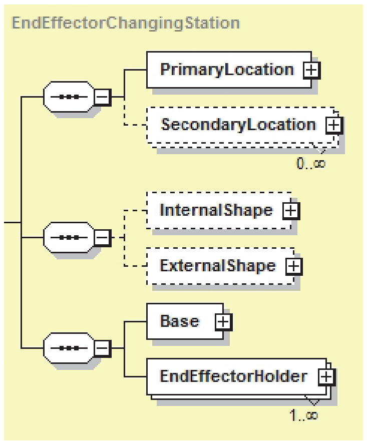


Figure 25: End Effector Changing Station Model

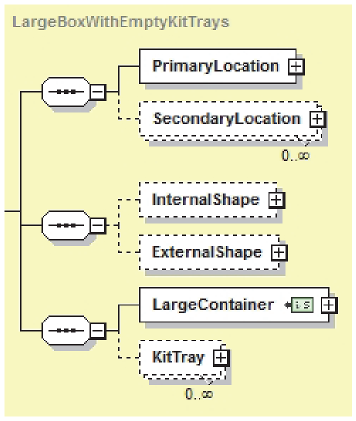


Figure 26: Large BoxWith Empty Kit Trays Model

The robot model is simple and does not currently have any kinematics or even any shape for the robot. It is likely that additional data members will be added in the future.

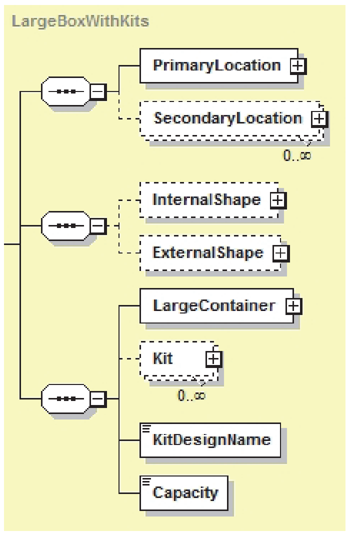


Figure 27: Large Box With Kits Model

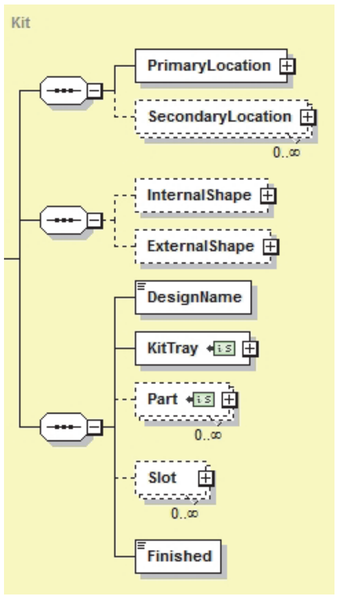


Figure 28: Kit Model

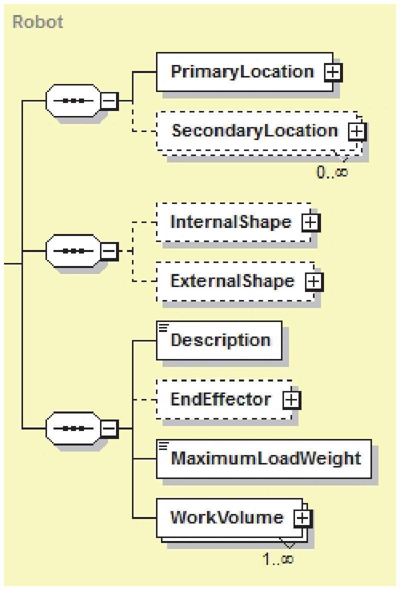


Figure 29: Robot Model

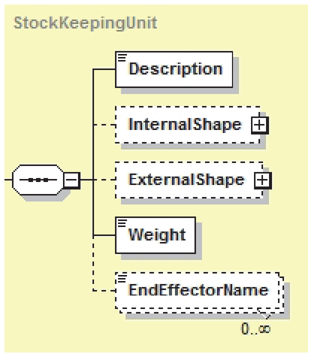


Figure 30: Stock Keeping Unit Model

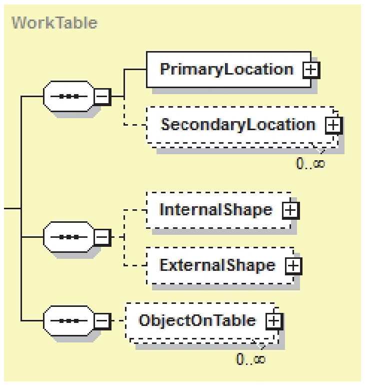


Figure 31: Work Table Model

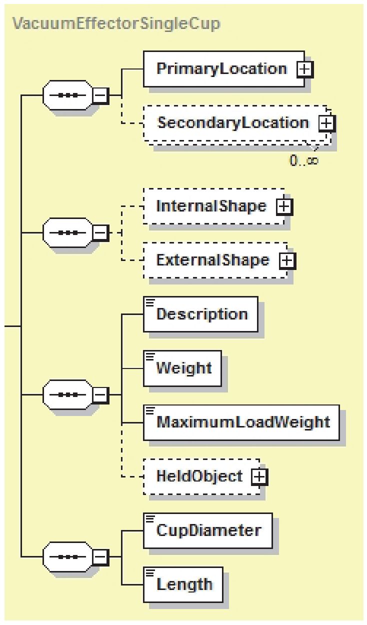


Figure 32: Vacuum Effector Single Cup Model

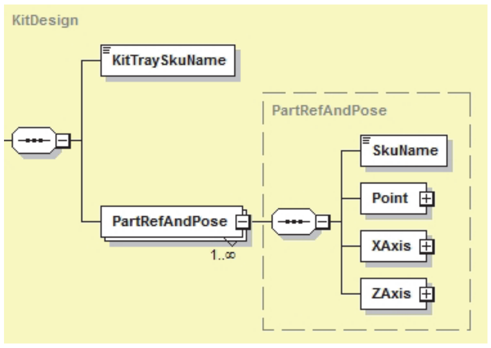


Figure 33: Kit Design Model

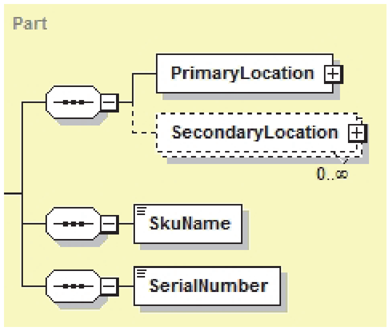


Figure 34: Part Model

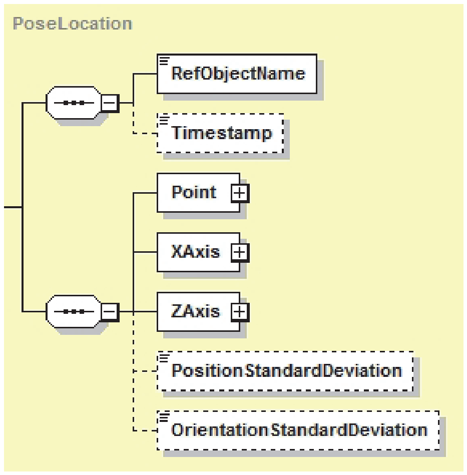


Figure 35: Pose Location Model

3.3.3.2. Data Types

The OWL language borrows most of the XSDL built-in datatypes. In addition, specialized OWL datatypes may be derived from the built-in datatypes by DatatypeDefinitions. These correspond to XSDL derivations of simple types by restricting built-in types. The OWL datatypes used in the kitting workstation ontology include:

- **AngleUnit**- The **angleUnit** may be one of “degree” or “radian.” It specifies that any property that represents angles will be expressed in **angleUnit** units. This is defined by a DatatypeDefinition;
- **Boolean**- A **boolean** is the XSDL xsd:boolean. Valid values for a **boolean** are true, false, 0, and 1, where 0 is equivalent to false, and 1 equivalent to true;
- **DateTime-** a **dateTime** is the XSDL xsd:dateTime. It represents the date and time in the form YYYY-MM-DDThh:mm:ss.sss, where YYYY is the year, MM the month, DD the day, T a literal T, hh the hour (00 to 23), mm the minutes, and ss.sss the seconds (from which the .sss may be removed or extended);
- **Decimal**- a **decimal** is the XSDL xsd:decimal. It represents a decimal number of arbitrary precision. The format of a **decimal** is a sequence of digits optionally preceded by a sign ("+" or "-") and optionally containing a period. The value may start or end with a period. If the fractional part is 0 then the period and trailing zeros may be omitted. Leading and trailing zeros are permitted, but they are not considered significant. For example, the decimal values 8.0 and 8.000 are considered equal;
- **LengthUnit**- The **lengthUnit** may be one of “inch,” “meter,” or “millimeter.” It specifies that any property that represents lengths will be expressed in **lengthUnit** units. This is defined by a DatatypeDefinition;
- **NMTOKEN**- The **NMTOKEN** is the XSDL xsd:NMTOKEN. It represents a single string token. **NMTOKEN** values may consist of letters, digits, periods (.), hyphens (-), underscores (_), and colons (:). They may start with any of these characters. **NMTOKEN** does not preserve white space, so any leading or trailing whitespace will be removed. In addition, no whitespace may appear within the value itself;
- **NonNegativeInteger-** A **nonNegativeInteger** is the XSDL xsd:nonNegativeInteger. This is defined as an arbitrarily large nonnegative integer. The digits may be optionally preceded by a plus (+) sign. Leading zeros are permitted, but decimal points are not;
- **PositiveDecimal**- A **positiveDecimal** is derived from the XSDL xsd:decimal and is restricted to be greater than zero. This is defined by a DatatypeDefinition;
- **PositiveInteger**- A **positiveInteger** is the XSDL xsd:positiveInteger. This is defined as an arbitrarily large positive integer. The digits may be optionally preceded by a plus (+) sign. Leading zeros are permitted, but decimal points are not;
- **String-** A **string** is the XSDL xsd:string. The type represents a character string that may contain any Unicode character allowed by OWL. The **string** type preserves white space, which means that all whitespace characters (spaces, tabs, carriage returns, and line feeds) are preserved;
- **WeightUnit-** The **weightUnit** may be one of “gram”, “kilogram”, “milligram” “ounce”, or “pound”. It specifies that any property that represents weights will be expressed in **weightUnit** units. This is defined by a DatatypeDefinition.

3.3.3.3. Activity Representation in the Manufacturing Kitting Ontology

In addition to representing objects, the ontology also represents activities [146]. Activities are needed show the transition between states. To represent activities in the manufacturing kitting ontology, both the actions and the pre- and post-conditions of those actions need to be represented. Preconditions and post-conditions (effects) are formed by combining state relations, as described earlier in this chapter. An example of the take-kittray (take kit tray) action is shown in Equation 18. In natural language, the take-kt action involves a robot (*robot*) equipped with an end effector (*eff*) picking up a kit tray (*kittray)* from within a large box with empty kit trays (*lbwekt*). The action is formally defined in the State Variable Representation [147] as:

$$\text{take-kittray}(\textit{robot, kittray, lbwekt, eff, worktable}) \tag{18}$$

Table 3 shows the preconditions and effects (predicates) that are associated with this action.

Table 3: Preconditions and Effects for the Action *take-kittray*

precondition	***Effects***
robot-empty *(robot)*	¬robot-empty *(robot)*
lbwekt-not-empty *(lbwekt)*	kittray-loc-robot*(kittray, robot)*
robot-with-endeffector *(robot, eeff)*	robot-holds-kittray *(robot, kittray)*
kittray-loc-lbwekt *(kittray, lbwekt)*	¬kittray-loc-lbwekt *(kittray, lbwekt)*
endeffector-loc-robot *(eeff, robot)*	
worktable-empty *(worktable)*	
endeffector-type-kittray *(eef, kittray)*	

Each of the state relations in the table above is described below:

- robot-empty(*robot*): TRUE iff robot (*robot*) is not holding anything;
- lbwekt-non-empty(*lbwekt*): TRUE iff Large Box With Empty Kit Trays (*lbwekt*) is not empty;
- robot-with-endeffector(*robot, eeff*): TRUE iff Robot (*robot*) is equipped with an EndEffector (*eeff*);
- kittray-loc-lbwekt(*kittray, lbwekt*): TRUE iff the Kit Tray (*kittray*) is in the large box with empty kit trays (*lbwekt*);
- endeffector-loc-robot(*eeff*, *robot*): TRUE iff the EndEffector (*eeff*) is being held by the robot (*robot*);
- worktable-empty(*worktable*): TRUE iff there is nothing on the Worktable (*worktable*);

- endeffector-type-kittray(*eeff, kittray*): TRUE iff the end effector (*eeff*) is designed to handle the Kit Tray (*kittray*);
- ¬rhold-empty(*robot*): TRUE iff the robot (*robot*) is holding something;
- kittray-loc-robot(*kittray, robot*): TRUE iff the Kit Tray (kittray) is being held by the robot (*robot*);
- robot-holds-kittray(*robot, kittray*): TRUE iff the Robot (*robot*) is holding the kit tray (*kittray*);
- ¬kittray-loc-lbwekt(*kittray, lbwekt*): TRUE iff the kit tray (*kittray*) is not in the large box with empty kit trays (*lbwekt*).

There are many other actions that can be performed during the kitting operation, including putting down a kit tray, picking up and putting down a part, attaching/removing an end effector, etc. Each of these actions has associated preconditions and effects as well, but is not used in this example.

3.3.3.4. Representing Manufacturing States

When modeling the state relations in the preconditions and effects shown in the previous section, the first step is to precisely define the predicates in such a way as to determine if there were similar intermediate spatial relations that could be leveraged. We can start to formalize the previous definition of the state relations as shown in the Revised Definition column of Table 4.

The predicates **robot-empty**, **¬robot-empty**, and **robot-holds-kittray** depend on the type of effector that is being used to define the predicate. We will assume there are two types of end effectors: a vacuum end effector and a parallel gripper end effector. The vacuum end effector picks objects up by positioning itself on top of the object and uses air to create a vacuum to adhere to the object. The parallel gripper end effector picks objects up by squeezing the object from both sides. Because the vacuum end effector would not reasonably be used to pick up the kit tray, the vacuum-rhold(*r, kt*) state is not included below. In the case of the vacuum end effector, the relevant predicates would be:

- Vacuum-robot-empty(*robot*)- there is no object **under and in contact with** the robot *(robot)* vacuum effector;
- ¬Vacuum-rhold-empty(*robot*)- there is an object **under and in contact with** the robot *(robot)* vacuum effector.

Table 4: Revised Definitions of Spatial Relationships

Predicate	Previous Definition	Revised Definition
lbwekt-not-empty(*lbwekt*)	TRUE iff an object is in the Large Box With Empty Kit Trays (*lbwekt*)	there is an object that is **contained in** the Large Box With Empty Kit Tray
robot-with-endeffector(*robot, eff*)	TRUE iff Robot (*r*) is equipped with an EndEffector (*eff*)	the effector is **in contact with** the robot
kittray-loc-lbwekt(*kittray, lbwekt*)	TRUE iff the Kit Tray (*kt*) is in the Large Box With Empty Kit Trays (*lbwekt*)	the Kit Tray is **contained in** the Large Box With Empty Kit Trays
endeffector-loc-robot(*eff, robot*)	TRUE iff the EndEffector (*eff*) is being held by the robot (*r*)	the effector is **in contact with** the robot (Note: Same as (3) above)
worktable-empty(*worktable*)	TRUE iff there is nothing on the Worktable (*wtable)*	there is no object that is **on top of** and **in contact with** the Worktable
endeffector-type-kittray(*eff, kittray*)	TRUE iff the EndEffector (*eff*) is designed to handle the Kit Tray (*kt*)	the Effector can handle the Kit Tray
kittray-loc-robot(*kittray, robot*)	TRUE iff the Kit Tray (kt) is being held by the Robot (*r*)	the Kit Tray is **in contact with** the Robot and there is nothing **under and in contact with** the Kit Tray
¬kittray-loc-lbwekt(*kittray, lbwekt*)	TRUE iff the Kit Tray (*kt*) is not in the Large Box With Empty Kit Trays (*lbwekt*)	the Kit Tray is **not contained in** the Large Box With Empty Kit Trays

In the case of the parallel gripper end effector, the predicates would be

- Gripper-robot-empty(*robot*)- there is no object **partially in and in contact with** the robot *(robot)* gripper;
- ¬Gripper-robot-empty(*robot*)- there is an object **partially in and in contact with** the robot *(robot)* gripper;
- Gripper-holds-kittray(*robot, kittray*)- the Kit Tray *(kittray)* is **partially in and in contact with** the robot *(robot)* gripper.

There is also one predicate that does not rely on spatial relations. The definition of endeffector-type-kittray(*eff, kittray*) states that a specific effector must be able to be used on a kit tray. This information is included in the ontology class to describe the kit tray and therefore is out of scope of this document.

Based on the manufacturing kitting ontology and the spatial relations, we can formally define the 11 manufacturing kitting predicates:

$$\text{lbwekt-not-empty}(\textit{lbwekt}) \rightarrow \text{SolidObject}(\textit{obj1}) \land \text{Contained-In}(\textit{obj1}, \textit{lbwekt}) \tag{19}$$

$$\text{robot-with-endeffector}(\textit{robot, eff}) \rightarrow \text{In-Contact-With}(\textit{robot}, \textit{endeffector}) \tag{20}$$

$$\text{kittray-loc-lbwekt}(\textit{kittray, lbwekt}) \rightarrow \text{Contained-In}(\textit{kittray, lbwekt}) \tag{21}$$

$$\text{worktable-empty}(\textit{worktable}) \rightarrow \text{SolidObject}(\textit{obj1}) \land \neg\text{On-Top-Of}(\textit{obj1}, \textit{worktable}) \land \neg\text{In-Contact-With}(\textit{obj1}, \textit{worktable}) \tag{22}$$

$$\text{gripper-holds-kittray}(\textit{robot, kittray}) \rightarrow \text{GripperEffector}(\textit{eff}) \land \text{robot-with-endeffector}(\textit{robot, eff}) \land \text{Partially-In-And-In-Contact-With}(\textit{kittray}, \textit{eff}) \tag{23}$$

$$\neg\text{kittray-loc-lbwekt}(\textit{kittray, lbwekt}) \rightarrow \neg\text{Contained-In}(\textit{kittray, lbwekt}) \tag{24}$$

$$\text{vacuum-robot-empty}(\textit{r}) \rightarrow \text{SolidObject}(\textit{obj1}) \land \text{SolidObject}(\textit{obj2}) \land \text{VacuumEffector}(\textit{eff}) \land \text{robot-with-endeffector}(\textit{robot, eff}) \land \neg(\text{Under-And-In-Contact-With}(\textit{obj1}, \textit{eff}) \land \neg\text{Under-And-In-Contact-With}(\textit{obj2}, \textit{obj1})) \tag{25}$$

$$\begin{gathered}\neg\text{vacuum-robot-empty}(r) \rightarrow \\ \text{SolidObject}(obj1) \wedge \text{SolidObject}(obj2) \wedge \text{VacuumEffector}(eff) \wedge \\ \text{robot-with-endeffector}(robot, eff) \wedge \text{ Under-And-In-Contact-With}(obj1, \\ eeff) \wedge \\ \neg\text{Under-And-In-Contact-With}(obj2, obj1)\end{gathered} \tag{26}$$

$$\begin{gathered}\text{gripper-rhold-empty}(r) \rightarrow \\ \text{SolidObject}(obj1) \wedge \text{GripperEffector}(eff) \wedge \\ \text{robot-with-endeffector}(robot, eff) \wedge \\ \neg\text{Partially-In-And-In-Contact-With}(obj1, eff)\end{gathered} \tag{27}$$

$$\begin{gathered}\neg\text{gripper-rhold-empty}(r) \rightarrow \\ \text{SolidObject}(obj1) \wedge \text{GripperEffector}(eff) \wedge \\ \text{robot-with-endeffector}(robot, eff) \wedge \\ \text{Partially-In-And-In-Contact-With}(obj1, eff)\end{gathered} \tag{28}$$

$$\text{kittray-loc-robot}(kittray, robot) \rightarrow \text{gripper-holds-kittray}(robot, kittray) \tag{29}$$

The formal definitions of these predicates will allow their existence to be logically recognized in a manufacturing environment, which in turn can be used as input into a state-based intention recognition system. The presence of predicates in certain predefined orders can help a robot recognize the intention of a human in the environment, which would allow the robot to better assist the human in performing upcoming activities. This will be discussed in the next chapter.

3.4. Conclusion

In this chapter, we described an approach to state representation and reasoning that will serve as the basis for the intention recognition approach to be described in the next chapter. We also described how these state relations are represented in the ontology and gave some background about the manufacturing scenarios to which this could be applied. We then applied the state representation and reasoning approaches to the kitting domain and described the manufacturing kitting ontology and how states could be represented in this domain. Finally, we showed some exceptions to the algorithms and how these exceptions could be identified and resolved.

In the next chapter, we will describe the intention recognition approaches. These approaches are based on the output of the state representation algorithms.

4. Intention Recognition

In this chapter, we describe the Intention Ontology, how intentions are composed of states, and how the Intention Ontology is used to determine the likelihood of intentions that are perceived in the environment. [148].

4.1. The Intention Ontology

The Intention Ontology is composed of ordering constructs and intentions. Each is described below.

4.1.1. Ordering Constructs

In this thesis, an ordering of state relationships represents an intention. As such, we need a formal ontological mechanism to allow for this ordering. To do this, we borrow some concepts that are described in OWL-S (Web Ontology Language – Services) [8]. OWL-S is described on the website (http://www.w3.org/Submission/OWL-S/) as “an ontology of services enabling a user and software agents to discover, invoke, compose, and monitor Web resources offering particular services and having particular properties.” Though intended for web-based services, many of the same ordering constructs are equally applicable to the representation of the sequencing of states. OWL-S defines eight control constructs:

- **Perform -** execution of an action;
- **OrderedList -** a list of control constructs to be done in order;
- **Split-** a “bag” of process components to be executed concurrently. The Split completes when all of its component processes have been scheduled for execution;
- **Split+Join -** concurrent execution of a group of process components with synchronization. Split+Join completes when all of its components processes have completed;
- **Any-Order -** process components (specified as a bag) to be executed in some unspecified order, but not concurrently. All components must be executed;
- **Choice -** the execution of a single control construct from a given bag of control constructs;
- **If-Then-Else** - intended as ``Test If-condition; if True do Then, if False do Else'';
- **Iterate -** makes no assumption about how many iterations are made or when to initiate, terminate, or resume. The initiation, termination, or maintenance condition could be specified with a whileCondition or an untilCondition;
- **Repeat -**While/Repeat-Until- Both of these iterate until a condition becomes false or true. Repeat-While tests for the condition, exits if it is false and does the operation if the condition is true, then loops. Repeat-Until does the operation, tests for the condition, exits if it is true, and otherwise loops.

Table 5: Initial State Representation Ordering Constructs.

OWL-S Control Construct	**Adapted State Representation Ordering Construct**	**State Representation Ordering Construct Definition**
Perform	Exists	A state relationship must exist
Sequence	OrderedList	A set of state relationships that must occur in a specific order
Any-Order	Any-Order	A set of state relationships that must all occur in any order
Iterate	Count	A state relationship that must be present exactly the number of times specified (greater than one).
Choice	Choice	A set of possible state relationships that can occur after a given state relationship
Join	Co-Exist	Two or more state relationships that must be true

We adapt some of these control constructs to represent the ordering of states by changing their name and definition as shown in Table 5. Column 2 in Table 5 shows the state ordering constructs that we defined in this work. This is not an exhaustive list, but the constructs shown are sufficient for representing the kit assembly shown later.

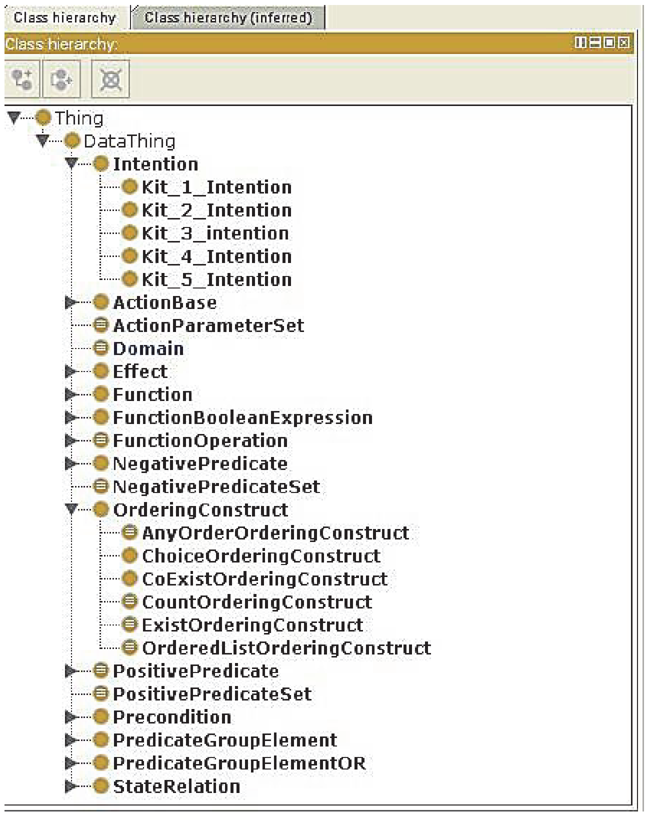
In the ontology, we represent the ordering constructs listed in Column 2 above as subclasses of the overall “OrderingConstructs” class. In addition, we also leverage the state relations that were modeled in the State Relation Ontology. In this context, a state refers to an unordered list (a bag) of state relationships which completely describe the state at a given point in time. A state contains one to many state relationships. StateRelationships are defined using the RCC8 relationships described earlier. Table 6 shows what each ontology construct points to and its cardinality restrictions.

Table 6: Construct Details

Construct	**Points to:**	**Cardinality**
State	StateRelationshipBag	1:n StateRelationships
StateRelationships	SolidObject	Exactly 2 SolidObjects
Ordering Constructs		
AnyOrder	OrderingConstructsBag	2:n OrderingConstructs
Choice	OrderingConstructsBag	2:n OrderingConstructs
CoExists	OrderingConstructsBag	2:n OrderingConstructs
Count	OrderingConstruct	Exactly 1 OrderingConstruct
Exists	StateRelationship	Exactly 1 StateRelationship
NotExists	StateRelationship	Exactly 1 StateRelationship
OrderedList	OrderingConstructsList	2:n OrderingConstructs

4.1.2. Intentions

The second part of the Intention Ontology is the representation of the intentions themselves. This part is very straightforward, since an intention is simply composed of its corresponding ordering construct. In this case, we have a ontological construct call intention, which contain subclasses representing each individual type of intention (e.g., Kit_1_Intention), and each type of intention has an attribute which points its appropriate ordering construct. Both the ordering constructs and the intentions represented in the ontology is shown in Figure 36.



```
Thing
  DataThing
    Intention
      Kit_1_Intention
      Kit_2_Intention
      Kit_3_intention
      Kit_4_Intention
      Kit_5_Intention
    ActionBase
    ActionParameterSet
    Domain
    Effect
    Function
    FunctionBooleanExpression
    FunctionOperation
    NegativePredicate
    NegativePredicateSet
    OrderingConstruct
      AnyOrderOrderingConstruct
      ChoiceOrderingConstruct
      CoExistOrderingConstruct
      CountOrderingConstruct
      ExistOrderingConstruct
      OrderedListOrderingConstruct
    PositivePredicate
    PositivePredicateSet
    Precondition
    PredicateGroupElement
    PredicateGroupElementOR
    StateRelation
```

Figure 36: The Intention Ontology

4.1.3. A Kitting Example

In this section, we will use the detailed scenario shown in Figure 37. In this scenario, a person (not shown) is constructing one of two possible kits. In this case, the type of kit being constructed constitutes the intention that is trying to be inferred. Both kits use the same kit tray (Kit Tray) and contain a series of parts placed in the kit in any order. Kit 1 contains two Part A's, two Part B's, one Part C, and one Part D. Kit 2 contains three Part A's, one Part B, and one Part C. This is shown in Figure 38. In this scenario, all parts with the same letter are identical (e.g., all A's are the same, all B's are the same). Other parts are available at the workstation and could be used for other kit assemblies that are not of interest for this example. A table is provided on which all work is performed. Parts and part trays are available from a set of boxes that can be refilled as needed. When a kit is completed, it is placed in a completed_kit box (not shown).

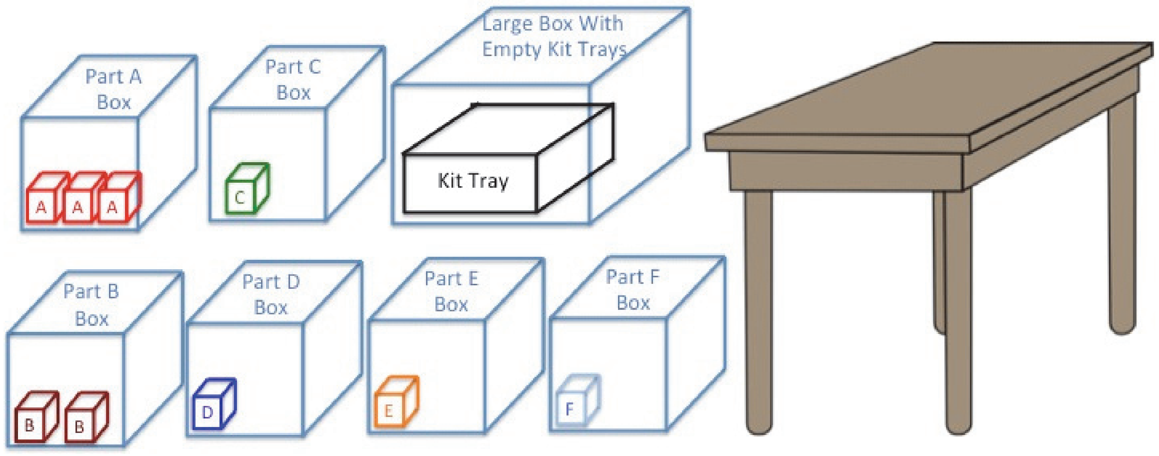


Figure 37: Sample Kitting Scenario

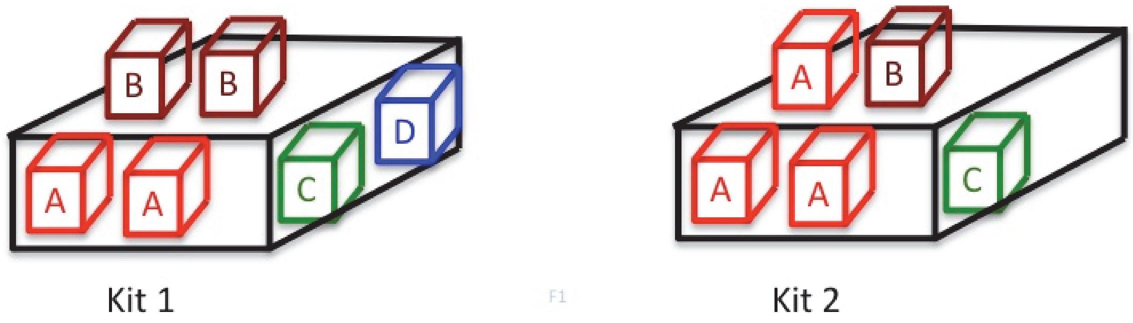


Figure 38: Completed Kits 1 and 2

A person is tasked with creating these kits. The person may choose to create either of these kits at any given time. A robot is available to assist the person by inferring which kit the person is intending to create at a given time and providing support where needed. The goal of the robot is to infer the intention of the human based on what it perceives (i.e., which kit tray the human is assembling).

4.1.4. Kitting Example Represented in the Intention Ontology

For Kit 1, we can infer from the description above that the process for assembling this kit would initially involve placing the kit tray (Kit Tray) on the table and then adding, in any order, two Part A's, two Part B's, one Part C, and one Part D. Similarly, for Kit 2, we can infer from the description above that the process for assembling this kit would initially involve placing the kit tray (Kit Tray) on the table and then adding, in any order, three Part A's, one Part B, and one Part C. We can now use the state relations and the ordering relationships to build intentions.

For Kit 1, we start by requiring that the Kit Tray be on the Table (where SR# stands for state relation and OC# stands for Ordering Construct):

$$\text{SR1 = On-Top-Of(KitTray, Table)} \tag{30}$$

$$\text{OC1 = Exists(SR1)} \tag{31}$$

Next, we represent all of the states that can happen in any order and the number of occurrences that must happen for each state relationship:

$$\text{SR2 = Contained-In(PartA, KitTray)} \tag{32}$$

$$\text{SR3 = Contained-In(PartB, KitTray)} \tag{33}$$

$$\text{SR4 = Contained-In(PartC, KitTray)} \tag{34}$$

$$\text{SR5 = Contained-In(PartD, KitTray)} \tag{35}$$

$$\text{OC2 = Count(SR2, 2)} \tag{36}$$

$$\text{OC3 = Count(SR3, 2)} \tag{37}$$

$$\text{OC4 = Count(SR4, 1)} \tag{38}$$

$$\text{OC5 = Count(SR5, 1)} \tag{39}$$

$$\text{OC6 = AnyOrder(OC2, OC3, OC4, OC5)} \tag{40}$$

$$\text{SR6 = Contained-In(KitTray, CompletedKitBox)} \tag{41}$$

$$\text{OC7} = \text{Exists(SR6)} \tag{42}$$

We also need to represent the states that cannot be true if this kit assembly is indeed the intention. In reality, there are likely an infinite number of states that could make this intention be deemed false. However, for this work, we will specifically focus on the states that are possible in other intentions, but are not possible here. One of these states may include:

$$\text{SR7} = \text{Contained-In(PartE, KitTray)} \tag{43}$$

$$\text{OC8} = \neg\text{Exists(SR7)} \tag{44}$$

Based on the above, we can represent the intention of the Kit 1 assembly as follows:

$$\text{OC9} = \text{OrderedList(OC1, OC6, OC7) \& OC8} \tag{45}$$

The ‘&’ symbol in Equation 45 represents that the ordering constraint after the symbol (OC8 in this case) must be true throughout the entire process (OC9). In other words, Part E cannot exist within the Kit Tray during any part of the process.

Using OC9 above, and applying the same approach to the Kit 2 intention (not shown), we can use a template-based approach to represent the sequence associated with the Kit 1 and Kit 2 intentions, which may look something like what is shown in Table 7 and Table 8:

Table 7: Kit 1 Intention Template

State Relation	**1**	**2**	**2**	**3**	**3**	**4**	**5**	**6**	**7**
Kit 1	On_ Top_ Of (Kit Tray, Table)	Contained -In (PartA, KitTray)	Contained -In (PartA, KitTray)	Contained -In (PartB, KitTray)	Contained -In (PartB, KitTray)	Contained -In (PartC, KitTray)	Contained -In (PartD, KitTray)	Contained-In (KitTray, Completed KitBox)	Not(Contained -In(PartE, KitTray))
Previous State Relation	n/a	1	1	1	1	1	1	2, 3, 4, 5	n/a

Table 8: Kit 2 Intention Template

State Relation	**1**	**2**	**2**	**2**	**3**	**4**	**5**	**6**	**7**
Kit 2	On_ Top_ Of (Kit Tray, Table)	Contained -In (PartA, KitTray)	Contained -In (PartA, KitTray)	Contained -In (PartA, KitTray)	Contained -In (PartB, KitTray)	Contained -In (PartC, KitTray)	Contained -In (KitTray, Complete dKitBox)	Not (Contained-In(PartD, KitTray	Not(Contained -In(PartE, KitTray))
Previous State Relation	n/a	1	1	1	1	1	2, 3, 4	n/a	n/a

All state relationships are listed, and the ones that must be true more than once (as represented by the Count construct in the ontology) are represented in separate columns. The state relationships that cannot be true for a specific intention are represented at the end of the table. Under each state relationship is a pointer to the state relationship that must occur before this one is relevant. For example, in Table 7, before the *Contained-In*(PartA, KitTray) state relationship can be evaluated as being true, the *On-Top_Of*(KitTray, Table) (as indicated by the number 1 at the bottom of the table) must have occurred. This table will be used in subsequent sections to show how intentions can be recognized based upon observations.

4.2. The Intention Recognition Algorithm

In this section, we describe the intention recognition algorithm and explain how it is used to track states and state relations as they occur, match observed state relations to intentions in the ontology, and assign likelihoods to those intentions.

4.2.1. Tracking States and State Relations as Actions Occur

Now that we have specified how the states, state relationships, and intentions are represented in the ontology, we will show how state relationships are tracked by observations and associated with the intentions. Based on the detailed kitting example described in Section 4.1.3, the first step is to determine which objects and state relations of interest are relevant to be tracked. These are referred to as objects of interest (OI) and state relations of interest (SRI). If a state relation appears in any tracked intention, that state relations becomes an SRI. If an object appears in any intention, then that object becomes an OI.

In this domain, the OIs are:

- Part A;
- Part B;
- Part C;
- Part D;
- Part E;
- Kit Tray;
- Table;
- Part A Box;
- Part B Box;
- Part C Box;
- Part D Box;
- Completed Kit Box.

Note that Part E is of interest not because it is a part if the kit assembly, but because it is explicitly prohibited from being in one of the kit assemblies. Even though Part F is available within the environment, it is not of interest to either intention and is therefore not tracked.

Using the approaches described above, the robot first extracts the state relationships that are relevant to the various intentions it is trying to perceive. In the cases of the two kit assemblies, the relevant state relationships include:

- Part A is in Part A Box;
- Part B is in Part B Box;
- Part C is in Part C Box;
- Part D is in Part D Box;
- KitTray is in Large Box With Empty Kit Trays;
- KitTray is on Table;
- Part A is in the KitTray;
- Part B is in the KitTray;
- Part C is in the KitTray;
- Part D is in the KitTray;
- Part D is not in the KitTray;
- Part E is not in the KitTray;
- Kit Tray is in Completed Kit Box.

The truth-value of these state relationships can be evaluated at a given point in time by applying the state representation approach based on RCC8 as described in the previous chapter. In tabular format, the state relationships can be represented as shown in Table 9. Columns in this table represent states. As defined earlier, a state is the union of the truth-values of the state relations on interest. Rows in this table represent individual state relations of interest. The value in the cells represents the number of instances in which the state relation is true in a given state. This will be described further in the next few paragraphs.

Actions occur between each state to cause the truth-value of the relevant state relations to change. However, for this approach, we care about the truth-value of the state relations in each state without needing to track the actions that cause them to be true or false.

New states (columns in the table) can occur at different frequencies. A new state only occurs when the truth-value of at least one state relation of interest changes. There may be multiple state relations (that are not of interest) that change their truth-value (e.g., Part F moves its location) before a new state of interest occurs. New states only come into existence when a state relation of interest changes its truth-value.

State 1 in Table 9 shows the state of the environment as described in Figure 37. Numbers in the cells represent how many instances of the state relationship are true. In this example, there are three instances of Part A in the Part A Box, as shown in Figure 37. If, at the next state, the Kit Tray was removed from the Kit Tray Holder and placed on the Table, the next state would be represented as in State 2. If one of the Part A's is then removed from the Part A Box and placed into the Kit Tray, the state representation would look like State 3.

Table 9: Real-Time State Relation Tracking Table

State Relation	State 1 (Initial State)	State 2 (Kit Tray on Table)	State 3 (Part A in Kit Tray)	…
Part A is in Part A Box	3	3	2	
Part B is in Part B Box	2	2	2	
Part C is in Part C Box	1	1	1	
Part D is in Part D Box	1	1	1	
Kit Tray is in Large Box With Empty Kit Trays	1	0	0	
Kit Tray is on Table	0	1	1	
Part A is in Kit Tray	0	0	1	
Part B is in Kit Tray	0	0	0	
Part C is in Kit Tray	0	0	0	
Part D is in Kit Tray	0	0	0	
Part E is in Kit Tray	0	0	0	
Kit Tray in Completed Kit Box	0	0	0	

Understanding which states are relevant also plays a significant role in determining which sensors to use and where to place the sensors that are meant to track objects in the environment [149]. For example, if we assume that the Part A Box is not a clear box and there is an opening at the top, a sensor would need to be placed above the box so that it is evident if (and how many) objects are present within the box (addressing the state relationship “Part A is in the Part A Box”).

4.2.2. Matching Perceived Real-Time State Relations to Intention Templates

As mentioned above, a new state in the State Relation Tracking Table comes into existence when a state relation of interest that is being tracked changes its truth-value. As an example, if the Kit Tray goes from *not being on the table* to being *on the table*, a new state is formed. At each state, the state relationships that have changed value are compared to the intention templates shown in Table 7 and Table 8.

Only the *eligible* state relationships in Table 7 are available for matching. A state relationship becomes eligible if all required previous state relationships have occurred. For example, in The Kit 1 Intention Template in Table 7, the state relationship *Contained-In*(PartA, KitTray) can not occur until *On_Top_Of* (Kit Tray, Table) has occurred. Therefore, even if *Contained-In*(PartA, KitTray) is true in the observed scene, it is not eligible for matching unless *On_Top_Of* (Kit Tray, Table) has previously been matched.

As each observed state occurs and is entered in the State Relation Tracking Table, the corresponding state relationships are compared with those associated with the possible intentions. When the observed state relationship(s) match an eligible state relationship in an Intention Template, it is “checked off” indicating that the state relationship has occurred. This same process occurs with every new observed state that occurs. Continuing with our example, if the State Relation Tracking Table was updated to look like Table 10, the corresponding Kit 1 and Kit 2 Intention Templates would look like Table 11 and Table 12, respectively, with green cells representing those state relations which have been observed and whose precondition state relations have been met.

One benefit of this approach is the lack of the underlying assumption that the observer started watching the intention from the beginning. In the example of the kit tray above, once the kit tray is on the table and once the parts are in the kit tray, they will continue to be in that location and the corresponding state relations will be true independent of when the observer started to observe. When the intention recognition system starts up, and once the initial state of the environment is assessed, the relevant Intention Templates will be matched with the state relations that are true at that time. The subsequent “filled in” tables should be identical to what is shown in Table 11 and Table 12, even if the intentions were not observed from the beginning. This is a clear advantage over intention recognition systems that rely on activity recognition, because if the activity occurs before the intention is being observed, there is no way for the system to be able to account for the activities that were not perceived.

Table 10: Real-Time State Relation Tracking Table (Version 2)

State Relation	State 1 (Initial State)	State 2 (Kit Tray On Table)	State 3 (Part A In Kit Tray)	State 3 (2nd Part A In Kit Tray)	State 4 (Part B in Kit Tray)	State 5 (Part C In Kit Tray)
Part A is in Part A Box	3	3	2	1	1	1
Part B is in Part B Box	2	2	2	2	1	1
Part C is in Part C Box	1	1	1	1	1	0
Part D is in Part D Box	1	1	1	1	1	1
Kit Tray is in Large Box With Empty Kit Trays	1	0	0	0	0	0
Kit Tray is on Table	0	1	1	1	1	1
Part A is in Kit Tray	0	0	1	2	2	2
Part B is in Kit Tray	0	0	0	0	1	1
Part C is in Kit Tray	0	0	0	0	0	1
Part D is in Kit Tray	0	0	0	0	0	0
Part E is in Kit Tray	0	0	0	0	0	0
Kit Tray in Completed Kit Box	0	0	0	0	0	0

Table 11: Kit 1 Intention Template (Version 2)

State Relation	**1**	**2**	**2**	**3**	**3**	**4**	**5**	**6**	**7**	**Sum**
Kit 1	On_ Top_ Of (Kit Tray, Table)	Contained -In (PartA, KitTray)	Contained -In (PartA, KitTray)	Contained -In (PartB, KitTray)	Contained -In (PartB, KitTray)	Contained -In (PartC, KitTray)	Contained -In (PartD, KitTray)	Contained -In (KitTray, Complete dKitBox)	Not(Contained -In(PartE, KitTray	5
Previous State Relations	n/a	1	1	1	1	1	1	2, 3, 4, 5	n/a	

Table 12: Kit 2 Intention Template (Version 2)

State Relation	**1**	**2**	**2**	**2**	**3**	**4**	**5**	**6**	**7**	**Sum**
Kit 2	On_ Top_ Of (Kit Tray, Table)	Contained -In (PartA, KitTray)	Contained -In (PartA, KitTray)	Contained -In (PartA, KitTray)	Contained -In (PartB, KitTray)	Contained -In (PartC, KitTray)	Contained -In (KitTray, CompletedKitBox)	Not(Contained -In(PartD, KitTray	Not(Contained -In(PartE, KitTray	5
Previous State Relations	n/a	1	1	1	1	1	2, 3, 4	n/a	n/a	

The exception to this is intentions that require an object to be moved multiple times. For example, if the steps to creating a kit required a part to be first moved to the table and then to be moved to the kit tray, and the observer started observing only after the part was in the kit tray, there would be no way for the system to know that the part was previously on the table. The observer would therefore not have the knowledge to know that the state relation of the part of the table was ever true. However, the focus of this thesis does not include intentions that required multiple moves of a single object.

4.2.3. Associating Likelihoods to Intentions

Based on the content of Table 11 and Table 12 above, along with other supporting information, we can start to associate likelihoods to the intentions being considered [150]. Section 4.2.3.1. describes the overall approach to do so.

4.2.3.1. Overall Likelihood Equation

The overall equation that is used to determine the likelihood of intentions is as follows:

$$L_{i,s} = \left(\frac{\sum_{k=1}^{n}(A_{k,i,s} * W_{A_k})}{\sum_{k=1}^{n} W_{A_k}}\right) * 100 \qquad (46)$$

where:

- $\boldsymbol{L_{i,s}}$ is the numeric likelihood of an intention *i* in state *s;*
- $\boldsymbol{A_{k,i,s}}$ is the numeric result of applying intention recognition approach *k* for intention *i* in state *s;*

- $\boldsymbol{W_{A_k}}$ is the weight of intention recognition approach *k;*
- $\boldsymbol{n}$ is the total number of intention recognition approaches

The output of all individual intention recognition approaches must contain a value between 0 and 1, where 0 is the lowest value and 1 is the highest value. Intention recognition approaches (multiplied by their associated weights) are added together and then divided by the sum of all of their weights. Weights are associated with the intention recognition approaches to show the relative importance of one approach over another. The larger the weight, the greater the impact the approach will have on the overall likelihood. These weights can contain any value greater than or equal to zero (no upper bound).

The concept behind this equation is shown in Figure 39, where the individual intention recognition approaches, multiplied by their respective weights, are fed into the overall likelihood equation presented in Equation 46.

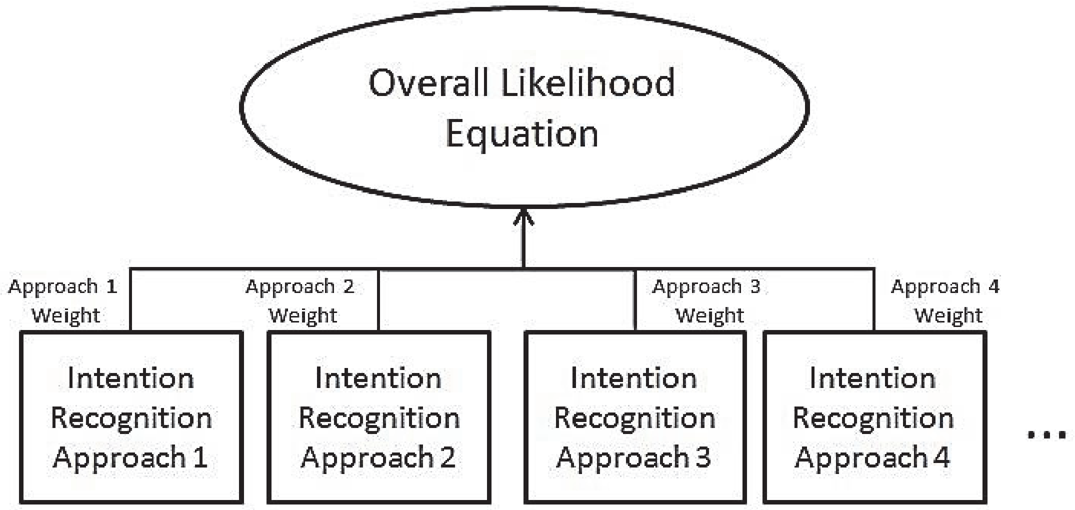

Overall Likelihood Equation
Approach 1 Weight
Approach 2 Weight
Approach 3 Weight
Approach 4 Weight
Intention Recognition Approach 1
Intention Recognition Approach 2
Intention Recognition Approach 3
Intention Recognition Approach 4
...

Figure 39: Diagram Explaining Overall Likelihood Equation

4.2.4. Individual Intention Recognition Approaches Explored

There were four intention recognition approaches explored as part of this thesis. The first, a Bayesian approach, is a fairly well documented approach in the literature. The next three approaches are unique to this thesis. These additional three approaches were determined by performing a simple non-scientific experiment. I asked approximately twenty people to look at data in Table 13. As described below, the table represents true state relations in five different possible intentions. I also explained that intentions could start in the middle of the table; they did not need to start at the beginning. Based on this table, I asked them to guess which intention was being performed and why they though that it was. All twenty people different answers, but what was common was the rationale behind why they chose the intention that they did. Three common rationales emerged:

- "I chose Intention X because it had the more true state relations."
- "I chose Intention X because it had the highest percentage of state relations complete."
- "I chose Intention X because it seems that most of the true state relations occurred most recently."

These three responses served as the basis for three of the intention recognition approaches described in this section.

All approaches are explained through the use of a simple example. For the purpose of the experiment described in the next chapter, the Bayesian approach was initially used, with the additional three intention recognition approaches added afterwards to show their effect on the overall performance.

4.2.4.1. Intention Recognition Example

Table 13 show five sample intentions listed as one through five in the left-most column. Each intention has a total number of state relations that must be true for the intention to be considered complete. This is shown in column two. The next eight columns show the progression of eight states (time periods) in which the environment was observed. Green boxes show that a single state relation because true in an intention at that state. In State 1, some set of state relations occurred that caused a state relation to become true in intentions 2, 3, and 5. In state 2, an addition set of state relations occurred that also caused a state relation to become true in intentions 2, 3, and 5. In state 3, some set of state relations occurred that caused a state relation to become true in intentions 1, 2, and 6, and so on. This chart does not show what state relation became true in each intention; it only shows that some state relation became true.

Table 13: Intentions vs. Matching States Relations

		States							
Intention	Total State Relations	1	2	3	4	5	6	7	8
1	10					🟩	🟩	🟩	🟩
2	16	🟩		🟩		🟩		🟩	
3	6	🟩	🟩						
4	20							🟩	🟩
5	6	🟩	🟩	🟩	🟩				

Based on the data in Table 13, we can determine the value of the following variables that will be used as input into some of the subsequent intention recognition approaches.

- **matchedSR$_{i,s}$** - Number of matched state relations (*SR*) in an intention *i* as of the current state *s;*
- **totalSR$_{I}$** – Total number of state relations (*SR*) (whether matched or not) in an intention *I;*
- **allSR$_{s}$** – Number of matched state relations (*SR*) in all possible intentions as of the current state *s;*
- **S$_{total}$** – Number of states (*S*) that have occurred since observation began;
- **rmatchedSR$_{r,i,s}$** - Number of matched state relations (*SR*) in the past *r* states in intention *i* as of the current state *s*. In other words, in the most recent *r* states, how many true state relations for an intention exist?

We can summarize the variables for each intention as of State 8 in Table 14 (the whole table) as follows:

Table 14: Summary of Intention Variables at State 8

Intention (*i*)	**Total True State Relations (matchedSR$_{i,8}$)**	**Total State Relations (totalSR$_{i}$)**	**True State Relations that Have Occurred in the Past r=4 States (rmatchedSR$_{4,i,8}$)**
1	4	10	4
2	4	16	2
3	2	6	0
4	2	20	2
5	4	6	0

In addition for State 8:

- Number of states that have occurred since observation began (S_{total}) = 8;
- Number of matched state relations in all possible intentions at the current state (allSR_s) = 16 (sum of 4, 4, 2, 2, 4).

These variables will now be used as input into the approaches described in the next sections.

The subsequent intention recognition algorithms, apart from the Bayesian approach, were chosen more empirically than theoretically. Based on an informal experiment, I showed a set of students Table 13 and asked them which intention they thought was being developed and why. I got answers such as:

- The intention that had the most state relations that were true (independent of when they happened and independent of how many total state relations possibilities there were in the intention).
- The intention that had the highest percentage of state relations that were true
- The intention that had the most true state relations recently

These formed the basis for the additional intention algorithms described below.

4.2.4.2. Bayesian Approach

Bayesian probability theory provides a mathematical framework for performing inference, or reasoning, using probability. It is based on Bayes rule, which involves the manipulation of conditional probabilities. In Bayesian probability theory, one of these "events" is the hypothesis and the other is data, and we wish to judge the relative truth of the hypothesis given the data. It also used a likelihood function to assess the probability of the observed data arising from the hypothesis. Usually this is known by the experimenter, as it expresses one's knowledge of how one expects the data to look given that the hypothesis is true. One of the advantages of Bayesian probability theory is that one's assumptions are made up front, and any element of subjectivity in the reasoning process is directly exposed. [25, 28, 51]

The formula describing the chance that kit *j* is being made after a given set of observed parts is:

$$\Pr\left(kit_j \middle| \, observed\ parts\right) = \frac{\Pr\left(observed\ parts \middle| kit_j\right)\Pr(kit_j)}{\sum_{j=1}^{m}\Pr\left(observed\ parts \middle| kit_j\right)\Pr(kit_j)} \qquad (47)$$

Here, *m* is the total number of intentions and $\mathbf{Pr}\left(\boldsymbol{observed\ parts}\middle|\boldsymbol{kit_j}\right)\mathbf{Pr}(\boldsymbol{kit_j})$ describes the chances of seeing a particular combination of state relations composed of parts ("observed parts") given that the parts are being taken for kit *j*. If each time a new state relation occurs, it is created randomly from the remaining parts, then this is described by a multivariate hypergeometric distribution. For this work, there is the assumption that each part is randomly selected from the set of part remaining for the kit that is being built.

$\mathbf{Pr}(\boldsymbol{kit_j})$ denotes the proportion of all kits made that are kit *j*. The assumption is that each kit is equally likely to be built, but this can be easily changed.

A description of the flow of the Bayesian algorithm is as follows:

Suppose a kit is described by the number of pieces it contains for each type. That is, kit *i* = (ni1,ni2, …, niq) has ni1 pieces of type "1", ni2 pieces of type "2" and so on. Suppose an observation is described by the number of pieces seen for each type. Observation *j* = (xj1, xj2, …, xjq) has seen xj1 pieces of type “1”, xj2 pieces of type “2” and so on. The likelihood, **L()**, of observation j under kit i is given by the multivariate hypergeometric distribution:

$$A(observation\ j\ |kit\ i) = \frac{product_p = 1, \dots ,\ \ q\ (nip\ choose\ xjp)}{\left(sum\ p = 1, \dots , q(nip)\right) choose\ (sum\_p = 1, \dots , q\ (xip))} \qquad (48)$$

Below is a solution written in the form of commented R code. (R is a free, open-source statistical programming language available for download at www.r-project.org). The areas highlighted in yellow represent output from the code. The code captures part of the experiment that is presented in the next chapter, not the simple example that was provided in the previous section. The previous simple example will be used for subsequent intention recognition approaches.

```
#### R code ###

### Computer output is highlighted in yellow

install.packages("BiasedUrn")
library(BiasedUrn)

### Create part.list to match the following description
##- Kit one has four red parts, three green parts, three blue parts
##- Kit two has four red parts, four green parts, two blue parts
##- Kit three has five blue parts, three green parts, two red parts
##- Kit four has four red parts, three green parts, two blue parts, and one orange part
##- Kit five has thee blue parts, three green parts, two red parts, one orange part, and one

yellow part
part.list<-list(c(4,3,3,0,0),
c(4,4,2,0,0),
c(2,3,5,0,0),
c(4,3,2,1,0),
c(2,3,3,1,1))
names(part.list)<-paste("kit",1:5)
for(i in 1:length(part.list)) names(part.list[[i]])<-c("red","green","blue","orange","yellow")
part.list

$`kit 1`
  red  green   blue orange yellow
```

```
   4     3     3     0     0

$`kit 2`
   red  green   blue orange yellow
     4     4     2     0     0

$`kit 3`
   red  green   blue orange yellow
     2     3     5     0     0

$`kit 4`
   red  green   blue orange yellow
     4     3     2     1     0

$`kit 5`
   red  green   blue orange yellow
     2     3     3     1     1
```

```
kit.probs <- function(obs, part.list, prior = NULL) {

    ## If unspecified, assume all kits are equally likely to be made
    if (is.null(prior))
        prior <- rep(1, length(part.list))

    ## Normalize prior so it sums to 1
    prior <- prior/sum(prior)

    ## Compute Pr(obs|kit) for each kit assuming multivariate hypergeometric distribution
    like <- NULL

   for (i in 1:length(part.list)) like <- c(like, dMWNCHypergeo(x = obs, m = part.list[[i]],
   n =  sum(part.list[[i]]), odds = rep(1, length(part.list))))

   if (!any(like != 0)) {
        print("Observed parts do not match part list for any kit")
        prob <- like
    }

   if (any(like != 0)) {
        prob <- like * prior
        prob <- prob/sum(prob)
    }

   names(prob) <- names(part.list)
    ## Return Pr(kit|obs) for each kit
    prob
}
```

```
### Example progression when observed parts order is:  1. Red, 2. Green, 3. Green, 4.
Orange

kit.probs(c(red = 1, green = 0, blue = 0, orange = 0, yellow = 0), part.list = part.list)

kit 1 kit 2 kit 3 kit 4 kit 5
0.250 0.250 0.125 0.250 0.125

kit.probs(c(red = 1, green = 1, blue = 0, orange = 0, yellow = 0), part.list = part.list)

kit 1     kit 2     kit 3     kit 4     kit 5
0.2307692 0.3076923 0.1153846 0.2307692 0.1153846

kit.probs(c(red = 1, green = 2, blue = 0, orange = 0, yellow = 0), part.list = part.list)

kit 1 kit 2 kit 3 kit 4 kit 5
  0.2   0.4   0.1   0.2   0.1

kit.probs(c(red = 1, green = 2, blue = 0, orange = 1, yellow = 0), part.list = part.list)

kit 1     kit 2     kit 3     kit 4     kit 5
0.0         000 0.0000000 0.6666667 0.3333333
```

For this simple example, a Bayesian approach does not lend well because we are only focusing on cases where a state relation is true or not true, not what state relation is true (e.g., the red block was put into the kit tray). In the experiment describes later in this thesis, we will show how the Bayesian approach provides significant value in determining the intention. For the purpose of this simple example, we will assume that the Bayesian approach indicated that all intentions are equally possible (each have a value of 0.20).

$$AM_{1,1,8} = 0.20 \tag{49}$$

$$AM_{1,2,8} = 020 \tag{50}$$

$$AM_{1,3,8} = 0.20 \tag{51}$$

$$AM_{1,4,8} = 0.20 \tag{52}$$

$$AM_{1,5,8} = 0.20 \tag{53}$$

We need to assign a weight capturing the importance of this intention recognition approach with respect to the others. This could be any value greater than zero. In this example, we will use a value of 10 (which is the same value we will use for all of the approaches). Later in the thesis, we will explore the optimal set of weights for the individual intention recognition approaches.

At this stage, we only have one intention recognition approach (*n*=1), and therefore Equation 46 for each intention would be represented as follows:

$$L_{1,8} = \left(\frac{\sum_{k=1}^{1}(A_{k,i,s} * W_{A_k})}{\sum_{k=1}^{1} W_{A_k}}\right) * 100 = \frac{0.20 * 10}{10} * 100 = 20 \qquad (54)$$

$$L_{2,8} = \left(\frac{\sum_{k=1}^{1}(A_{k,i,s} * W_{A_k})}{\sum_{k=1}^{1} W_{A_k}}\right) * 100 = \frac{0.20 * 10}{10} * 100 = 20 \qquad (55)$$

$$L_{3,8} = \left(\frac{\sum_{k=1}^{1}(A_{k,i,s} * W_{A_k})}{\sum_{k=1}^{1} W_{A_k}}\right) * 100 = \frac{0.20 * 10}{10} * 100 = 20 \qquad (56)$$

$$L_{4,8} = \left(\frac{\sum_{k=1}^{1}(A_{k,i,s} * W_{A_k})}{\sum_{k=1}^{1} W_{A_k}}\right) * 100 = \frac{0.20 * 10}{10} * 100 = 20 \qquad (57)$$

$$L_{5,8} = \left(\frac{\sum_{k=1}^{1}(A_{k,i,s} * W_{A_k})}{\sum_{k=1}^{1} W_{A_k}}\right) * 100 = \frac{0.20 * 10}{10} * 100 = 20 \qquad (58)$$

4.2.4.3. Number of Observed State Relations That Are True in an Intention (Compared to Other Intentions)

The second intention recognition approach is the number of state relations that are true in an intention. Recall above the variable *matchedSR*$_{i,s}$ represents the number of true state relations in Intention *i* at State *s* and the variable *allSR*$_s$ represents all matched state relations in all possible intentions as of the State *s*. The formula for this intention recognition approach for Intention *i* in State *s* ($A_{2,i,s}$) is:

$$A_{2,i,s} = \frac{matchedSR_{i,s}}{allSR_{,s}} \tag{59}$$

This formula represents the percentage of true state relations that are in Intention *i* as compared to the sum of all of the true state relations in all of the intentions of interest. It is evaluated for every intention of interest at every state. When there are no true state relations in any intention of interest, the denominator will equal zero and the equation will be undefined. To address this, a rule is included which states that when $allSR_{s}$ = 0, then the overall equation should be set to zero.

As with all of the intention recognition approaches, they must contain a value between 0 and 1. Based on this formula, at each state, the sum of all of the $A_{2,i,s}$values will equal one (since each one is a percentage of the whole), thus the value of each $A_{2,i,s}$ will be between 0 and 1.

Using our example above and the data in Table 14, we can evaluate the value for $A_{2,i,s}$ for State 8 as follows. Note that there are five intentions of interest, so *m* = 5 in this case. In addition, because there are two intention recognition approaches, *n* = 2. The same procedure could be followed for each previous and subsequent state.

$$A_{2,1,8} = \frac{matchedSR_{1,8}}{allSR_{8}} = \frac{4}{16} = 0.25 \tag{60}$$

$$A_{2,2,8} = \frac{matchedSR_{2,8}}{allSR_{8}} = \frac{4}{16} = 0.25 \tag{61}$$

$$A_{2,3,8} = \frac{matchedSR_{3,8}}{allSR_{8}} = \frac{2}{16} = 0.13 \tag{62}$$

$$A_{2,4,8} = \frac{matchedSR_{4,8}}{allSR_{8}} = \frac{2}{16} = 0.13 \tag{63}$$

$$A_{2,5,8} = \frac{matchedSR_{5,8}}{allSR_{8}} = \frac{4}{16} = 0.25 \tag{64}$$

In addition, we need to assign a weight capturing the importance of this intention recognition approach with respect to the others. This could be any value greater than zero. In this example, we will again use a value of 10.

At this stage, we have two intention recognition approaches (*n*=2), and therefore Equation 46 for each intention would be represented as follows:

$$L_{1,8} = \left(\frac{\sum_{k=1}^{2}(A_k * W_{A_k})}{\sum_{k=1}^{2} W_{A_k}}\right) * 100 = \frac{(0.20 * 10) + (0.25 * 10)}{10 + 10} * 100 = 22.5 \tag{65}$$

$$L_{2,8} = \left(\frac{\sum_{k=1}^{2}(A_k * W_{A_k})}{\sum_{k=1}^{2} W_{A_k}}\right) * 100 = \frac{(0.20 * 10) + (0.25 * 10)}{10 + 10} * 100 = 22.5 \tag{66}$$

$$L_{3,8} = \left(\frac{\sum_{k=1}^{2}(A_k * W_{A_k})}{\sum_{k=1}^{2} W_{A_k}}\right) * 100 = \frac{(0.20 * 10) + (0.13 * 10)}{10 + 10} * 100 = 16.3 \tag{67}$$

$$L_{4,8} = \left(\frac{\sum_{k=1}^{2}(A_k * W_{A_k})}{\sum_{k=1}^{2} W_{A_k}}\right) * 100 = \frac{(0.20 * 10) + (0.13 * 10)}{10 + 10} * 100 = 16.3 \tag{68}$$

$$L_{5,8} = \left(\frac{\sum_{k=1}^{2}(A_k * W_{A_k})}{\sum_{k=1}^{2} W_{A_k}}\right) * 100 = \frac{(0.20 * 10) + (0.25 * 10)}{10 + 10} * 100 = 22.5 \tag{69}$$

Adding this new approach, Intentions 1, 2, and 5 become the most probable, which Intentions 3 and 4 have a lower likelihood. This is not surprising since Intentions 1, 2, and 5 have the highest number of state relations that are true as of State 8.

4.2.4.4. Percentage of an Intention That Is Complete

The third intention recognition approach is the percentage of state relations that are true in an intention. Recall above the variable *matchedSR*$_{i,s}$ represents the number of true state relations in Intention *i* at State *s* and *totalSR*$_i$ represents the total number of state relations (whether matched or not) in Intention *i*. The formula for the percent complete for Intention *i* in State *s* (*PercentComplete*$_{i,s}$) is:

$$PercentComplete_{i,s} = \frac{matchedSR_{i,s}}{totalSR_i} \tag{70}$$

We then normalize this for all intentions of interest to find the value of this intention recognition approach for Intention *i* in State *s* ($A_{3,i,s}$):

$$A_{3,i,s} = \frac{PercentComplete_{i,s}}{\sum_{i=1}^{p} PercentComplete_{i,s}} \tag{71}$$

where *p* represents the total number of intentions.

This formula represents the percentage of state relations that have been evaluated as true in an intention compared to the total number of state relations in all intentions. It is evaluated for every intention of interest at every state. As with all of the approaches, it must contain a value between 0 and 1.

Using our example above and the data in Table 14, we can evaluate the value for $AM_{2,i,s}$ for State 8 as follows.

$$PercentComplete_{1,8} = \frac{matchedSR_{1,8}}{totalSR_1} = \frac{4}{10} = 0.40 \tag{72}$$

$$PercentComplete_{2,8} = \frac{matchedSR_{2,8}}{totalSR_2} = \frac{4}{16} = 0.25 \tag{73}$$

$$PercentComplete_{3,8} = \frac{matchedSR_{3,8}}{totalSR_3} = \frac{2}{6} = 0.33 \tag{74}$$

$$PercentComplete_{4,8} = \frac{matchedSR_{4,8}}{totalSR_4} = \frac{2}{20} = 0.10 \tag{75}$$

$$PercentComplete_{5,8} = \frac{matchedSR_{5,8}}{totalSR_5} = \frac{4}{6} = 0.66 \tag{76}$$

The sum of all of these percentages is 0.40 + 0.25 + 0.33 + 0.10 + 0.66 = 1.75.

As before, because there are five intentions of interest, *p* = 5. We can normalize these as follows:

$$A_{3,1,8} = \frac{PercentComplete_{1,8}}{\sum_{i=1}^{5} PercentComplete_i} = \frac{0.40}{1.75} = 0.23 \tag{77}$$

$$A_{3,2,8} = \frac{PercentComplete_{2,8}}{\sum_{i=1}^{5} PercentComplete_i} = \frac{0.25}{1.75} = 0.14 \tag{78}$$

$$A_{3,3,8} = \frac{PercentComplete_{3,8}}{\sum_{i=1}^{5} PercentComplete_i} = \frac{0.33}{1.75} = 0.19 \tag{79}$$

$$A_{3,4,8} = \frac{PercentComplete_{4,8}}{\sum_{i=1}^{5} PercentComplete_i} = \frac{0.10}{1.75} = 0.06 \tag{80}$$

$$A_{3,5,8} = \frac{PercentComplete_{5,8}}{\sum_{i=1}^{5} PercentComplete_i} = \frac{0.66}{1.75} = 0.38 \tag{81}$$

As with the other intention recognition approaches, we need to assign a weight capturing the importance of this approach with respect to the others. In this example, we will again use a value of 10. In addition, because there are three intention recognition approaches, *n* = 3.

Equation 46 for each intention would be represented as follows:

$$L_{1,8} = \left(\frac{\sum_{k=1}^{3}(A_k * W_{A_k})}{\sum_{k=1}^{3} W_{A_k}}\right) * 100 = \frac{(0.20 * 10) + (0.25 * 10) + (0.23 * 10)}{10 + 10 + 10} * 100 = 22.6 \tag{82}$$

$$L_{2,8} = \left(\frac{\sum_{k=1}^{3}(A_k * W_{A_k})}{\sum_{k=1}^{3} W_{A_k}}\right) * 100 = \frac{(0.20 * 10) + (0.25 * 10) + (0.14 * 10)}{10 + 10 + 10} * 100 = 19.8 \tag{83}$$

$$L_{3,8} = \left(\frac{\sum_{k=1}^{3}(A_k * W_{A_k})}{\sum_{k=1}^{3} W_{A_k}}\right) * 100 = \frac{(0.20 * 10) + (0.13 * 10) + (0.19 * 10)}{10 + 10 + 10} * 100 = 17.2 \tag{84}$$

$$L_{4,8} = \left(\frac{\sum_{k=1}^{3}(A_k * W_{A_k})}{\sum_{k=1}^{3} W_{A_k}}\right) * 100 = \frac{(0.20 * 10) + (0.13 * 10) + (0.06 * 10)}{10 + 10 + 10} * 100 = 12.7 \tag{85}$$

$$L_{5,8} = \left(\frac{\sum_{k=1}^{3}(A_k * W_{A_k})}{\sum_{k=1}^{3} W_{A_k}}\right) * 100 = \frac{(0.20 * 10) + (0.25 * 10) + (0.38 * 10)}{10 + 10 + 10} * 100 = 27.7 \tag{86}$$

Adding this new approach, Intention 5 rises to the top by a considerable amount, followed by Intention 1, then Intention 2, 3, and 4 in that order. Intention 5 has the greatest percentage complete (67% of all of its state relations), which is why this approach had such a strong effect on it. Intention 4 has the lowest percentage complete, which is why this approach caused it to drop to the bottom.

4.2.4.5. Number of Productive States That Have Occurred Recently

The fourth intention recognition approach is the number of productive states that have occurred recently. This approach attempts to focus on the intentions that are likely occurring near the current time as opposed to those which have started many states in the past.

Recall from above that the variable $rmatchedSR_{r,i,s}$ represents the number of true state relations in the past *r* states in an Intention *i* as of the current State *s*. In this case, we will set *r* equal to four (the past four states), though this number can be varied by the user. The formula for this intention recognition approach for Intention i in State *s* ($A_{4,i,s}$) is:

$$A_{4,i,s} = \frac{rmatchedSR_{r,i,s}}{\sum_{i=1}^{p} rmatchedSR_{r,i,s}} \tag{87}$$

As before, *p* is the total number of intentions of interest.

As with the previous intention recognition approach, it is evaluated for every intention of interest at every state and must contain a value between 0 and 1.

Using our example above and the data in Table 14, we can evaluate the value for $A_{4,i,s}$ for State 8 as follows.

The sum of the productive state relations in the most recent four states to State 8 were: 4+2+0+2+0 = 8. Therefore, the equation becomes:

$$A_{4,1,8} = \frac{rmatchedSR_{4,1,8}}{\sum_{i=1}^{p} rmatchedSR_{4,i,8}} = \frac{4}{8} = 0.50 \tag{88}$$

$$A_{4,2,8} = \frac{rmatchedSR_{4,2,8}}{\sum_{i=1}^{p} rmatchedSR_{4,i,8}} = \frac{2}{8} = 0.25 \tag{89}$$

$$A_{4,3,8} = \frac{rmatchedSR_{4,3,8}}{\sum_{i=1}^{p} rmatchedSR_{4,i,8}} = \frac{0}{8} = 0.00 \tag{90}$$

$$A_{4,4,8} = \frac{rmatchedSR_{4,4,8}}{\sum_{i=1}^{p} rmatchedSR_{4,i,8}} = \frac{2}{8} = 0.25 \tag{91}$$

$$A_{4,5,8} = \frac{rmatchedSR_{4,5,8}}{\sum_{i=1}^{p} rmatchedSR_{4,i,8}} = \frac{0}{8} = 0.00 \tag{92}$$

We again need to assign a weight capturing the importance of this approach with respect to the others. In this example, we will again use a weight of 10 for this approach.

With these four intention recognition approaches (*n*=4), Equation 46 for each intention would be represented as follows:

$$L_{1,8} = \left(\frac{\sum_{k=1}^{4}(A_k * W_{A_k})}{\sum_{k=1}^{4} W_{A_k}}\right) * 100 = \frac{(0.20 * 10) + (0.25 * 10) + (0.23 * 10) + (0.50 * 10)}{10 + 10 + 10 + 10} * 100 = 29.5 \tag{93}$$

$$L_{2,8} = \left(\frac{\sum_{k=1}^{4}(A_k * W_{A_k})}{\sum_{k=1}^{4} W_{A_k}}\right) * 100 = \frac{(0.20 * 10) + (0.25 * 10) + (0.14 * 10) + (0.25 * 10)}{10 + 10 + 10 + 10} * 100 = 21.1 \tag{94}$$

$$L_{3,8} = \left(\frac{\sum_{k=1}^{4}(A_k * W_{A_k})}{\sum_{k=1}^{4} W_{A_k}}\right) * 100 = \frac{(0.20 * 10) + (0.13 * 10) + (0.19 * 10) + (0 * 10)}{10 + 10 + 10 + 10} * 100 = 12.9$$

$$L_{4,8} = \left(\frac{\sum_{k=1}^{4}(A_k * W_{A_k})}{\sum_{k=1}^{4} W_{A_k}}\right) * 100$$
$$= \frac{(0.20 * 10) + (0.13 * 10) + (0.05 * 10) + (0.25 * 10)}{10 + 10 + 10 + 10} * 100 = 15.8 \tag{95}$$

$$L_{5,8} = \left(\frac{\sum_{k=1}^{4}(A_k * W_{A_k})}{\sum_{k=1}^{4} W_{A_k}}\right) * 100$$
$$= \frac{(0.20 * 10) + (0.25 * 10) + (0.38 * 10) + (.00 * 10)}{10 + 10 + 10 + 10} * 100 = 20.8 \tag{96}$$

Adding this new intention recognition approach causes a significant change in ordering in the likelihoods of the intentions. Intention 1 jumps to the top by a significant amount, followed by Intention 5, the Intentions 2, 3, and 4 (in that order). Intention 1 had productive state relations in all four of the most recent states, while the next highest intention had no more than two. Two intentions had zero true state relations in the last four states. These intentions (three and five) had their likelihood drop significantly when applying this approach.

4.2.4.6. Exploring the Impact of the Intention Recognition Approaches

Table 15 shows how the likelihood of the intention changes once additional approaches are added into the equation.

Table 15: The Effect of Metrics on Likelihood

Intention	Approach 1 – Bayesian Approach (Weight 10)	Approach 2 – Quantity of True State Relations (Weight 10)	Approach 3 – Percentage Intention Complete (Weight 10)	Approach 4 – Recent True State Relations (Weight 10)
1	20	22.5	22.6	29.5
2	20	22.5	19.8	21.1
3	20	16.25	17.2	12.9
4	20	16.25	12.7	15.8
5	20	22.5	27.7	20.8

Table 16: Intentions vs. Matching States Relations

		States							
Intention	Total State Relations	1	2	3	4	5	6	7	8
1	10					🟩	🟩	🟩	🟩
2	16	🟩		🟩		🟩		🟩	
3	6	🟩	🟩						
4	20							🟩	🟩
5	6	🟩	🟩	🟩	🟩				

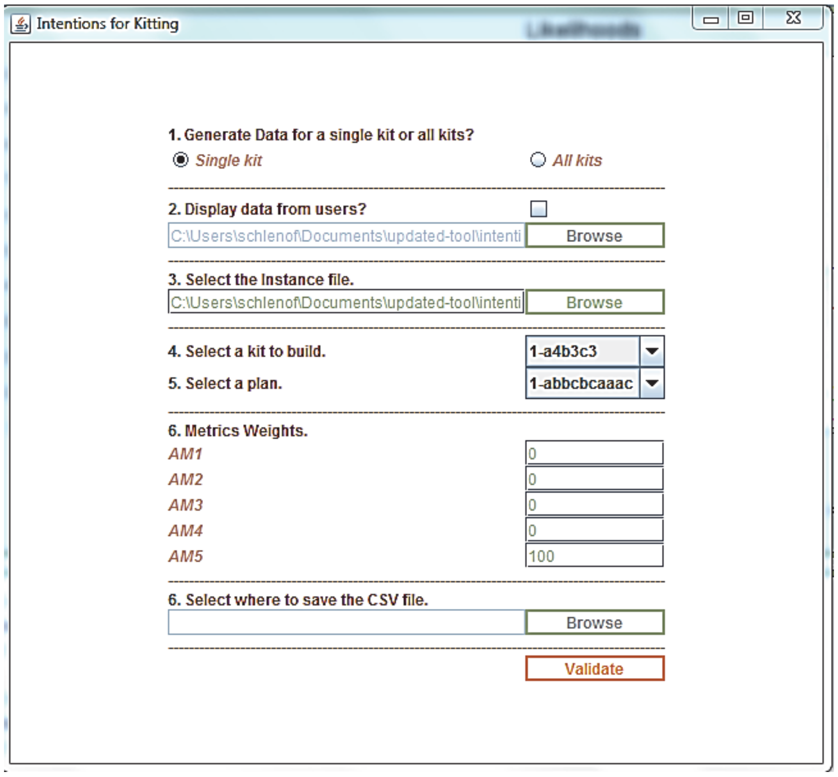
It is interesting to look at Table 15 and compare it to data presented in Table 16 (same as Table 13 but included here for easy reference). As mentioned earlier, Approach 1 (the Bayesian Approach) was not aligned with this simple example and thus is not discussed in detail in this section. Approach 2 (Quantity of True State Relations) bumped up the likelihood of the intentions that had the most true state relations, independent of how many state relations composed the intentions. Intentions 1, 2, and 5 all had four true state relations, thus they had the highest likelihood. Ironically, from a percentage perspective, the first two of these intentions had a relatively low percentage of their overall state relations completed, but this was not accounted for in this approach. The next approach (Approach 3) looked specifically at the percentage of the intention that was complete. When combined with the other approaches, it bumped Intention 5 up to the top, since it had the highest percentage of the intention complete (67%), even though the true state relations happened earlier in the process and no true state relations occurred in the last four states. The last approach (Approach 3) explored which state relations happen most recently. When applying this approach, Intention 1 jumped to the top by a significant amount (when combined with the other approaches) because it had four true state relations in the last four states, while no other intention had more than two.

Clearly, the weights assigned to these metrics played a very big factor in their effect. Different weights have different strengths of impacts. The determination of the proper weights is more of an art than a science, and will take some trial and error. Chapter 6 of this thesis starts to explore the optimum combination of weights for the manufacturing kitting domain. It will likely change with different domains. The purpose of this section was to explore the various metrics that were being considered and their impact on a sample intention being performed in the environment.

4.2.4.7. A Tool to Adjust Weights

To facilitate the process of determining the impact of the intention recognition approaches' weights on the output of the Intention Recognition Algorithm, a Java-based tool was developed. This tool allows the user to specify the weights associated with each approach and to generate a line chart showing the results of the overall Intention

Recognition Algorithm for each kit and for each state during the run. The interface for the tool is shown in Figure 40.


Intentions for Kitting

1. Generate Data for a single kit or all kits?
[✓ Single kit] All kits

2. Display data from users? ☐
C:\Users\schlenof\Documents\updated-tool\intenti Browse

3. Select the Instance file.
C:\Users\schlenof\Documents\updated-tool\intenti Browse

4. Select a kit to build. 1-a4b3c3

5. Select a plan. 1-abbcbcaaac

6. Metrics Weights.

AM1	0
AM2	0
AM3	0
AM4	0
AM5	100

6. Select where to save the CSV file.
Browse

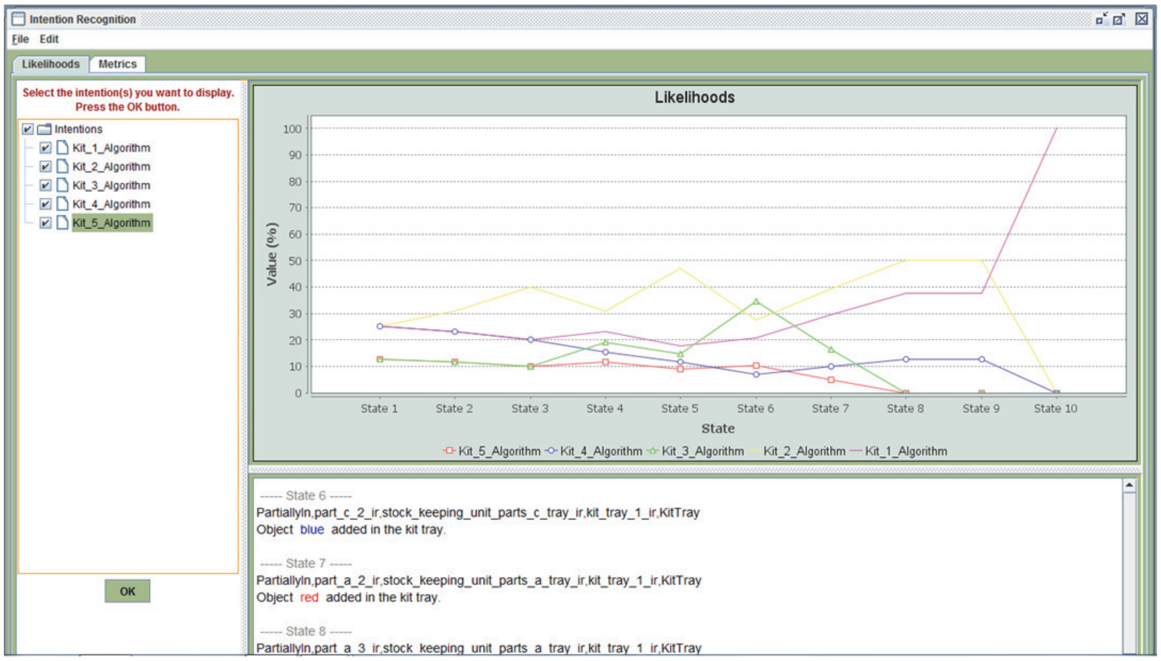
Validate

Figure 40: Kitting Intention Tool Interface

The interface provides the option for the user to run the approaches and associated weights over a single kit or for all kits. If the user selects all kits, the information will be saved in a comma separated value (csv) file in the location specified in the field at the bottom of the screen. If the user specifies only one kit, the results will be provided in a separate pop-up window in a set of line charts. The user also has a checkbox to indicate whether the comparable data provided by the user should be shown on the same line graph. Having the user data on the line graph allows for the direct comparison between the results of the human and the results of the algorithm. This option was chosen for some of the graphs shown in the next chapter. The next field asks the user to specify the location of the ontology instance file. This instance file is needed as input to the Intention Recognition Algorithm.

Items 4 and 5 in Figure 40 allow the user to specify the kit (1 through 5) that represents the intention and the plan (1 through 5) that represents the sequence (plan) in which the kit is created. There are five unique plans (arbitrarily chosen) for each kit. Lastly, in item 6, the user can specify the weights that s/he wants to associate with each intention recognition approach. These can be any value greater than or equal to zero. In the figure, because every intention recognition approach has a weight of zero except for the last approach (the Bayesian approach), only that approach is being used. Note that the interface was set up to allow for five different approaches to be used, but only four were implemented.

The output of the tool is shown in Figure 41. In this figure, the user results are not presented, but could have been if the option was chosen in the tool interface. The left pane gives the user an option of which kit intention to display. Each kit intention is represented as a line in the top right pane, with a key to show which color represents which kit intention. There are ten states in each plan (as shown in the x-axis), and the likelihood of each intention (as shown on the y-axis) is presented for each kit at each state. For example, if we look at Kit 1, which is represented by the purple line, the algorithm shows that there is a 25% chance that the kit is being created in State 1, 22% in State 2, and so on until State 10 where the likelihood is 100%. The bottom right pane gives a textual description of what happened in each state (e.g., Object blue added to the kit tray in State 6).



```
----- State 6 -----
PartiallyIn,part_c_2_ir,stock_keeping_unit_parts_c_tray_ir,kit_tray_1_ir,KitTray
Object  blue  added in the kit tray.

----- State 7 -----
PartiallyIn,part_a_2_ir,stock_keeping_unit_parts_a_tray_ir,kit_tray_1_ir,KitTray
Object  red  added in the kit tray.

----- State 8 -----
PartiallyIn,part a 3 ir,stock keeping unit parts a tray ir,kit tray 1 ir,KitTray
```

Figure 41: Output of the Intention Recognition Tool

In the next chapter, we will take a detailed look at applying the main Bayesian approach to various kitting intentions and use this tool to show some results of the experiment.

5. System Architecture

In this chapter, we discuss the results of applying the techniques presented in this thesis. We will start by describing the overall architecture that was used to recognize states in the environment and to infer intentions.

The overall system architecture that was used for the Intention Recognition Algorithm can be found in Figure 42. This is the same architecture that was presented in the introduction of this thesis, with more detail provided. A recap of flow of information is provided here while a detailed description of how this was implemented for the experiment is described in the next few sections.

A kitting simulation environment was created in the simulation tool. The robotic arm was meant to simulate a human developing a kit. The arm ran through a kit development plan that was not made known to the state recognition or intention recognition systems. The only information that was provided out of the simulation system was through the USARTruth system, which provided information about the coordinates, orientation, and dimensions of the objects in the environment. Taking the information as input, the State Recognition Algorithms were constantly assessing the truth value of the state relations of interest to the domain using the information stored in the state relations ontology and the kitting ontology. The state relations of interest were determined based on the domain, as discussed earlier in this thesis, and captured in the ontology. The results of the State Recognition Algorithms were output in a tabular format when a state relation of interest changed it truth value. The results of the state relation algorithms, along with the pre-defined intentions captured in the intention ontology, were fed as input into the Intention Recognition Algorithm. The Intention Recognition Algorithm used metrics to match the state relations perceived in the environment to the intentions that were predefined in the algorithms. The output of the Intention Recognition Algorithm was a set of likelihoods, which show which intentions were most likely to be occurring in the environment at a given time.

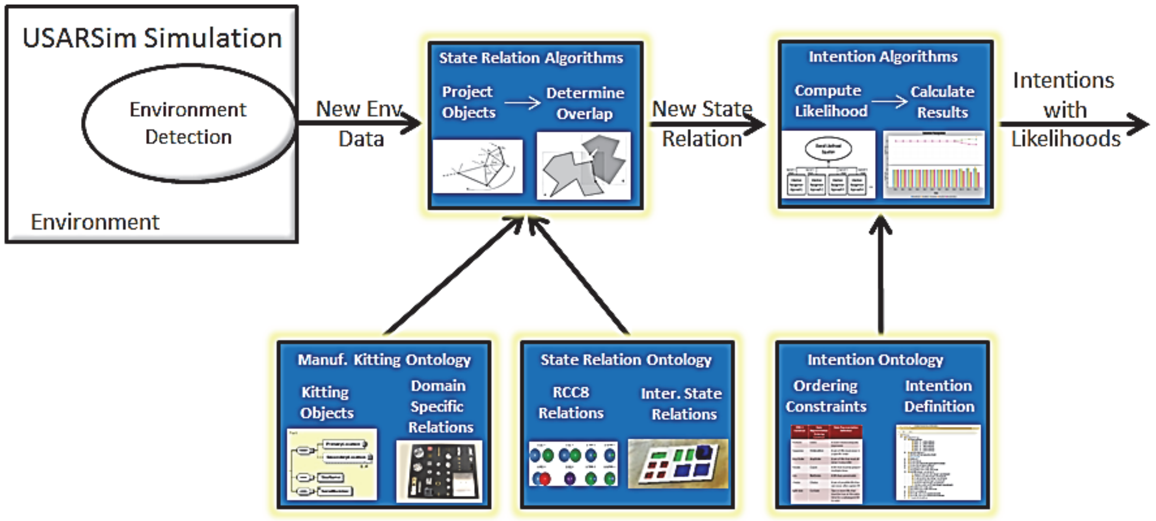


Figure 42: Experimentation Architecture

A detailed description of each box above is provided below.

5.1. USARSim

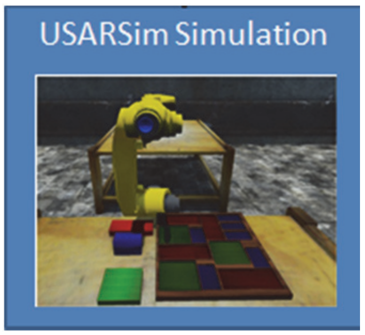


Figure 43: USARSim Kitting Environment

USARSim (Unified System for Automation and Robot Simulation) [151] [152] was originally designed as a high-fidelity simulation of Urban Search And Rescue robots and environments intended as a research tool for the study of Human-Robot Interaction (HRI) and multi-robots coordination. USARSim is an open source simulation environment based on Epic Games Unreal Tournament 2004. Since its initial release, USARSim has been expanded to support many diverse environments including manufacturing robots, highway robots, the DARPA urban challenge, robotic soccer, submarines, humanoids, and helicopters. USARSim is designed as a simulation companion to the National Institute of Standards and Technology's (NIST) Reference Test Facility for Autonomous Mobile Robots for Urban Search and Rescue [153].

USARSim utilizes the Karma Physics engine and high-quality 3D rendering facilities of the Unreal game engine to create a realistic simulation environment that provides the embodiment of a robotic system. The current release of the USARSim consists of various environmental models, models of commercial and experimental robots, and sensor models. High fidelity at low cost is made possible by building the simulation on top of a game engine. In the development of this simulation environment, mcuch effort was devoted to the robotics-specific tasks of modeling platforms, control systems, sensors, interface tools and environments. These tasks are in turn, accelerated by the advanced editing and development tools integrated with the game engine leading to a virtuous spiral in which a wide range of platforms could be modeled with greater fidelity in short time.

For this experiment, a manufacturing kitting environment was developed, as shown in Figure 43.

This environment consisted of:

- A robot arm (which was meant to represent the arm of a human);
- One of five kit trays composed of colored areas to represent the type and placement of various parts;
- A series of parts of different size and color.

The part shapes in the simulation were generated from models created in the Blender tool.[8] Blender provides a broad spectrum of modeling, texturing, lighting, animation and video post-processing functionality.

The robot arm had a vacuum effector at the end which it used to pick up the parts. The suction for the gripper was turned on when the effector was first in contact with the part, and was then turned off after the part was in its final location.

[8] http://www.blender.org/

5.2. USARTruth

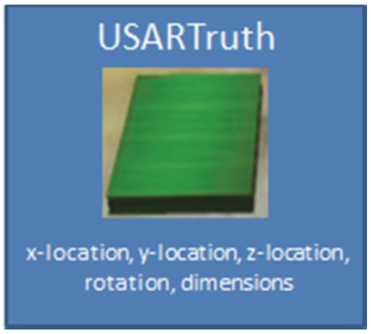


Figure 44: USARTruth Channel

Information about the location (x, y, and, z coordinates) and orientation (roll, pitch, and yaw) of each object in the simulation was provided by the USARTruth channel within USARSim. USARTruth is capable of reading out information on objects in USARSim by connecting as a client to TCP socket port 3989. The simulator USARTruth-Connection object listens for incoming connections on the port and receives queries over a socket in the form of strings formatted into key-value pairs. The USARTruth connection accepts two different keys, \class" and \name," which are both optional. When USARSim receives a new string over the connection, it sends a sequence of key-value formatted strings back over the socket, one for each Unreal Engine Actor object that matches the requested class and object names. An example of the strings returned by USARSim is given below along with a description for each key.

[Name P3AT_0] [Class P3AT] [Time 29.9739] [Location 0.67,2.30,1.86] [Rotation -0.00,0.46,0.00]

The variables are as follows:

- **Name**- the name of the object;
- **Class**- the class type of the object;
- **Time**- the time that the information was returned;
- **Location**- the x, y, and z coordinates of the center of gravity of the object;
- **Rotation**- the roll, pitch, and yaw of the object.

For this work, the frequency of these updates was twice per second, though this value could be changed as necessary.

One could either specify the specific objects they want information about by adding an argument (e.g., USARTruth.exe -s [Actor]) or the system could return all objects in the simulation if no argument was specified.

In addition to the location and orientation of objects, additional information was needed about the dimension of the objects. The x-, y-, and z-dimensions of the objects were stored within the kitting ontology. Using the location information provided by USARTruth and the information about the dimensions of the object from the ontology, information about the space in which an object occupies could be determined.

5.3. MySQL Database

The MySQL database was developed to address some of the shortcomings of ontology-based approaches. Ontologies are very strong at representing detailed information about objects in the environment, but do not lend themselves well to real-time applications requiring frequent updates of information. It is desirable to represent the dynamic information in a dynamic database. For this reason, a technique was developed to: 1)automatically generate tables for storing data; and 2)access functions for obtainingthe data from the ontology in a MySQL database. [154]

Reading data to and from the MySQL database instead of the ontology file offers the easier access to live data structure. Furthermore, it is more practical to modify the information stored in a database than that stored in an ontology, which in some cases, requires the deletion and re-creation of the whole file. A literature review reveals many efforts and methodologies that have been designed to produce SQL databases from ontologies. Our effort builds upon the work of Astrova et al. [155].

In addition to generating and filling the database tables, tools were created that automatically generate a set of C++ classes for reading and writing information to the kitting MySQL database. The Generator tool is a graphical user interface developed in Java, allowing the user to transfer and store data from OWL files into a MySQL database. This tool also permits the user to query the database using the C++ function calls. The Generator tool is composed of the following functionalities:

- Convert OWL documents (ontologies) into SQL syntax (OWL to SQL);
- Translate SQL syntax to OWL language in order to modify an OWL document (SQL to OWL);
- Convert the OWL language into C++ classes (OWL to C++).

In order to generate the SQL database and C++ classes, the OWL object model must be mapped to the C++ object model and the relational SQL model. Therefore, the notions of single-valued and multi-valued properties as well as the inheritance must be mapped from the ontology to the SQL database and C++ classes. The mapping from OWL proceeds as follows:

- **Data properties-** In an ontology, data properties link an individual to a data value. Single-valued data properties are mapped into a SQL table entry or C++ class variable with the corresponding type of the original property. For example, in the manufacturing kitting ontology, a robot has a single-valued data property *hasRobot_Description*, represented in the SQL database as a variable character (*varchar)* and in the corresponding C++ class as *std:string*. Multi-valued data properties are mapped from the ontology into the SQL database as a table and into the C++ class as a *std:vector* with the corresponding type of the original property. For example, in the ontology, a stock-keeping unit has a multi-valued data property *hasSku_EndEffectorRefs*. This maps to a SQL table containing *varchar* entries and the C++ *std::vector<std::string>* in the corresponding C++ class;

- **Object property**- In an ontology, object properties link one individual to another individual. The single-valued object properties are mapped to a SQL table entry or C++ class variable. Their type is a pointer to the range of the object properties. For example, in the manufacturing kitting ontology a solid object has the object property *hasRobot_Description* linking it to a physical location. In the SQL database, we use a foreign key to link the two entries. In the C++ classes, this is represented by a reference to a physical location: *PhysicalLocation* hasSolidObject_PrimaryLocation*. Multi-valued object properties are mapped from the ontology into the SQL database as a table and into the C++ class as a *std:vector* of pointers referencing objects of the range of the property. For example, a solid object also has a list of secondary locations corresponding to a multi-valued object property in the ontology: *std::vector <PhysicalLocation*> hasSolidObject_SecondaryLocation.*

5.3.1. MySQL Database Generation

This section provides basic information on the Generator Java tool. Converting an OWL ontology to SQL script files is easily performed using the *OWL to SQL* tab (Figure 45). The required fields are:

- **Ontology Path**- This is the OWL file to be converted. Note that all *Import* statements in this file must use absolute paths;
- **Saves Path**- This is the directory where you want to save the SQL files.

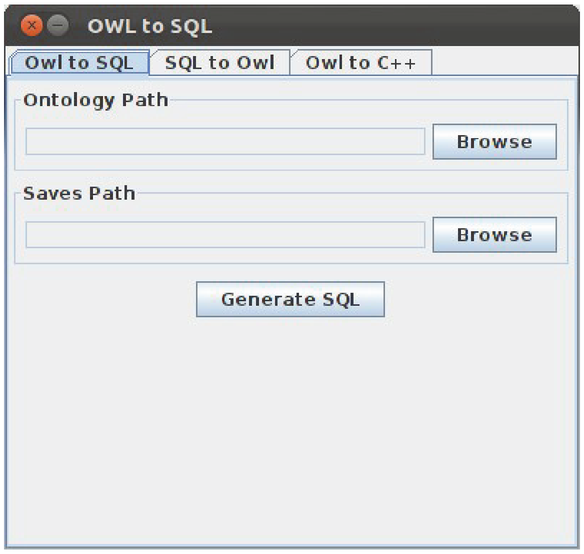


Figure 45: OWL to SQL Tab

Clicking on the ``Generate SQL'' will generate the SQL script files. Two files will be created by the tool:

- ***owlCreateTable.sql***- This is the file used to create tables in the database: *<inputfile>;*
- ***owlInsertInto.sql*** This is the file used to populate the database tables: *<inputfile>.*

These files may then be used with the SQL command line interface to create and populate the database.

5.3.2. C++ Class Generation and Usage

As previously mentioned, the C++ classes were automatically generated by the Generator tool. In addition to the class structure, Data Access Objects (DAOs) that are needed to interact with the MySQL database were generated. To map the MySQL database and indirectly the ontology to C++ classes, both the C++ classes and the DAO must be generated.

The C++ class files (.cpp) and header files (.h) are generated in a two-step process.
The first step does not depend on the content of the ontology, it only initializes the specific objects related to the MySQL connector driver. The second step generates all the C++ headers and class files relative to the ontology. All of the *include* statements are made directly in the C++ class files; and only forward declarations are performed in the headers. This resolves problems associated with circular includes or multiple includes.

The actual data access was provided through the use of a Data Access Object (DAO). A DAO provides an abstract interface to some type of database or other persistence mechanism. A DAO maps application calls to the database or persistence mechanism, thus providing some specific data operations without exposing details of the database. The use of the DAO separates the data accesses that the application needs from how these needs could be satisfied with a specific Database Management System (DBMS), database schema, etc. The different methods of the DAO are the same for any ontology. The concern here is not about the data, but only about the way to retrieve or store it.

When the DAO is generated, four vectors are built as follows:

- A structure with the SQL query to select the characteristics of an entity is created. The table relative to the entity itself and the ones relative to its super classes are queried;
- A structure with the SQL query to select multi-valued attributes (multi-valued data) for a given entity is created;
- A structure with the names of the tables linked to this entity in the ontology is created;
- A structure with the names of the association tables linked to an object is created.

With these four structures, one is able to read (*get* method) and write (*set* method) data from and to the MySQL database. The *get* method fills a C++ map and gets the object itself, while the *copy* method handles the data. The *set* method is called with a C++ map containing the values of the different attributes as input and writes these values into the MySQL database.

5.3.3. C++ Classes to Access Data from the MySQL Database

Figure 46 depicts an example using the generated classes to retrieve the location of the kit tray *kit_tray_name* from the MySQL database. The different sections of the example are described below:

- **Lines 1-4-** Includes the different headers necessary to query MySQL tables. Here, the tables Point, PoseLocation, Vector, and KitTray are required;
- **Line 9-** Initializes an object from the class *KitTray* by passing its name;
- **Line 10-** Allows access to any data from the table KitTray;
- **Lines 12-13-** Initializes an object of type *PoseLocation* and allows access to any data from the table PoseLocation;
- **Lines 18-19-** Retrieves X, Y, and Z coordinates from the table Point for the kit tray *kit_tray_name*;
- **Lines 22-23-** Retrieves the X axis vector (X_i, X_j, X_k) from the table Vector for the kit tray *kit_tray_name*;

- **Lines 26-27-** Retrieves the Y axis vector (Y_i, Y_j, Y_k) from the table Vector for the kit tray *kit_tray_name*.

```
 1. #include "Point.h"
 2. #include "PoseLocation.h"
 3. #include "Vector.h"
 4. #include "KitTray.h"
 5.
 6. void CanonicalRobotCommand::
 7.  getKitTrayLocation(string kit_tray_name){
 8.
 9.   KitTray* kit_tray = new KitTray(kit_tray_name);
10.   kit_tray->get(kit_tray_name);
11.
12.   PoseLocation* kit_tray_pose = new PoseLocation(
13.   kit_tray->gethasSolidObject_PrimaryLocation()->
14.                getname());
15.   kit_tray_pose->get(kit_tray_pose->getname());
16.
17.   //--Retrieve hasPoseLocation_Point
18.   Point * kit_tray_point =
19.   kit_tray_pose->gethasPoseLocation_Point();
20.
21.   //--Retrieve hasPoseLocation_XAxis
22.   Vector * kit_tray_x_axis  =
23.   kit_tray_pose->gethasPoseLocation_XAxis();
24.
25.   //--Retrieve hasPoseLocation_ZAxis
26.   Vector * kit_tray_z_axis  =
27.   kit_tray_pose->gethasPoseLocation_ZAxis();
28. }
```

Figure 46: Example Using the Generated C++ Classes

5.4. State Relation and Kitting Ontology

Figure 47: State Relation and Kitting Ontologies

The state relations and kitting ontology discussed in Chapter 3 were used during the experiment discussed in this chapter. Instances of certain classes were created in the kitting ontology, including:

- Parts:
    - 18 instances of Part A (red in Figure 48);
    - 12 instances of Part B (green in Figure 48);
    - 8 instances of Part C (blue in Figure 48);
    - 9 instances of Part D (dark yellow in Figure 48);
    - 6 instances of Part E (light yellow in Figure 48);
- Five unique instances of kits, containing the following configurations (also shown in Figure 48):
    - Kit 1- two Part A's, three Part B's, three Part C's, one Part D, one Part E;
    - Kit 2- two Part A's, three Part B's, five Part C's;
    - Kit 3- four Part A's, three Part B's, two Part C's, one Part D;
    - Kit 4- four Part A's, three Part B's, three Part C's;
    - Kit 5- four Part A's, four Part B's, two Part C's;

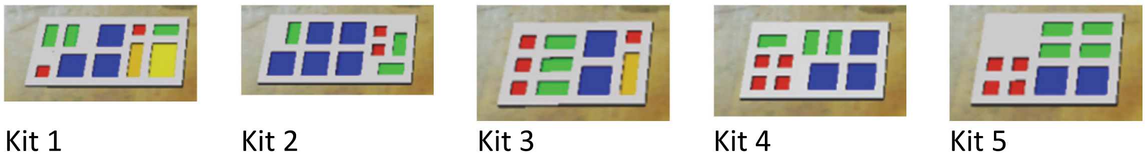


Figure 48: Five Possible Kits (Intentions)

- Five unique instances of part trays, each holding a different type of part;
- Five unique instances of kit trays, one for each kit that could be created;
- An instance of a kitting workstation, where the kitting operation was taking place;
- An instance of a work table;
- An instance of a robot (representing a human arm for this work);
- Two instances of a large container, one representing an empty kit tray box (where the empty kit trays were retrieved from) and one representing a finished kit box (where the finished kit were placed).

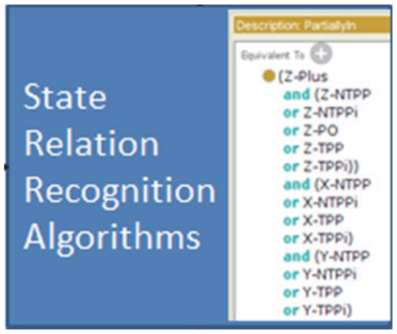
Similarly, specific classes were created in the state relation ontology to match the RCC state relations, intermediate state relations, and predicates. Only the relevant relations to the domain in the experiment were included.

- RCC8 Relations:
    - xy-DC, xy-EC, xy-EQ, xy-NTPP, xy-NTPPi, xy-PO, xy-TPP, xy-TPPi;
    - yz-DC, yz-EC, yz-EQ, yz-NTPP, yz-NTPPi, yz-PO, yz-TPP, yz-TPPi;
    - xz-DC, xz-EC, xz-EQ, xz-NTPP, xz-NTPPi, xz-PO, xz-TPP, xz-TPPi;
    - x-Minus, x-Plus, y-Minus, y-Plus, z-Minus, z-Plus;
- Intermediate State Relations:
    - OnTopWithContact;
    - PartiallyIn;
    - UnderWithContact;

- Predicates:
    - Kit_Location_In_Large_Box_With_Empty_Kit_Trays;
    - Kit_Location_On_Worktable;
    - Part_Location_on_Table;
    - Part_Location_In_Kit;
    - Part_Location_In_Robot_Gripper;
    - Robot_Gripper_Empty;
    - Robot_Holding_Part;
    - Worktable_Empty.

Within the state relation classes, the logical formulas are captured as data properties for the relevant class. An example of this is shown in Figure 49 for the PartiallyIn intermediate state relation.

5.5. State Recognition Algorithms



```
(Z-Plus
 and (Z-NTPP
 or Z-NTPPi
 or Z-PO
 or Z-TPP
 or Z-TPPi))
 and (X-NTPP
 or X-NTPPi
 or X-TPP
 or X-TPPi)
 and (Y-NTPP
 or Y-NTPPi
 or Y-TPP
 or Y-TPPi)
```

Figure 49: State Recognition Algorithms

The kitting and state relation ontology and the output from USARTruth were used as inputs to the State Recognition Algorithms. The information in the ontology was extracted using the OWL-API[9], which is an open source Java Application Programming

[9] http://owlapi.sourceforge.net/

Interface (API) and reference implementation for creating, manipulating and serializing OWL Ontologies. The latest version of the API is focused towards OWL 2, in which the ontology for this effort is represented. These state representation rules were then converted into Java code for evaluation. An example of this Java code for the Externally Connected RCC8 relation is shown in Figure 50.

```
if (equal(objsX.get(0).getX() + objsX.get(0).getWidth() / 2.0, objsX
            .get(1).getX() - objsX.get(1).getWidth() / 2.0))
      rcc.put("xy-EC", true);
if (equal(objsZ.get(0).getZ() + objsZ.get(0).getHeight() / 2.0, objsZ
            .get(1).getZ() - objsZ.get(1).getHeight() / 2.0))
      rcc.put("xz-EC", true);
if (equal(objsY.get(0).getY() + objsY.get(0).getDepth() / 2.0, objsY
            .get(1).getY() - objsY.get(1).getDepth() / 2.0))
      rcc.put("yz-EC", true);
```

Figure 50: Java Code for Evaluating Externally Connected RCC8 Relation

The Java code first used the input from USARTruth and the relevant dimension information of the objects to determine which RCC8 relations were true for any pair of relevant objects in the environment. Once new RCC8 relations were discovered, the intermediate state relations were evaluated, and were defined based on the RCC8 relations. As new intermediate state relations were evaluated as true, the domain-specific predicates were evaluated, which were based on the intermediate state relations. Postfix notation[10] is used for evaluating these expressions. This is a mathematical notation used to write down equations and other mathematical formulae. When postfix notation is used, no grouping elements, such as parentheses, are needed. The operations are noted after their arguments.

Though not shown in Figure 50, the code introduced a tolerance for evaluating the RCC8 relations. Due to inaccuracies in the simulation system and real perception systems, a two centimeter tolerance was introduced. In order to evaluate the RCC8 relation Externally_Connected, if the values in the x-, y-, or z-dimensions were within two centimeters of each other (as opposed to being equal as shown in Figure 50), then the statement would evaluate to true. This value of two centimeters could be set to any appropriate value by the user, depending on the domain and the accuracy of the systems being used.

As mentioned earlier, this process was run twice per second as new input was received from USARTruth.

[10] http://simple.wikipedia.org/wiki/Postfix_notation

5.6. State Recognition Output

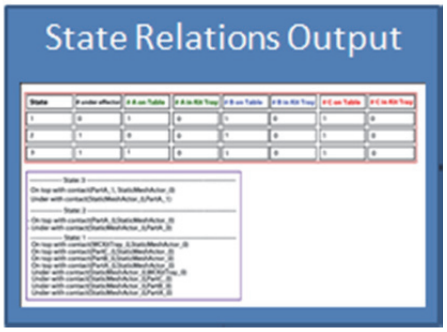
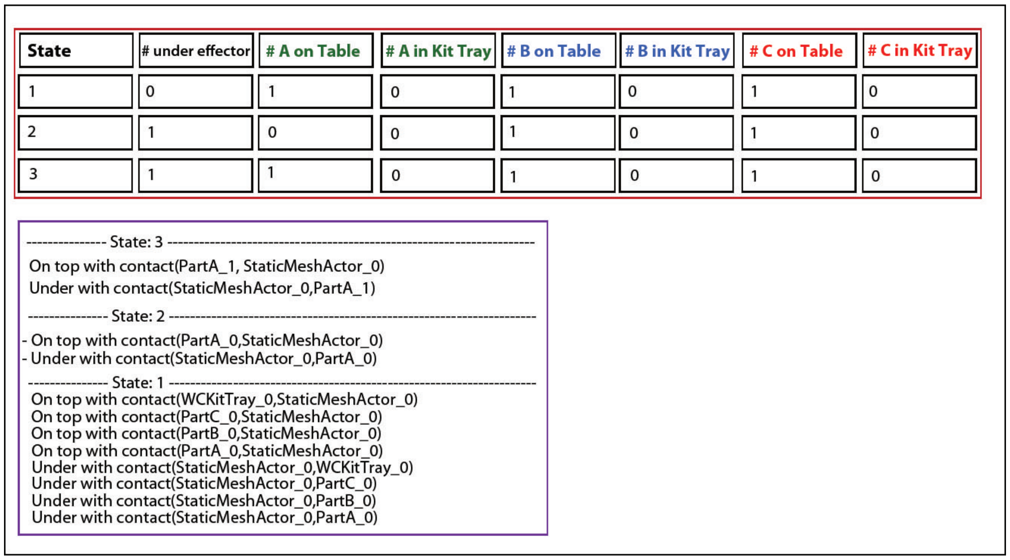

State Relations Output

Figure 51: State Relation Output

The output of the state relation algorithms could be visualized as a table and series of states, as shown in Figure 52.



State	# under effector	# A on Table	# A in Kit Tray	# B on Table	# B in Kit Tray	# C on Table	# C in Kit Tray
1	0	1	0	1	0	1	0
2	1	0	0	1	0	1	0
3	1	1	0	1	0	1	0

```
--------------- State: 3 --------------------------------------------------------------------
On top with contact(PartA_1, StaticMeshActor_0)
Under with contact(StaticMeshActor_0,PartA_1)
--------------- State: 2 --------------------------------------------------------------------
- On top with contact(PartA_0,StaticMeshActor_0)
- Under with contact(StaticMeshActor_0,PartA_0)
--------------- State: 1 --------------------------------------------------------------------
On top with contact(WCKitTray_0,StaticMeshActor_0)
On top with contact(PartC_0,StaticMeshActor_0)
On top with contact(PartB_0,StaticMeshActor_0)
On top with contact(PartA_0,StaticMeshActor_0)
Under with contact(StaticMeshActor_0,WCKitTray_0)
Under with contact(StaticMeshActor_0,PartC_0)
Under with contact(StaticMeshActor_0,PartB_0)
Under with contact(StaticMeshActor_0,PartA_0)
```

Figure 52: Results of State Recognition Algorithms

The top of Figure 52 shows the truth value of state relations of interest for the domain. Not all state relations of interest are shown in the table so the table can fit on the page. Each row in the table represents a state. A state in this table was created when a state relation of interest changes its truth value. For example, if a part went from being on the table to in (under) the robot vacuum effector, a new state was created. This is shown in the transition from State 1 to State 2. Originally, in State 1, Part A is shown as being on the table (as represented by a 1 in the cell under the heading "# A on Table").

In State 2, Part A is shown as being in the gripper (as represented by a 0 in the cell under the heading “# A on Table” and a 1 in the cell under the heading “# under Effector.”)

The first column in the table represents the sequential number of the state. Each subsequent column represents the truth value of that state relation described in the header of the column. State relations could be true more than once in a state. For example, there could be two Part A’s in a kit tray. When this happens, there would be the number 2 in the appropriate cell.

In addition to the tabular format, the system also outputted the information in a textual format. This is shown in the blue box under the table in Figure 52. As with the table, this textual description was updated every time a new state was created. State 1 at the bottom shows all of the state relations of interest that were true in that state. These include:

- On_Top_Of_With_Contact(WC_KitTray_0, StaticMeshActor_0);
- On_Top_Of_With_Contact(PartC_0, StaticMeshActor_0);
- On_Top_Of_With_Contact(PartB_0, StaticMeshActor_0);
- On_Top_Of_With_Contact(PartA_0, StaticMeshActor_0);
- Under_With_Contact(StaticMeshActor_0, WC_KitTray_0);
- Under_With_Contact(StaticMeshActor_0, PartC_0);
- Under_With_Contact(StaticMeshActor_0, PartB_0);
- Under_With_Contact(StaticMeshActor_0, PartA_0).

In the bullets above, the ‘_0’ after each attribute signifies the identifier for the instance of that class. In addition, the StaticMeshActor was the table. There are two ways of representing each state relation. One could either say that PartA was *on_top_of_with_contact* the table or that the table was *under_with_contact* PartA. Both state relations are true. For our work, we identify these redundant state relations, but only represent this once.

In subsequent states, only the state relations that change from the previous state were listed. If a state relation was true in state ***n***, it was also true in state ***n+1*** if it was not explicitly stated otherwise. If a state relation was no longer true, there will be a ‘-‘ before it. If a new state relation becomes true, it will be shown in State 2 just like all other true state relations in State 1. For State 2, the following state relations were true in State 1, but were no longer true in State 2:

- On_Top_Of_With_Contact(PartA_0, StaticMeshActor_0);
- Under_With_Contact(StaticMeshActor_0, PartA_0).

In addition, but not shown in the table, the relation Under_Effector(PartA_0, Effector_0 was also true).

In State 2, another instance of Part A appeared in the table while the previous instance of Part A was still in the robot effector, so the following two state relations became true:

- On_Top_Of_With_Contact(PartA_1, StaticMeshActor_0);
- Under_With_Contact(StaticMeshActor_0, PartA_1).

This table and associated textual descriptions would continue to grow until the kit building intention was complete.

5.7. Intention Ontology

Figure 53: Intention Ontology

The structure of the intention ontology was discussed in Section 4.1 and was used during the experiment discussed in this chapter. Instances of certain classes were created in the intention ontology, including:

- **Intentions**- there were five kitting intention instances developed, one of each kit design. They named to represent the number of each part in the kit, namely:
    - intention-kit-a2b3c3d1e1;
    - intention-kit-a2b3c5;
    - intention-kit-a4b3c2d1;
    - intention-kit-a4b3c3;
    - intention-kit-a4b4c2;
- **Ordering Construct Subclasses**- four ordering constructs subclasses were developed that were relevant to the kitting examples. They were:
    - AnyOrder;
    - Count;
    - Exists;
    - OrderedList;
- **Ordering Construct Instances**- Each of the ordering construct subclasses listed above have multiple instances which were used to describe the intentions as

described in Section 4.1.1. Each of the ordering construct instances bottom out at the state relations described above.

5.8. Intention Recognition Algorithm

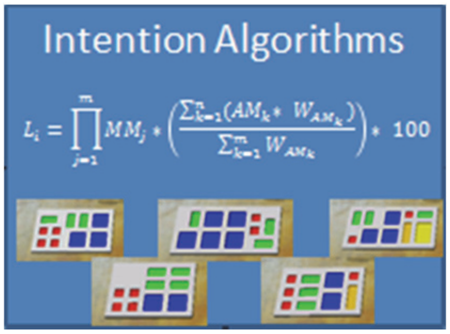

Intention Algorithms
$L_i = \prod_{j=1}^{m} MM_j * \left(\frac{\sum_{k=1}^{n}(AM_k * W_{AM_k})}{\sum_{k=1}^{m} W_{AM_k}}\right) * 100$

Figure 54: Intention Recognition Algorithm

The Intention Recognition Algorithm has a Java-based GUI front-end with Eclipse IDE (http://www.eclipse.org) that is capable of analyzing intention and ordering construct information to compute and display the likelihood of each intention based on input state relations. To achieve these tasks, the tool reads the ontology that captures information on each intention described for the kitting domain and stores this information in memory. Using input data (true state relations of interest) from the State Recognition Algorithms, the Intention Recognition Algorithm matches these state relations with those that constitute each intention. The output of this process computes and displays the likelihood of each intention at each state of the process. The following sections detail the aforementioned procedures.

5.8.1. Loading and Reading an Ontology

To query and access data in the ontology we use the OWL-API (http://owlapi.sourceforge.net/). The OWL-API is a Java API and reference implementation for creating, manipulating, and serializing OWL ontologies. Individuals of these classes are stored in Java data structures (list, vector, etc). The following snippet depicts how to load and access the intention class in the ontology.

```
// Define the IRI (Internationalized Resource Identifier) for the ontology
String ontology_IRI = "http://www.semanticweb.org/ontologies/2013/0/soap.owl";

// Get hold of an ontology manager
OWLOntologyManager manager = OWLManager.createOWLOntologyManager();

// Get hold of a data factory
OWLDataFactory factory = manager.getOWLDataFactory();

// Load the Ontology
manager.loadOntologyFromOntologyDocument(soap.owl).getImports();


// Access the Intention Class
OWLClass                                    intentionClass                                    =
factory.getOWLClass(IRI.create(ontology_IRI.concat("#Intention")));
```

5.8.2. Methodology to Compute Likelihoods

Figure 55 displays the methodology to compute and display intention likelihoods. To compute the likelihood for each intention, the Intention Recognition Algorithm received state relations information from the State Recognition Algorithms. The Intention Recognition Algorithm searched the intentions for which of these state relations held true and computed the likelihood for these intentions in each state. The constructs within an intention were organized through the ordering construct OrderedList. The algorithms first tried to match the input state relations with the ones that constitute the first element (which was also an ordering construct) of OrderedList. For instance, if an Intention *(intention1*) consists of an Exists condition (*exist1*) as its first element and an Ordering Construct (*anyorder1*) as its second element, the algorithms would first explore *exist1* to ensure that the condition is met. If none of the state relations satisfies the first element of *orderedlist1*, then the other elements of *orderedlist1* would not be checked. However, if the first element of *orderedlist1* was satisfied, the next set of input state relations would be tested against the second element of *orderedlist1*.

To date, the state relations coming out of the State Recognition Algorithms were written in a text file. This text file was processed by the Intention Recognition Algorithm.

```
procedure COMPUTE AND DISPLAY LIKELIHOODS
    load ontology
    parse OWL class Intention
    for each individual from Intention do
        create a Java object of type Intention
        for each Intention object do
            store ordering constructs
            store state relations
            store predicates
        end for
        for each state input $s$ do
            read input state relation
            for each state relation do
                search for the state relation in each intention object
                if state relation is found in intention $i$ then
                    for each intention $i$ do
                        for each additive metric $AM_j$ do
                            set the weight for $AM_j$ through the user interface
                            compute metric $AM_{j,i,s}$
                        end for
                        set the weight for $MM_1$ through the user interface
                        compute metric $MM_{1,i,s}$
                    end for
                end if
            end for
            compute likelihood
            display likelihood in user interface
        end for
    end for
end procedure
```
1

Figure 55: Methodology to Compute and Display Likelihoods

To represent the likelihoods, the Intention Recognition Algorithm encoded each approach in its own Java class. Each approach required its own weight (integers) that could be adjustable by the user. As such, the user interface included a text field for each approach. The weights could be assigned before running the algorithms and after the algorithms were completed. At the end of the process, the Intention Recognition Algorithm displayed the likelihoods computed with the weights assigned at the beginning of the simulation. Once the simulation ended, the user could change the weights to determine their impact on the results without running the simulation from the beginning.

5.9. Intention Likelihoods

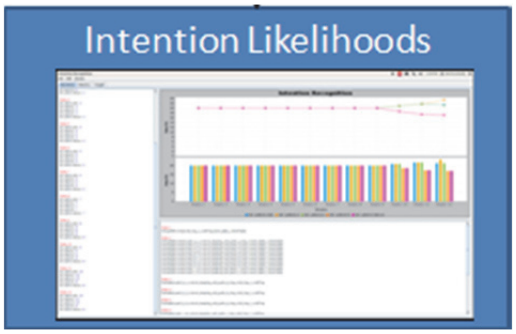


Figure 56: Intention Likelihoods

The likelihood for each intention is displayed as graphs using the free, Java-based JFreeChart (http://www.jfree.org/jfreechart/) library. Figure 57 depicts the likelihood for each intention for each state of the 12 states presented. The colors of the lines and the bars correspond to individual intentions, as shown at the bottom of the figure. Both graphs (bar and line) displayed the same result in different formats. As shown, before State 10, the likelihoods for each intention (5 intentions in this case) were the same. From State 10 to State 12, the likelihoods branch out for some intentions. This is due to the existence of new state relations in these states that were part of some intentions and not part of others.

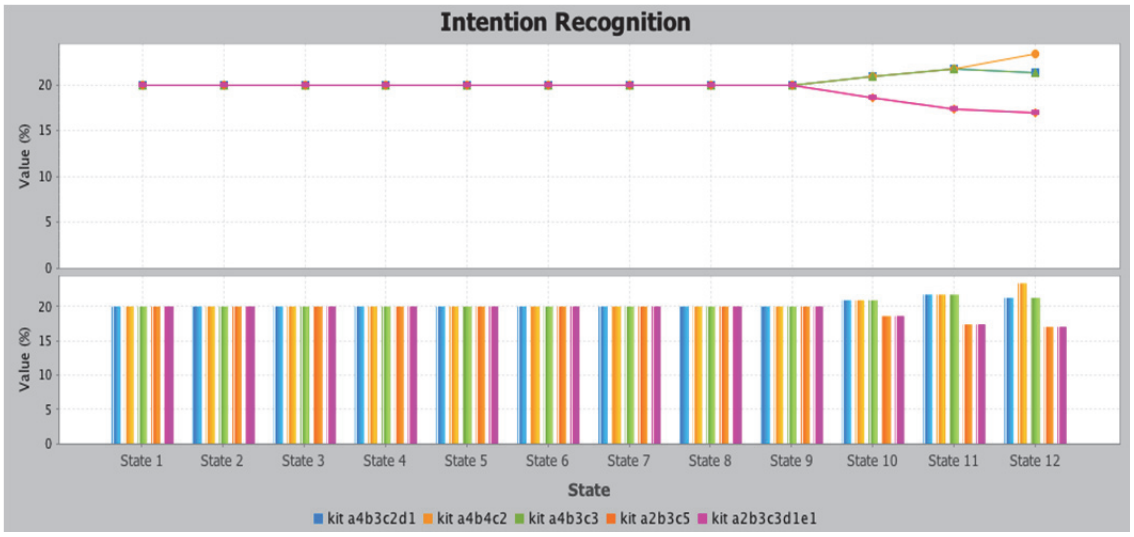


Figure 57: Likelihoods for Each Intention at Different States

In the next chapter, we will describe an experiment that uses the infrastructure described in the chapter to validate the overall intention recognition approach presented in this thesis.

6. Intention Recognition Experiment

An experiment was run to assess the performance of the state-based Intention Recognition Algorithm described above. This was done by comparing the output of our algorithm to the performance of several humans watching agents performing the same intentions. This section described the experimental procedures and the results.

6.1. Experimental Procedures

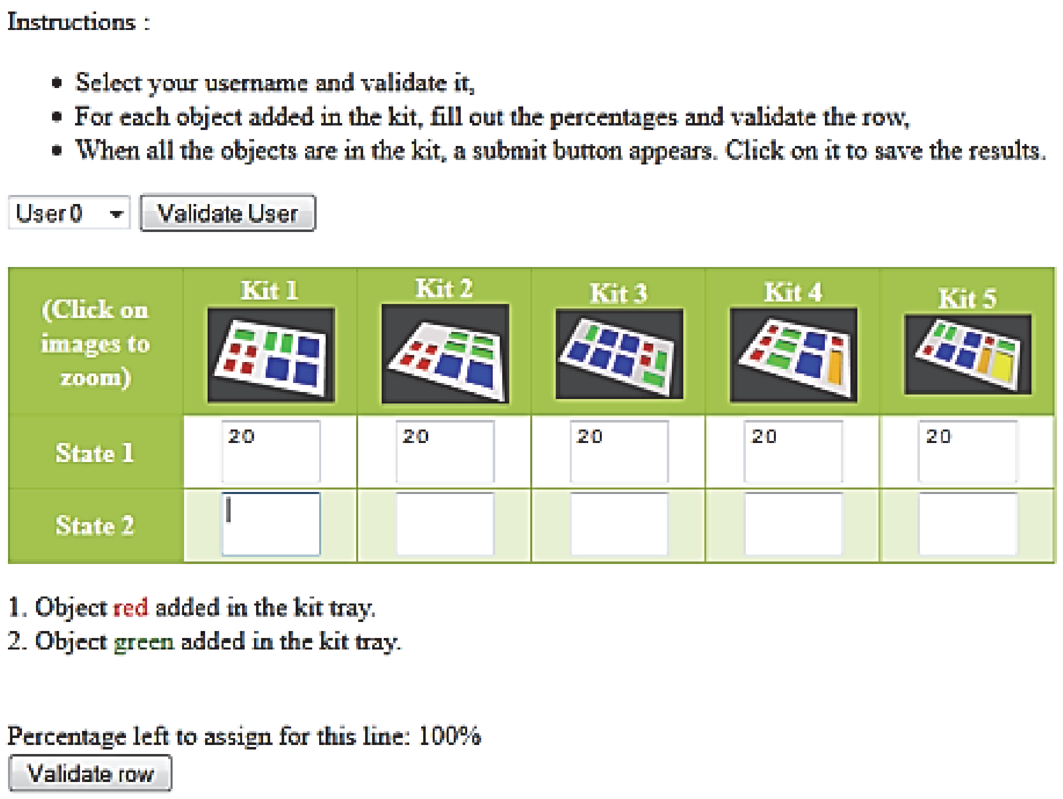
Since there are no clear metrics to assess the performance of intention recognition algorithms, an experiment was developed to compare the results of the Intention Recognition Algorithm to the performance of humans [156].

Five kits were used that each contained ten blocks (as shown in Figure 58), but each contained a unique combination of different-shaped blocks. In the figure, blocks of different sizes were represented by different colors. The combination of blocks was carefully chosen to provide an appropriate level of challenge to the humans and the algorithm. We used a distance metrics (specifically the L1 distance) to characterize the difference between kits. In this case, the L1 distance represents the sum of the absolute differences between the number of each colored part between two kits. For example, Kit 4 has four red blocks, two blue blocks, three green blocks, zero yellow blocks, and one orange block. Kit 5 has two red blocks, three blue blocks, three green blocks, one yellow block, and one orange block. The distance metric between these two kits would be:

$$|4-2| + |2-3| + |3-3| + |0-1| + |1-1| = 4 \qquad (97)$$

The smallest the distance measure could be zero if both kits were identical and the largest is 20 if there were no common blocks between the two. For this work, we chose values between two and six to represent challenging cases where any two kits were very similar but there were enough difference for an intention recognition system to be able to distinguish between the two.

Using these kits, a set of simulations was developed (in USARSim [157]) capturing the process in which one could assemble these kits. For a kit with five types of parts and ten total instances of parts, there are approximately 5^{10} (almost 10 million) possible orders in which the parts can be placed in the kit tray. For this experiment, we randomly chose five different orders for each of the five kits, resulting in 25 total runs (shown in Appendix B). At each state, the State Recognition Algorithm (described in Chapter 3) was run to identify the state relations in the environment. Based on these state relations, the Intention Recognition Algorithm in Chapter 4 was run to assign likelihoods to intentions. The resulting likelihoods were captured for each possible intention (kit) at each state.


Instructions :

- Select your username and validate it,
- For each object added in the kit, fill out the percentages and validate the row,
- When all the objects are in the kit, a submit button appears. Click on it to save the results.

User 0 Validate User

(Click on images to zoom)	Kit 1	Kit 2	Kit 3	Kit 4	Kit 5
State 1	20	20	20	20	20
State 2					

1. Object red added in the kit tray.
2. Object green added in the kit tray.

Percentage left to assign for this line: 100%

Validate row

Figure 58. Experiment User Interface

To determine how well these likelihoods compared to what a human would perceive in the same situation, we built a data set based on these same five kits. To compile the data set, 15 students served as the human participants. They were considered novice users. They were presented with the interface shown in Figure 58. Included in the interface were five images of kits as shown in the figure. The kit images could be enlarged by clicking on them. Each kit represents an intention. Students were not told how many of each kit were being built in each of the 25 runs. All kits were presented in random order and the order varied from participant to participant.

One by one, the human was presented with a textual description of something that happened in the environment, for example, “Red object added in the kit tray.” Based on this information, the human assigned likelihoods as to which kit was being developed (i.e., what the intention was based on the observed events). The likelihoods that the human assigned were based on personal preferences. In the example shown in Figure 58, the human assigned a 20% likelihood to all kits based on the first state because a red object exists in all kit trays. In some cases, the human scaled the likelihood based upon how many red objects existed in each kit tray. The only rule was that all likelihoods for each state must sum to 100%. An update of how many percentage points were available to assign to each row is shown at the bottom of the figure. Once a human finished with a given row, he pressed the “Validate Row” button at the bottom. If the sum of the percentages added to 100%, a new environment observation was provided, a new row was added to the table, and the process started again. If they did not equal 100%, the human was alerted to this and asked to change their percentages until they equalled 100%. Once a human clicked the “Validate Row” button and it successfully validated, they could not go back and edit the previous row. Because there were exactly ten objects in each kit, exactly ten states were presented per kit. After ten states were completed, the human moved on to the next intention until all 25 intentions were complete.

To compare these results to the output of the Intention Recognition Algorithm, we averaged the likelihoods for all 15 humans for each plan, each state, and each kit. Based upon this, we identified two performance metrics for evaluation:

1. **The average state in which the humans first correctly identified the kit that was being created (and consistently identified that correct kit for the remainder of the states)-** This was measured by averaging likelihoods that all humans assigned to each kit within a state, and then determining if the kit with the highest likelihood was the correct kit being created;
2. **The average state in which the humans were over 20 percentage points more confident that the correct kit was being developed compared to the second most probable kit:** The 20 percentage point value can be changed as necessary, but appeared to be a reasonable value to show that the humans were confident that their choice was correct.

Table 17: Detailed Human Intention Recognition Data

Kit	Plan	State	Kit 1	Kit 2	Kit 3	Kit 4	Kit 5	Highest Probability Kit	Correct Kit	Correct Intention First Chosen	Over 20% Confident of Intention
1	5	1	21.6	21.7	17.8	21.2	17.8	2	NO	NO	NO
1	5	2	22.9	22.5	16.1	22.3	16.1	1	YES	NO	NO
1	5	3	20.8	19.7	21.8	19.6	18.0	3	NO	NO	NO
1	5	4	22.9	17.4	25	17.2	17.4	3	NO	NO	NO
1	5	5	39.7	29.6	1.7	27.3	1.7	1	YES	YES	NO
1	5	6	36.9	29.0	4.4	25.2	4.4	1	YES	YES	NO
1	5	7	36.7	30.2	4.1	24.8	4.13	1	YES	YES	NO
1	5	8	76.1	2.6	10.0	5.9	5.4	1	YES	YES	YES
1	5	9	81.3	8.06	1.5	7.7	1.5	1	YES	YES	YES
1	5	10	78.0	9.0	4.0	5.0	4.0	1	YES	YES	YES

To explain these two metrics, we refer to Table 17. This is actual data from one part of the experiment, in which Kit 1 was being developed using Plan 5 (as shown in the first two columns). All plans are included in Appendix B, but this specific plan is included below for reference.

Kit 1, Plan 5

1. Red object in the kit tray;
2. Red object in the kit tray;
3. Blue object in the kit tray;
4. Blue object in the kit tray;
5. Red object in the kit tray;
6. Green object in the kit tray;
7. Green object in the kit tray;
8. Blue object in the kit tray;
9. Red object in the kit tray;
10. Green object in the kit tray.

For each state, the average of all 15 humans' likelihoods associated with each kit is shown in the blue columns in the table. For example, in State 1, which corresponded to a red object being in the kit tray, the average of the humans' likelihoods for Kit 1 was 21.6, which means that on average, after the humans observed that the red object was in the kit tray, they thought that there was a 21.6% chance that the Kit 1 intention was being performed. Likewise, they thought there was a 21.7% chance that the Kit 2 intention was being performed, and so on. The table shows the likelihoods for every kit after every state for the duration of the experiment (Kit 1, Plan 5). The next column (highest likelihood kit) shows which kit (1-5) had the highest likelihood for each state. The next column (Correct Kit) indicates whether the highest likelihood kit is the kit intention that was actually being performed. In the example, because this part of the experiment focuses on Kit 1, this column will have a "yes" when the highest likelihood kit was Kit 1. The next column (Correct Intention First Chosen) is one of the metrics that was used in the results of the experiment. It indicates the first time that the correct kit was chosen and was subsequently chosen at every state thereafter. In the table, Kit 1 was chosen as the highest likelihood kit in State 2, but then not chosen in states 3 and 4. As such, State 2 does not count as the state in which the kit was "first correctly chosen." It is not until State 5 when the Kit 1 was correctly chosen (based on the highest likelihood) and then subsequently chosen in every state thereafter.

The last column (Over 20% Confident of Intention) is the second metric that was used in the results of the experiment; it represents the state in which the highest likelihood intention is at least 20 percentage points higher than the second highest intention. This is computed by simply subtracting the second highest likelihood intention from the highest likelihood intention and checking to see if it is greater than 20. As mentioned earlier, the 20% figure is meant to represent the case when the humans and algorithms are very confident which intention is being performed. That 20% value can be changed by the user as desired. In this example, this does not evaluate to true until State 8.

This same process was performed for all intentions (kits) and all plans for both the humans and the algorithm. The next section will compare the results of human experiments with the output of the Intention Recognition Algorithm.

6.2. Experimental Results (Overview)

Based on the experiments described in the previous section, we directly compared the results from the human experiments with the output of the Intention Recognition Algorithm described in Section 4 [158]. The Intention Recognition Algorithm was scored based on how closely it matched the human-generated results. We judged closeness by determining the difference between the algorithm's performance and the average humans' performance using the two performance metrics defined in the previous section (both the state in which the intention was first identified and the state in which the intention were confidently identified).

Table 18 shows the comparison of the output of the Intention Recognition Algorithm to that of humans' performance. Column 1 (Kit) shows the kit that was being developed and Column 2 (Plan) shows the plan that was used to developed the kit. The description of each plan is shown in Appendix B. As mentioned earlier, the two main points of comparison that were used in this experiment were 1) the state (from 1 to 10) when the humans or algorithms first correctly identified the kit that was being created (and consistently identified that correct kit for the remainder of the states), and 2) the state in which the humans were over 20% more confident that the correct kit was being developed compared to the second most probable kit.

The third through fifth columns (blue columns) of the table ("Correct Intention First Chosen") show the average state in which the humans first identified the correct kit that was being developed ("Human"), the state in which the algorithm first identified the correct kit that was being developed ("Algorithm"), and the difference between the two ("Difference"). Similarly the sixth through eighth columns (green columns) of the table ("Over 20% Confident of Intention") shows the average state in which the humans first identified the correct kit that was being developed with over a 20% confidence as compared to the second most probable intention ("Human"), the state in which the algorithm first identified the correct kit that was being developed with over a 20% confidence as compared to the second most probable intention ("Algorithm"), and the difference between the two ("Difference"). A positive value in the difference column means that the algorithm identified the correct kit that many states earlier than the humans. Conversely, a negative value in the difference column means the algorithm identified the kit that many states later than the humans. A zero means the humans and the algorithm identified the kit at the same state.

In analyzing Table 18, we see some very promising results about the performance of the Intention Recognition Algorithm as compared to humans' performance. If we first examine Columns 3-5 (the light blue columns) which compare the state which the algorithms and the humans first chose the correct intention, we see that in over half of the runs (13/25), the Intention Recognition Algorithm determined the proper kit at the exact same state as the humans (as indicated by the zero in the difference column). In eight of the runs, the Intention Recognition Algorithm determined the proper kit earlier than the humans (as indicated by the positive number in the difference column). This ranged from one to four states earlier than the humans. In only four runs did the Intention Recognition Algorithm determine the proper kits later than the humans. In three cases, this was one state later and in the other it was three states later.

When looking at the data that represents when the algorithms and humans were greater than 20% confident (as compared to the next most probable intention), the results are equally promising. In almost half of the runs (12/25), the Intention Recognition Algorithm determined the proper kit (with over 20% confidence compared to the next most probably kit) at the exact same state as the humans (as indicated by the zero in the difference column). In all of the remaining runs, the Intention Recognition Algorithm determined the proper kit earlier than the humans (as indicated by the positive number in the difference column). This ranged from one to five states earlier than the humans.

This data shows that the Intention Recognition Algorithm, in almost every case performed as good, if not better, than humans performing the same activity. In the next section, we will take a deeper look at the individual runs to assess the performance of the Intention Recognition Algorithm at a lower level.

Table 18. Comparison of Algorithm Output to Human Intention Recognition

Kit	Plan	Correct Intention First Chosen			Over 20% Confident of Intention		
		Human	Algorithm	Difference	Human	Algorithm	Difference
1	1	10	10	0	10	10	0
1	2	7	7	0	7	7	0
1	3	10	10	0	10	10	0
1	4	3	2	1	6	5	1
1	5	5	3	2	8	8	0
2	1	4	4	0	7	5	2
2	2	5	1	4	7	3	4
2	3	9	6	3	9	8	1
2	4	9	8	1	9	9	0
2	5	1	1	0	7	3	4
3	1	6	4	2	7	6	1
3	2	4	1	3	5	2	3
3	3	3	4	-1	7	5	2
3	4	1	1	0	6	3	3
3	5	6	3	3	8	6	2
4	1	8	8	0	8	8	0
4	2	5	5	0	6	5	1
4	3	7	8	-1	9	8	1
4	4	7	7	0	7	7	0
4	5	5	8	-3	8	8	0
5	1	4	4	0	4	4	0
5	2	3	3	0	3	3	0
5	3	6	6	0	6	6	0
5	4	8	9	-1	9	9	0
5	5	2	2	0	7	2	5

6.3. Experimental Results (Details)

In this section, we provide the results of all 25 runs (five kits with five runs each). For each run, we provide the following figures:

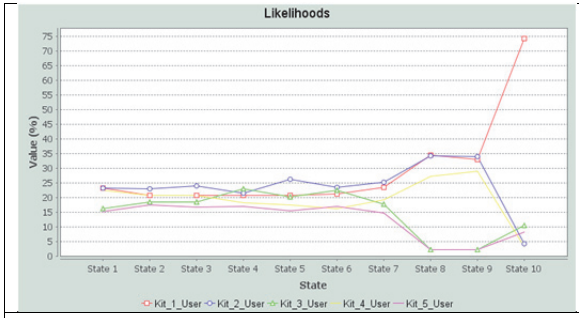
1. **Top Left Figure**- for the specified run, the results of the human experiment. The x-axis represents the sequence of the ten states and the y-axis represents the likelihood that the specified kit is being developed. In each of these figures are five lines, each representing one kit. Each line shows the perceived likelihood by the humans that the specific kit is being developed at each state;
2. **Top Right Figure**- for the specified run, the results of the Intention Recognition Algorithm. The x-axis represents the sequence of the ten states and the y-axis represents the likelihood that the specified kit is being developed. In each of these figures are five lines, each representing one kit. Each line shows the

perceived likelihood by the algorithm that the specific kit is being developed at each state;

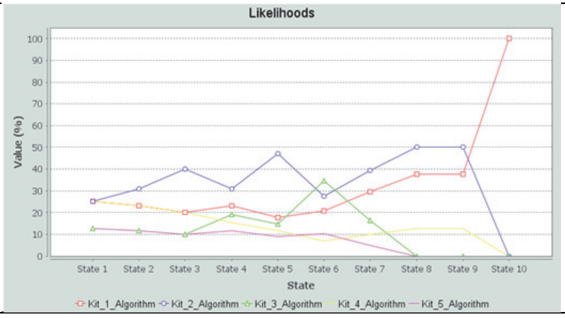
3. **Bottom Figure**- for the specified run, the comparison of the result of the humans and the algorithm for the individual kit that was being created (i.e. the correct kit). In all cases, both the humans and the algorithm ultimately predicted the correct kit that was being created.

The format of the section is consistent for all 25 runs. In each case, the figures described above are presented for the specific run (kit and plan), and a set of observations for each are presented, specifically focusing on the kit that we being created. The observations in this section are all subjective (as opposed to the objective metrics described in Table 18) and all focus on the bottom of the three figures. The top two figures are provided for reference. Three subjective observations that will be discussed in each observation are:

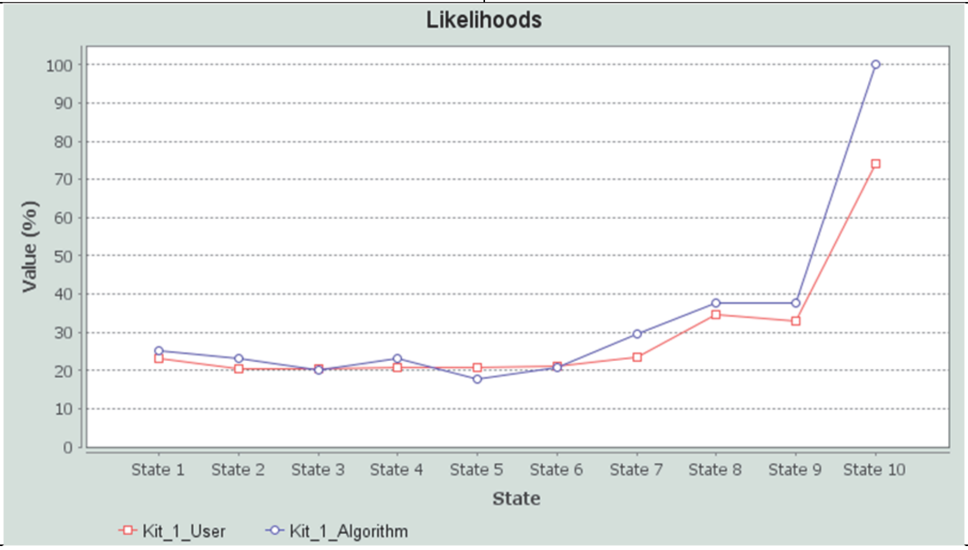
- **Observation 1 -** The comparison of the pattern of lines representing the humans and the algorithm results in the bottom figure. In other words, do these lines have the same general pattern, indicating that the algorithm and the humans are "thinking alike"?
- **Observation 2 -** In general, are the likelihoods at every state for the line representing the results of the algorithm greater or less than that representing the results of humans for the correct kit (the bottom figure)?
- **Observation 3 -** At what state did the likelihood of the correct kit dramatically increase (as represented by the steepest slope between any two states) and how does this compare between the humans and the algorithm? This shows the state in which the confidence level of the humans/algorithm greatly increased.



Humans' Likelihood Data for All Five Kits



Algorithm Likelihood Data for All Five Kits

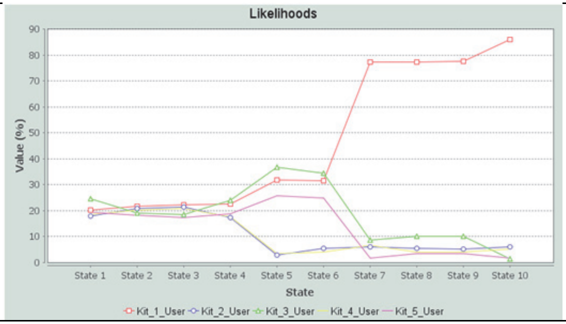


Comparison of Human and Algorithm Data for Kit 1 (Correct Kit)

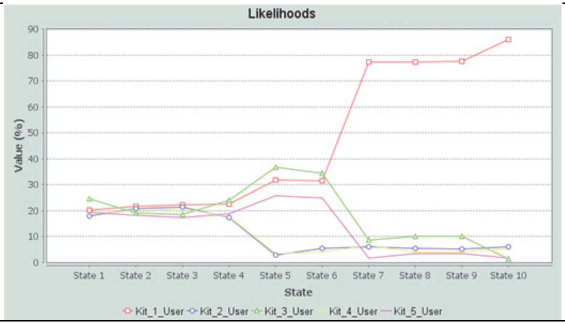
Figure 59: Human/Algorithm Comparison Data for Kit 1 Plan 1

Observations from Figure 59:

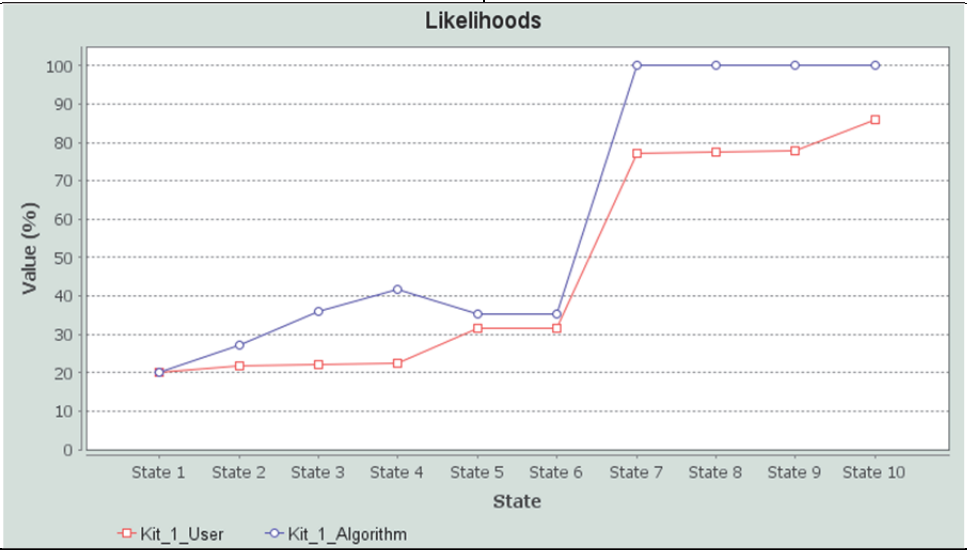
- **Pattern of the Lines-** Both the red line representing the humans' perceived likelihoods and the blue line representing the algorithm's likelihoods at each state have almost the exact same pattern, with the exception of State 10 where the algorithm's likelihood shoots to a higher relative percentage (100%);
- **Likelihoods at Every State-** Except for State 5, the algorithm's likelihood at each state is greater than or equal to that of the user, thus indicating better results by the algorithm;
- **Dramatic Increase in Likelihood-** For both the humans and the algorithm, the likelihood of the correct kit increases dramatically at State 10, where the percentages went up to 75% and 100%, respectively.



Humans’ Likelihood Data for All Five Kits



Algorithm Likelihood Data for All Five Kits

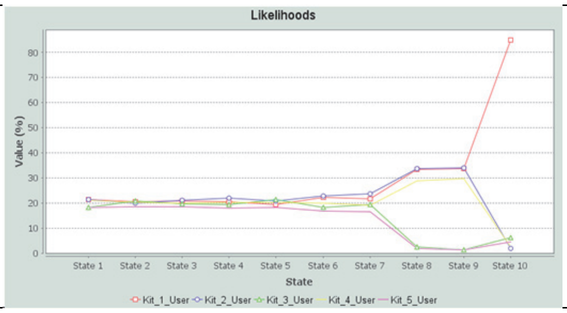


Comparison of Human and Algorithm Data for Kit 1 (Correct Kit)

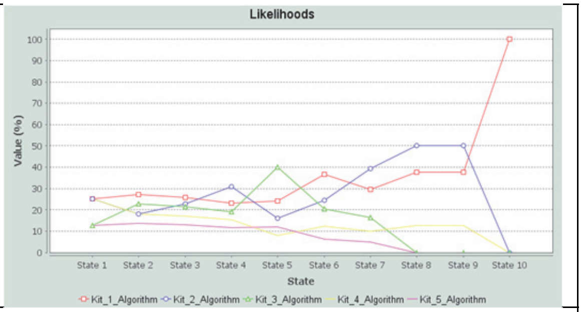
Figure 60: Human/Algorithm Comparison Data for Kit 1 Plan 2

Observations from Figure 60:

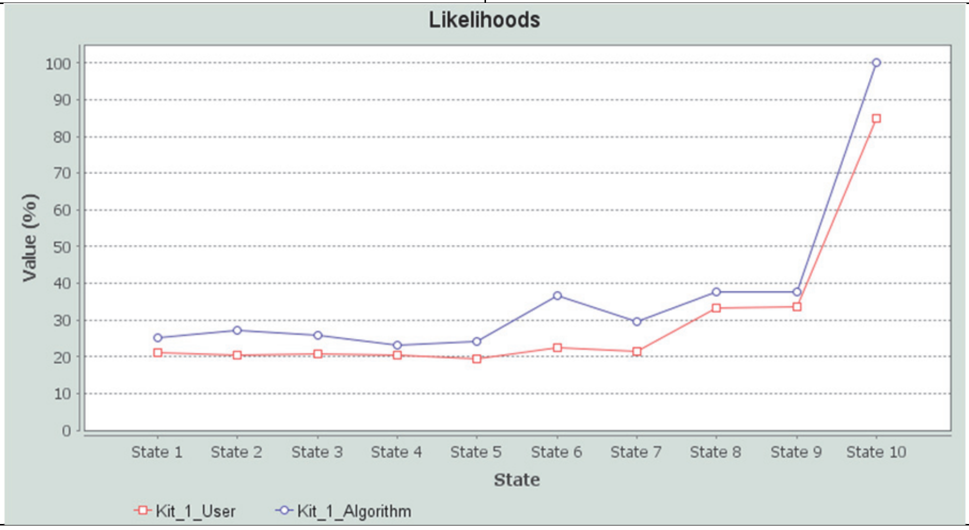
- **Pattern of the Lines**- Both the red line representing the humans’ perceived likelihoods and the blue line representing the algorithm’s likelihoods at each state have almost the exact same pattern. However, the algorithm’s line has a more pronounced “bump” at States 3 and 4 and a more pronounced jump at State 7;
- **Likelihoods at Every State-** The algorithm’s likelihood at each state is greater than that of the user, thus indicating better results by the algorithm;
- **Dramatic Increase in Likelihood-** For both the humans and the algorithm, the likelihood of the correct kit increases dramatically at State 7, where the percentages went up to 78% and 100%, respectively.



Human Likelihood Data for All Five Kits



Algorithm Likelihood Data for All Five Kits

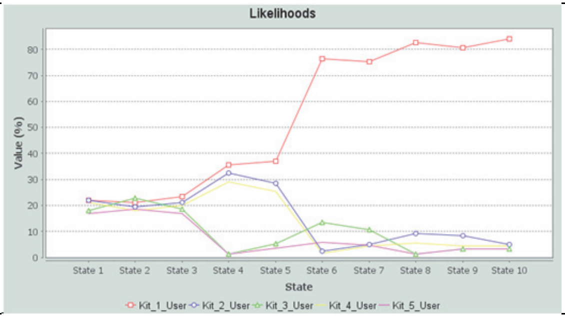


Comparison of Human and Algorithm Data for Kit 1 (Correct Kit)

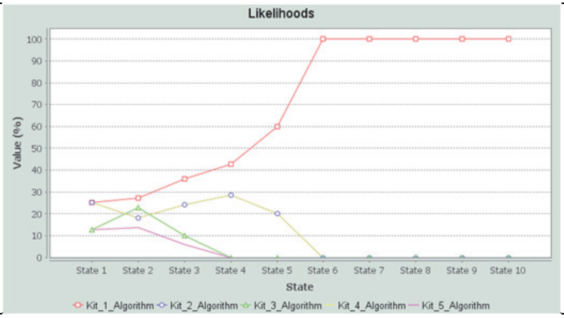
Figure 61: Human/Algorithm Comparison Data for Kit 1 Plan 3

Observations from Figure 61:

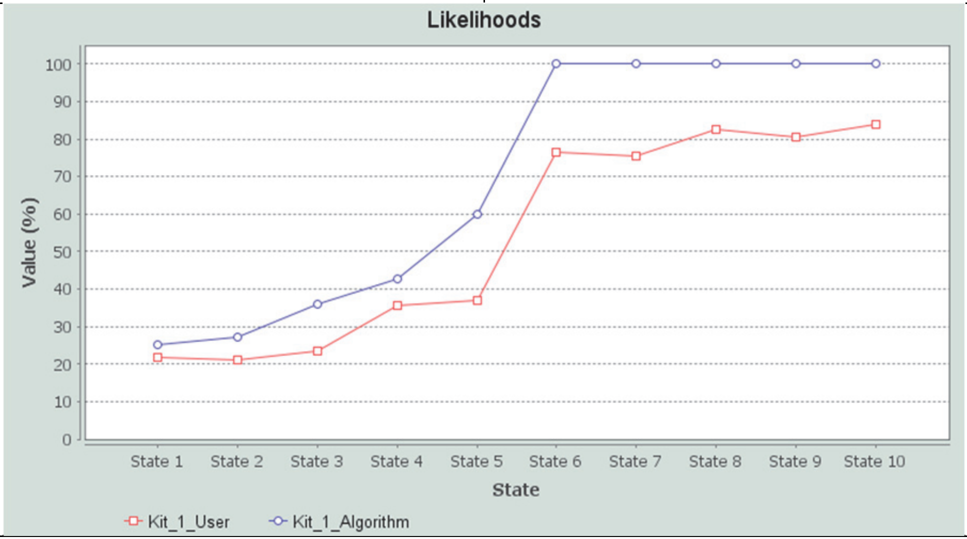
- **Pattern of the Lines**- Both the red line representing the humans' perceived likelihoods and the blue line representing the algorithm's likelihoods at each state have almost the exact same pattern. However, the algorithm line has a more pronounced "bump" at State 6;
- **Likelihoods at Every State-** The algorithm's likelihood at each state is greater than that of the user, thus indicating better results by the algorithm;
- **Dramatic Increase in Likelihood-** For both the humans and the algorithm, the likelihood of the correct kit increases dramatically at State 7, where the percentages went up to 85% and 100%, respectively.



Humans’ Likelihood Data for All Five Kits



Algorithm Likelihood Data for All Five Kits

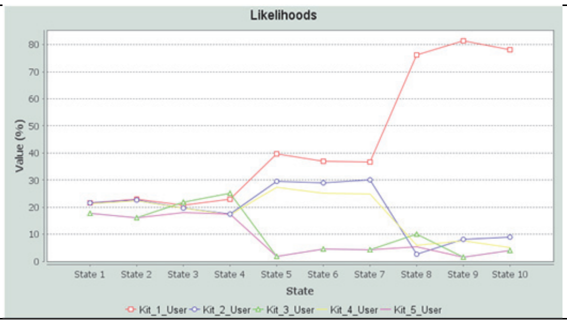


Comparison of Human and Algorithm Data for Kit 1 (Correct Kit)

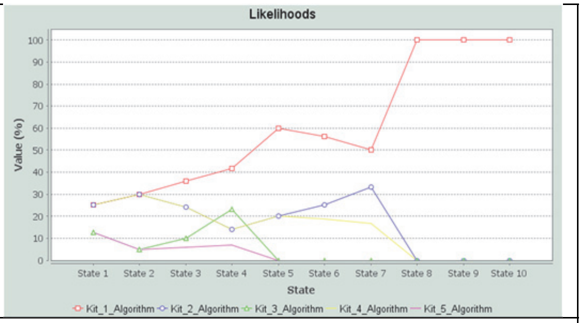
Figure 62: Human/Algorithm Comparison Data for Kit 1 Plan 4

Observations from Figure 62:

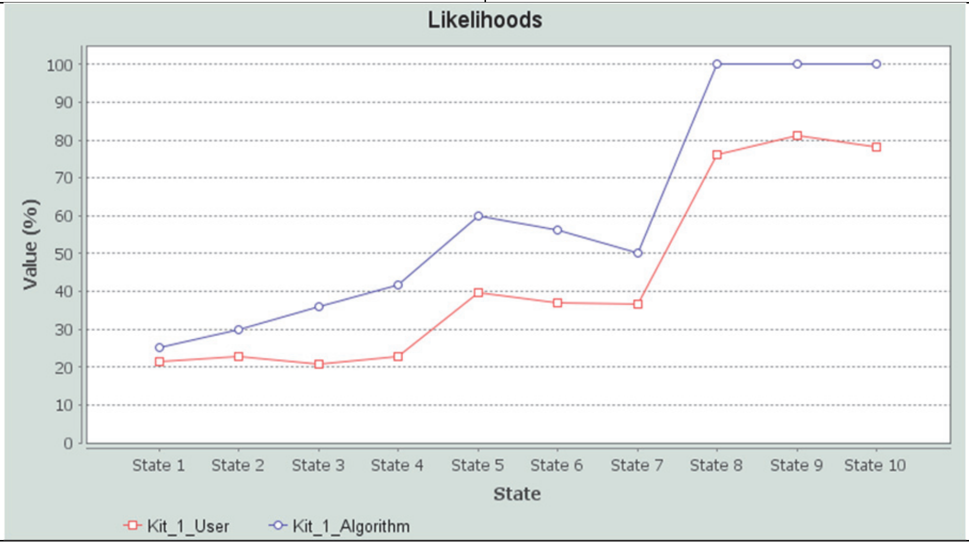
- **Pattern of the Lines**- Both the red line representing the humans’ perceived likelihoods and the blue line representing the algorithm’s likelihoods at each state have almost the exact same pattern. However, the algorithm line has a more pronounced jump at States 5 and 6;
- **Likelihoods at Every State-** The algorithm’s likelihood at each state is greater than that of the humans’, thus indicating better results by the algorithm;
- **Dramatic Increase in Likelihood-** For both the humans and the algorithm, the likelihood of the correct kit increases dramatically at State 6, where the percentages went up to 78% and 100%, respectively.



Humans' Likelihood Data for All Five Kits



Algorithm Likelihood Data for All Five Kits

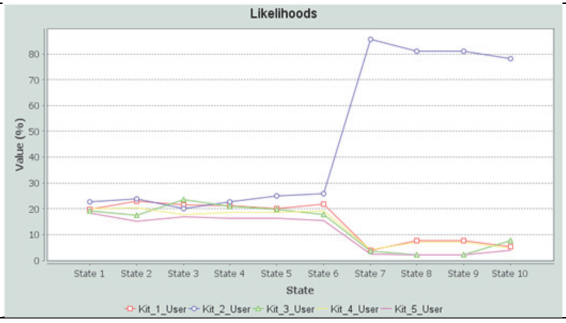


Comparison of Human and Algorithm Data for Kit 1 (Correct Kit)

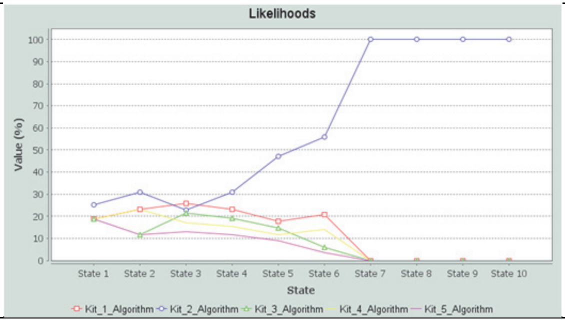
Figure 63: Human/Algorithm Comparison Data for Kit 1 Plan 5

Observations from Figure 63:

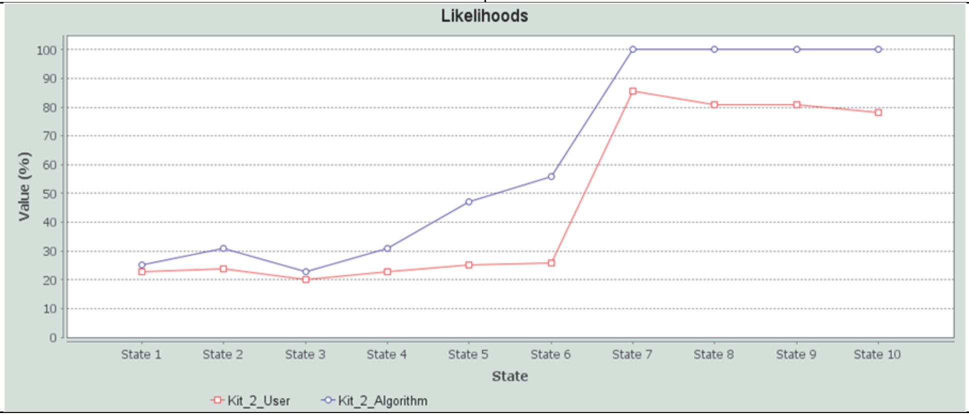
- **Pattern of the Lines**- Both the red line representing the humans' perceived likelihoods and the blue line representing the algorithm's likelihoods at each state have almost the exact same pattern. However, the algorithm's line has a more pronounced "bump" at States 3, 4, and 8;
- **Likelihoods at Every State-** The algorithm's likelihood at each state is greater than that of the humans', thus indicating better results by the algorithm-
- **Dramatic Increase in Likelihood-** For both the humans and the algorithm, the likelihood of the correct kit increases dramatically at State 8, where the percentages went up to 78% and 100%, respectively.



Humans' Likelihood Data for All Five Kits



Algorithm Likelihood Data for All Five Kits

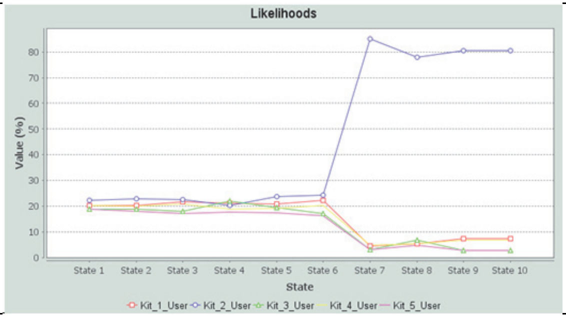


Comparison of Human and Algorithm Data for Kit 2 (Correct Kit)

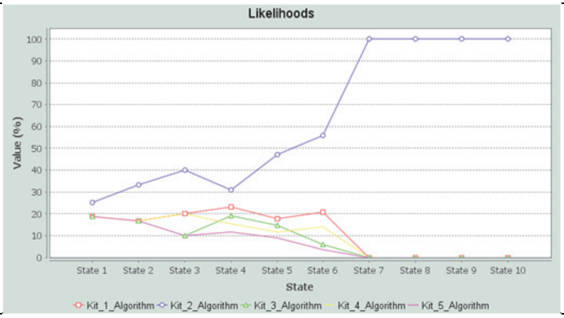
Figure 64: Human/Algorithm Comparison Data for Kit 2 Plan 1

Observations from Figure 64:

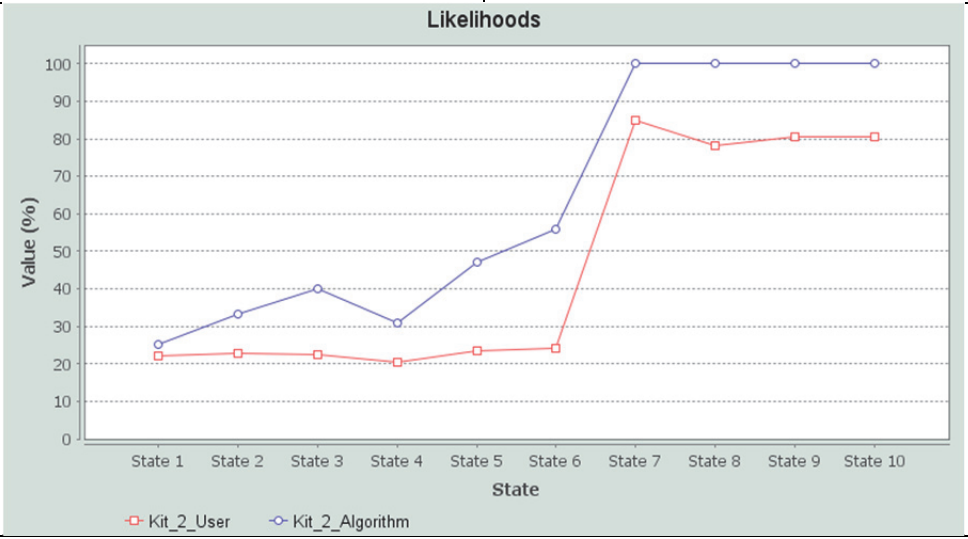
- **Pattern of the Lines**- Both the red line representing the humans' perceived likelihoods and the blue line representing the algorithm's likelihoods at each state have almost the exact same pattern. However, the algorithm's line has a more pronounced jump at State 5;
- **Likelihoods at Every State-** The algorithm's likelihood at each state is greater than that of the humans', thus indicating better results by the algorithm;
- **Dramatic Increase in Likelihood-** For both the humans and the algorithm, the likelihood of the correct kit increases dramatically at State 7, where the percentages went up to 86% and 100%, respectively.



Humans’ Likelihood Data for All Five Kits



Algorithm Likelihood Data for All Five Kits

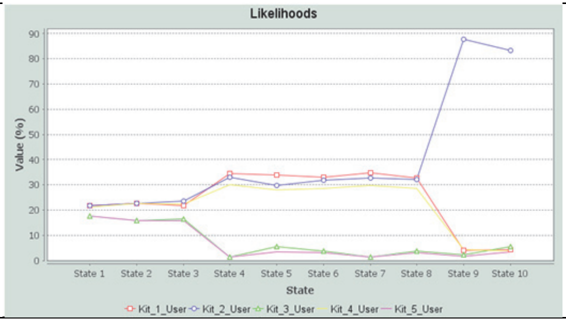


Comparison of Human and Algorithm Data for Kit 2 (Correct Kit)

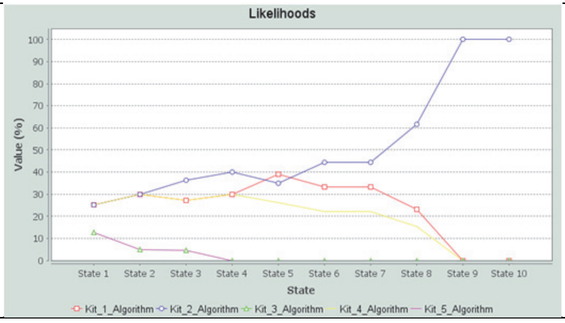
Figure 65: Human/Algorithm Comparison Data for Kit 2 Plan 2

Observations from Figure 65:

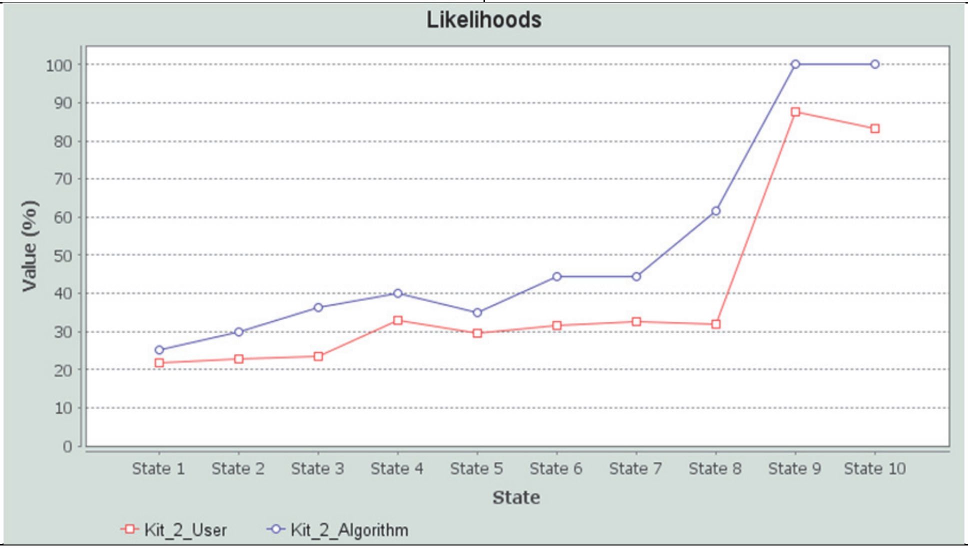
- **Pattern of the Lines**- Both the red line representing the humans’ perceived likelihoods and the blue line representing the algorithm’s likelihoods at each state have almost the exact same pattern. However, the algorithm’s line has a more pronounced jump at States 3, 5, and 6;
- **Likelihoods at Every State-** The algorithm’s likelihood at each state is greater than that of the humans’, thus indicating better results by the algorithm;
- **Dramatic Increase in Likelihood-** For both the humans and the algorithm, the likelihood of the correct kit increases dramatically at State 7, where the percentages went up to 85% and 100%, respectively.



Humans' Likelihood Data for All Five Kits



Algorithm Likelihood Data for All Five Kits

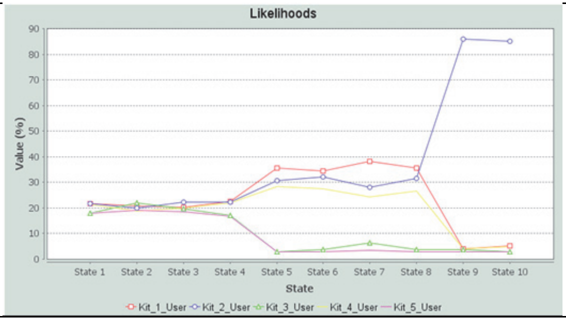


Comparison of Human and Algorithm Data for Kit 2 (Correct Kit)

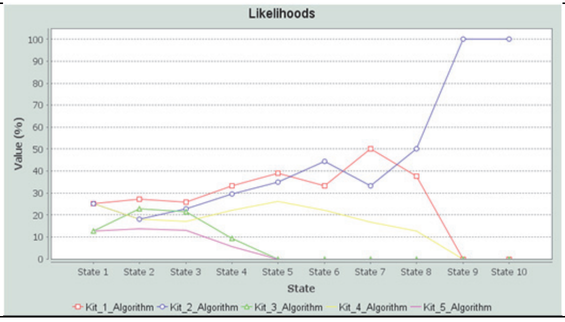
Figure 66: Human/Algorithm Comparison Data for Kit 2 Plan 3

Observations from Figure 66:

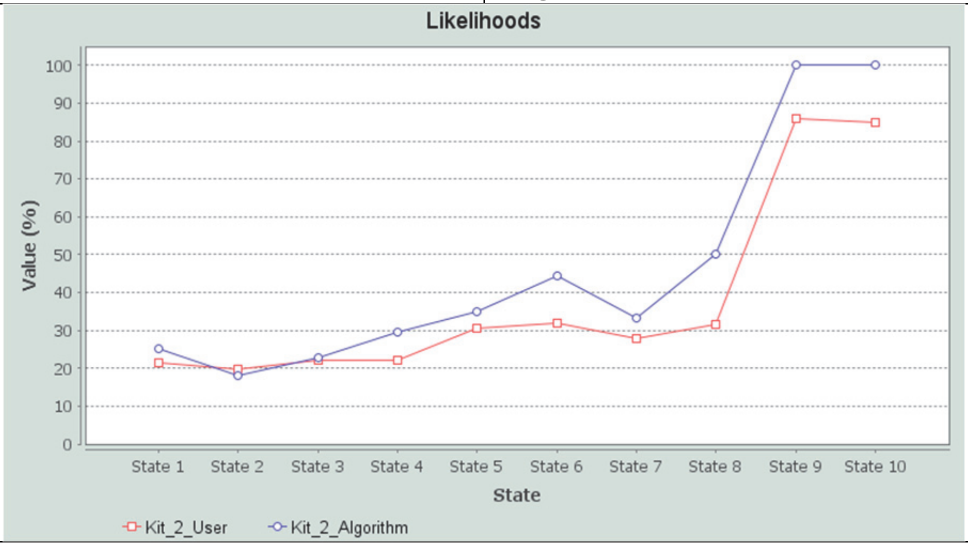
- **Pattern of the Lines**- Both the red line representing the humans' perceived likelihoods and the blue line representing the algorithm's likelihoods at each state have almost the exact same pattern. However, the algorithm's line has a more pronounced jump at State 8;
- **Likelihoods at Every State-** The algorithm's likelihood at each state is greater than that of the humans', thus indicating better results by the algorithm;
- **Dramatic Increase in Likelihood-** For both the humans and the algorithm, the likelihood of the correct kit increases dramatically at State 9, where the percentages went up to 89% and 100%, respectively.



Humans' Likelihood Data for All Five Kits



Algorithm Likelihood Data for All Five Kits

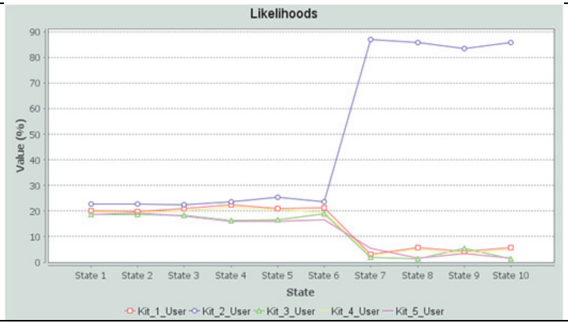


Comparison of Human and Algorithm Data for Kit 2 (Correct Kit)

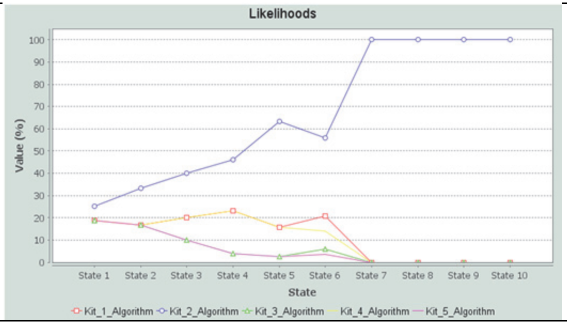
Figure 67: Human/Algorithm Comparison Data for Kit 2 Plan 4

Observations from Figure 67:

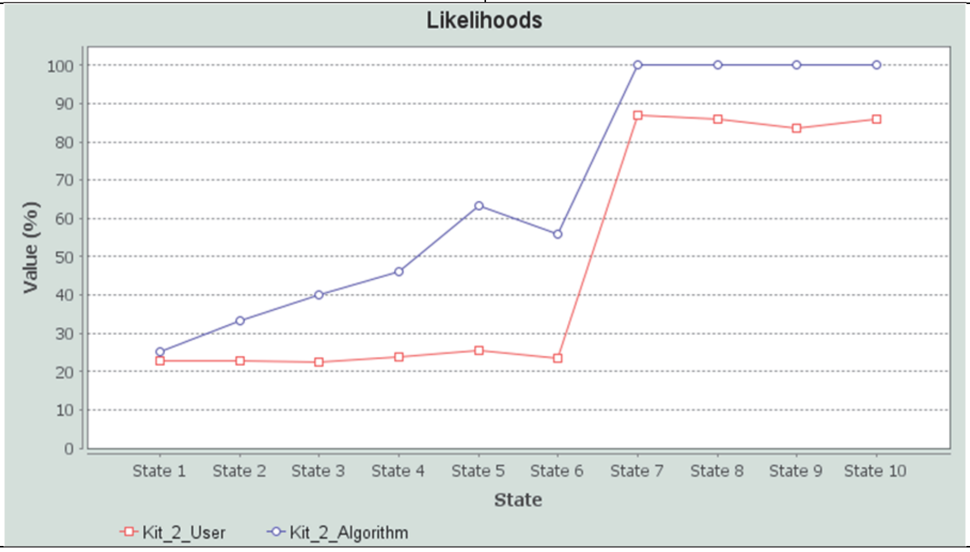
- **Pattern of the Lines**- Both the red line representing the humans' perceived likelihoods and the blue line representing the algorithm's likelihoods at each state have almost the exact same pattern. However, the algorithm's line has a more pronounced jump at States 6, 8, and 9;
- **Likelihoods at Every State-** The algorithm's likelihood at each state is greater than that of the humans', except for State 2, thus indicating better results by the algorithm;
- **Dramatic Increase in Likelihood-** For both the humans and the algorithm, the likelihood of the correct kit increase dramatically at State 9, where the percentages went up to 87% and 100%, respectively.



Humans’ Likelihood Data for All Five Kits



Algorithm Likelihood Data for All Five Kits

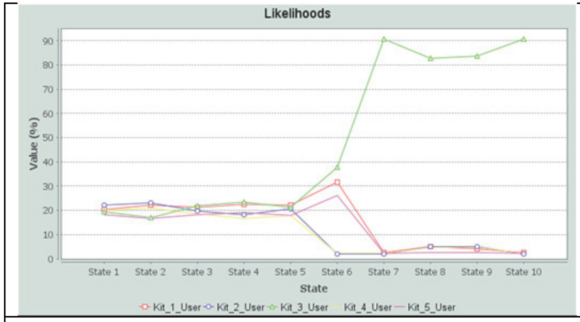


Comparison of Human and Algorithm Data for Kit 2 (Correct Kit)

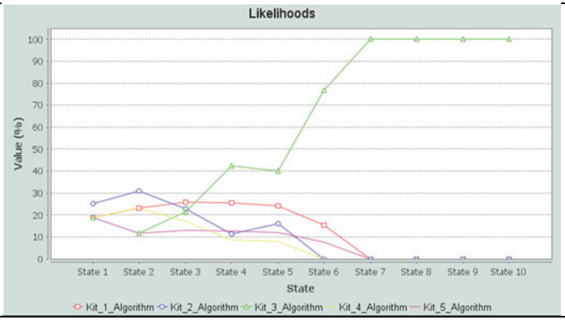
Figure 68: Human/Algorithm Comparison Data for Kit 2 Plan 5

Observations from Figure 68:

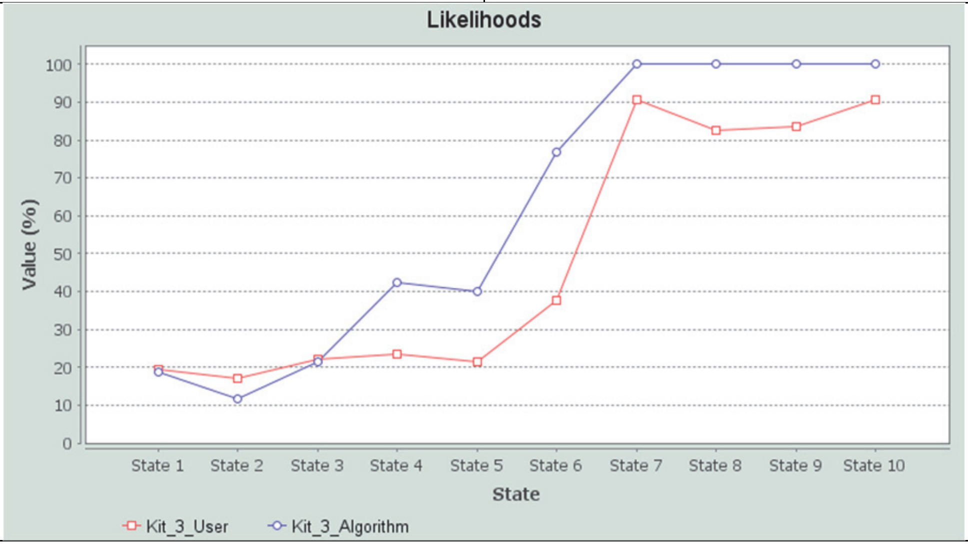
- **Pattern of the Lines**- Both the red line representing the humans’ perceived likelihoods and the blue line representing the algorithm’s likelihoods at each state have almost the exact same pattern with the exception of States 2-6. In these states, the likelihood from the algorithm consistently increases while the likelihood from the humans remains relatively constant;
- **Likelihoods at Every State-** The algorithm’s likelihood at each state is greater than that of the humans’, thus indicating better results by the algorithm;
- **Dramatic Increase in Likelihood-** For both the humans and the algorithm, the likelihood of the correct kit increases dramatically at State 7, where the percentages went up to 88% and 100%, respectively.



Humans' Likelihood Data for All Five Kits



Algorithm Likelihood Data for All Five Kits

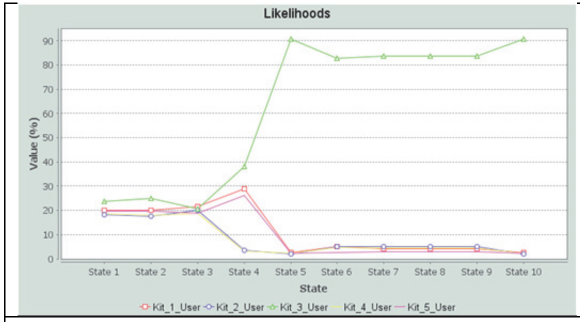


Comparison of Human and Algorithm Data for Kit 3 (Correct Kit)

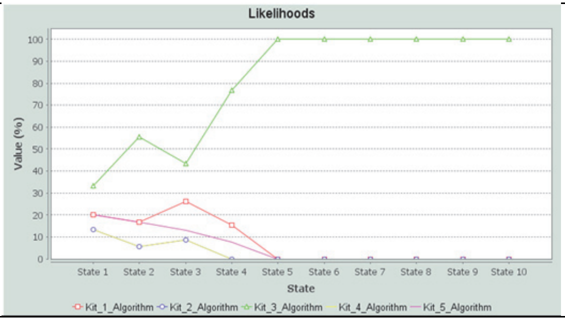
Figure 69: Human/Algorithm Comparison Data for Kit 3 Plan 1

Observations from Figure 69:

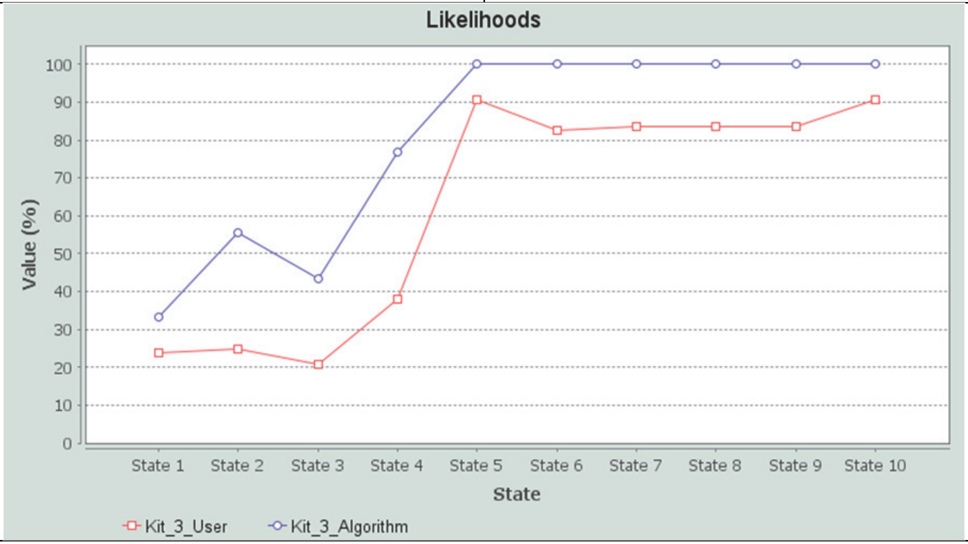
- **Pattern of the Lines**- Both the red line representing the humans' perceived likelihoods and the blue line representing the algorithm's likelihoods at each state have almost the exact same pattern. However, the algorithm's line has a jump at State 4 which does not exist to that extreme in the humans' results;
- **Likelihoods at Every State-** The algorithm's likelihood at each state is greater than that of the humans' with the exception of States 2 and 3;
- **Dramatic Increase in Likelihood-** For both the humans and the algorithm, the likelihood of the correct kit increases dramatically at State 7, where the percentages went up to 90% and 100%, respectively.



Humans' Likelihood Data for All Five Kits



Algorithm Likelihood Data for All Five Kits

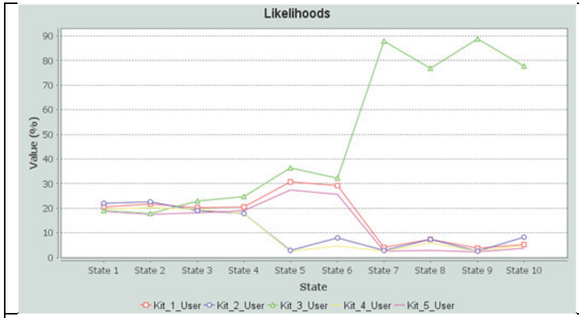


Comparison of Human and Algorithm Data for Kit 3 (Correct Kit)

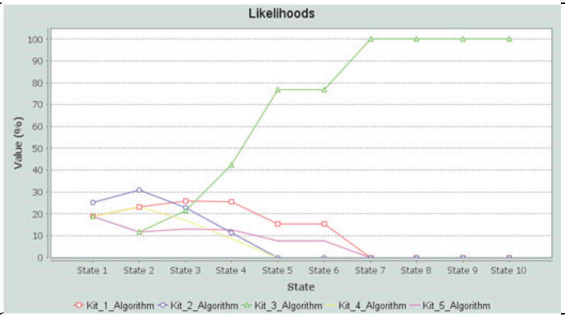
Figure 70: Human/Algorithm Comparison Data for Kit 3 Plan 2

Observations from Figure 70:

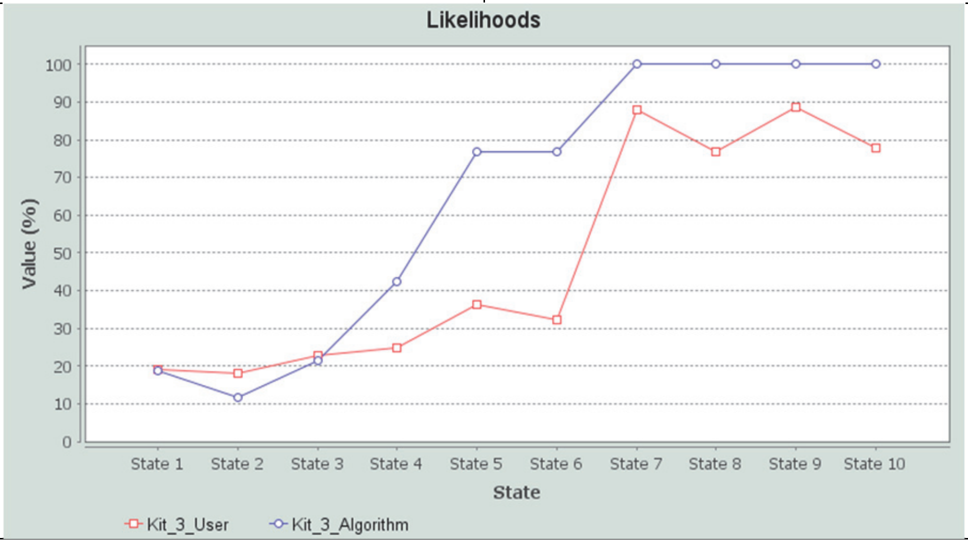
- **Pattern of the Lines**- Both the red line representing the humans' perceived likelihoods and the blue line representing the algorithm's likelihoods at each state have almost the exact same pattern. However, the algorithm's line has a more pronounced jump at State 2;
- **Likelihoods at Every State-** The algorithm's likelihood at each state is greater than that of the humans', thus indicating better results by the algorithm;
- **Dramatic Increase in Likelihood-** For both the humans and the algorithm, the likelihood of the correct kit increases dramatically at State 5, where the percentages went up to 90% and 100%, respectively.



Humans’ Likelihood Data for All Five Kits



Algorithm Likelihood Data for All Five Kits

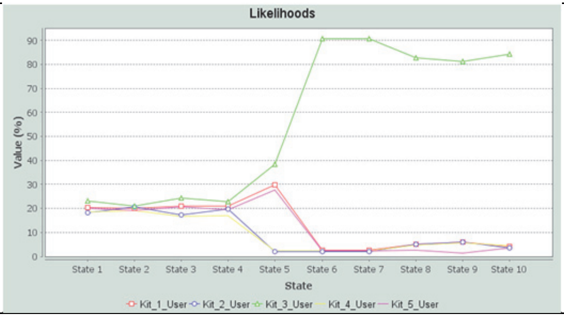


Comparison of Human and Algorithm Data for Kit 3 (Correct Kit)

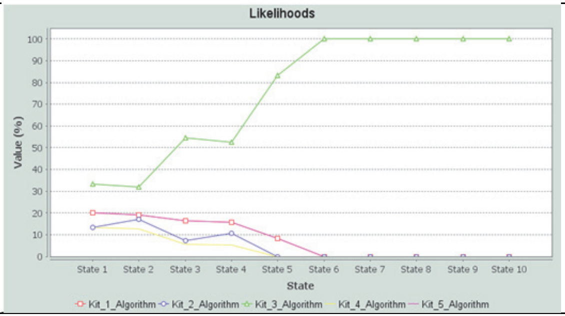
Figure 71: Human/Algorithm Comparison Data for Kit 3 Plan 3

Observations from Figure 71:

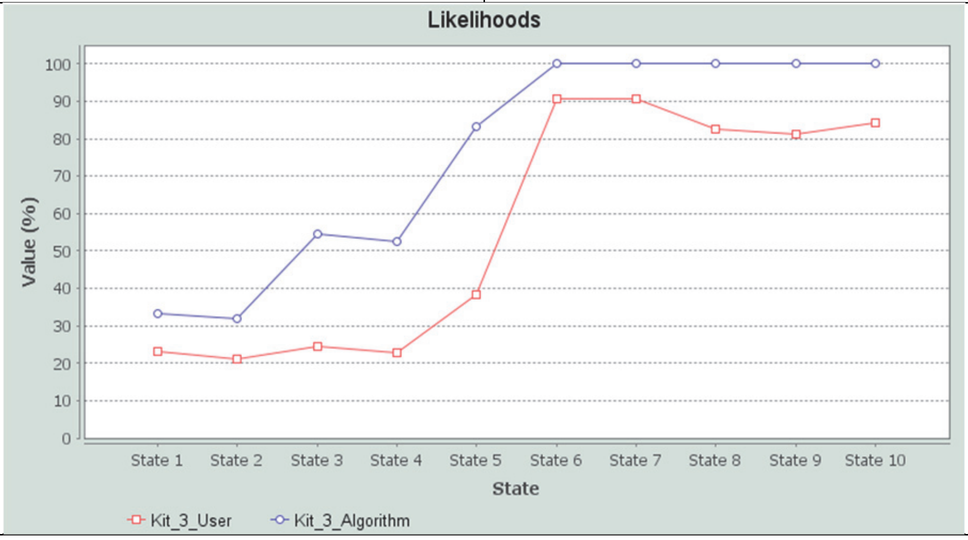
- **Pattern of the Lines**- The two lines have slightly different patterns. They start out about the same in States 1-3, but then the algorithm’s line has a quick jump at States 4 and 5. After that they remain in similar Pattern, with a more consistent likelihood (100%) through States 7-10;
- **Likelihoods at Every State-** The algorithm’s likelihood at each state after State 3 is greater than that of the humans’, while the humans’ line shows greater likelihoods from States 1-3;
- **Dramatic Increase in Likelihood-** The algorithm’s line shows two dramatic increases – one from States 4-5 and a second at State 7. The humans’ line only has a dramatic increase at State 7.



Humans' Likelihood Data for All Five Kits



Algorithm Likelihood Data for All Five Kits

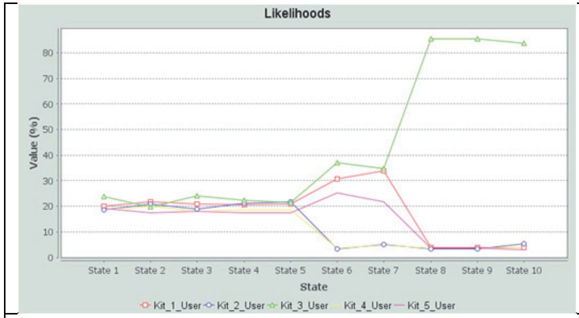


Comparison of Human and Algorithm Data for Kit 3 (Correct Kit)

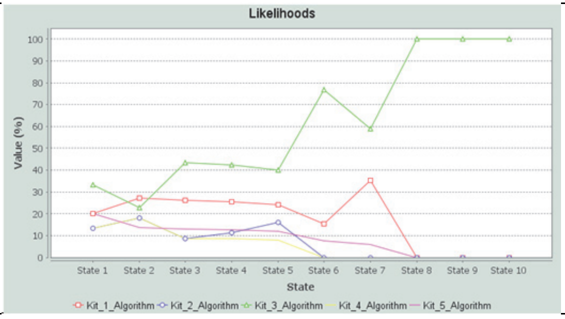
Figure 72: Human/Algorithm Comparison Data for Kit 3 Plan 4

Observations from Figure 72:

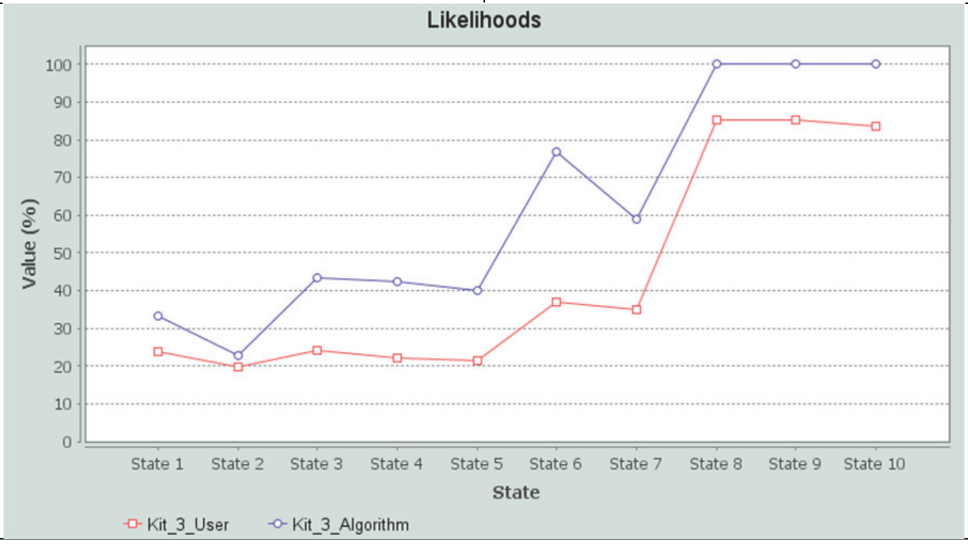
- **Pattern of the Lines**- Both the red line representing the humans' perceived likelihoods and the blue line representing the algorithm's likelihoods at each state have almost the exact same pattern. However, the algorithm's line has a more pronounced jump at State 3;
- **Likelihoods at every State-** The algorithm's likelihood at each state is greater than that of the humans', thus indicating better results by the algorithm-
- **Dramatic Increase in Likelihood-** For both the humans and the algorithm, the likelihood of the correct kit increases dramatically at State 6, where the percentages went up to 90% and 100%, respectively. The humans' line then trails off a bit at the end, but still keeps a very high likelihood.



Humans' Likelihood Data for All Five Kits



Algorithm Likelihood Data for All Five Kits

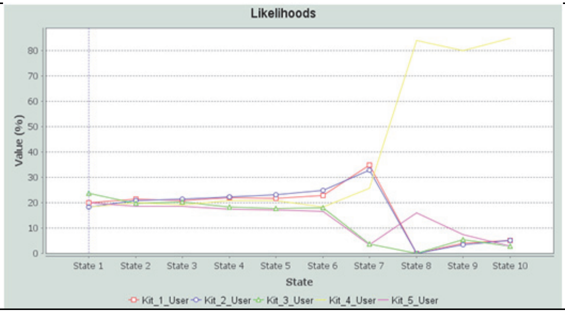
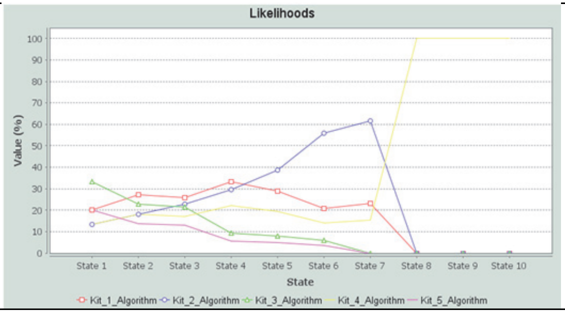


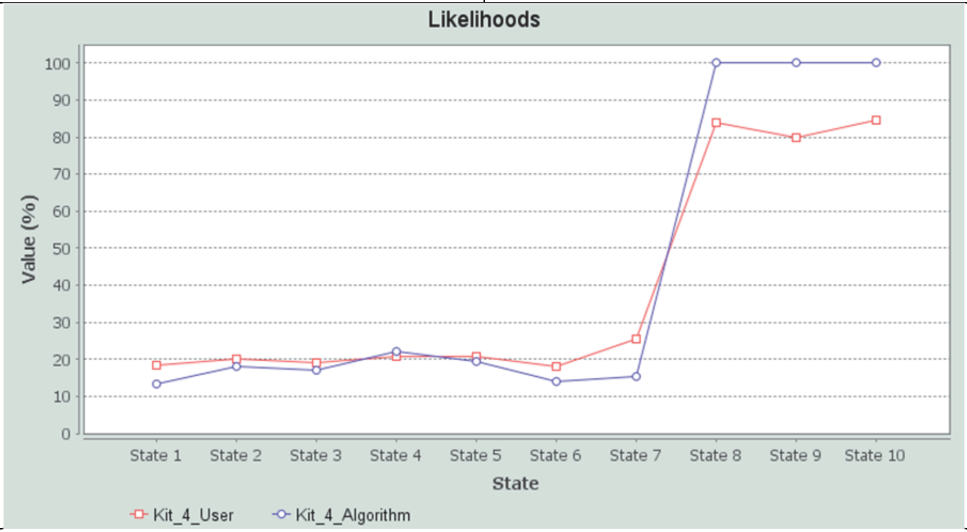
Comparison of Human and Algorithm Data for Kit 3 (Correct Kit)

Figure 73: Human/Algorithm Comparison Data for Kit 3 Plan 5

Observations from Figure 73:

- **Pattern of the Line**- Both the red line representing the humans' perceived likelihoods and the blue line representing the algorithm's likelihoods at each state have almost the exact same pattern. However, the algorithm's line has a more pronounced dip at State 2 and then a significant jump at State 6;
- **Likelihoods at Every State-** The algorithm's likelihood at each state is greater than that of the humans', thus indicating better results by the algorithm;
- **Dramatic Increase in Likelihood-** The algorithm line has two jumps, one at State 6 and one at State 8. Conversely, the humans' line one has one significant jump at State 8.

	
Humans’ Likelihood Data for All Five Kits	Algorithm Likelihood Data for All Five Kits

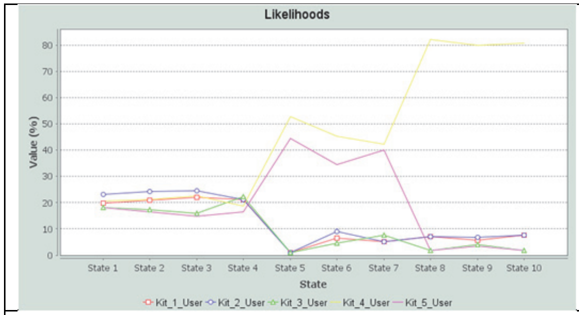


Comparison of Human and Algorithm Data for Kit 4 (Correct Kit)

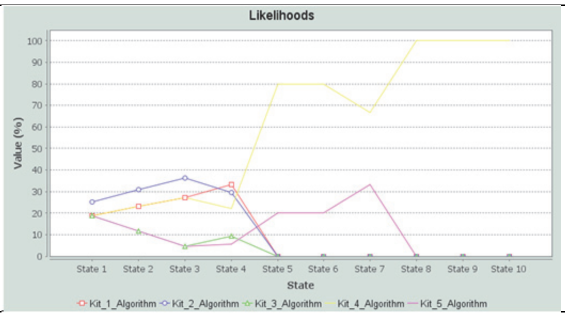
Figure 74: Human/Algorithm Comparison Data for Kit 4 Plan 1

Observations from Figure 74:

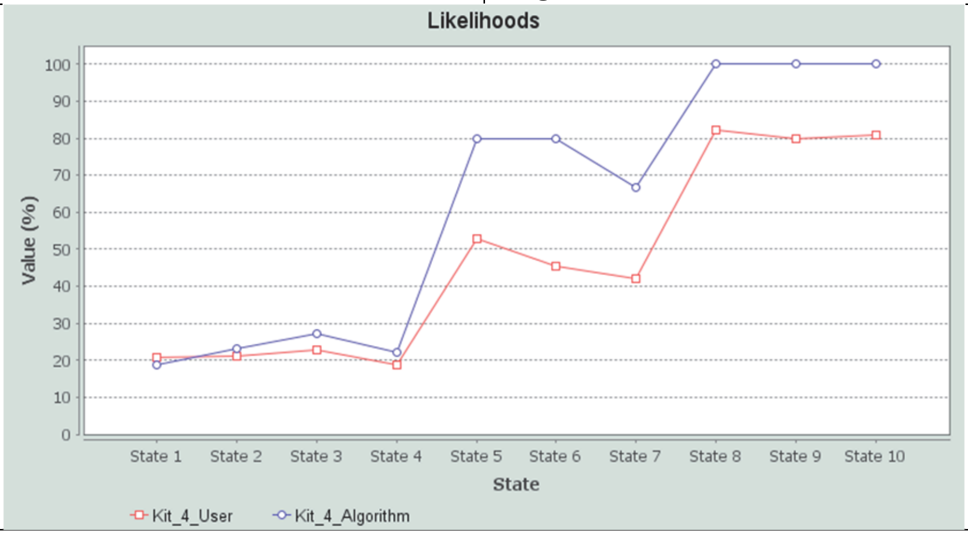
- **Pattern of the Lines**- Both the red line representing the humans’ perceived likelihoods and the blue line representing the algorithm’s likelihoods at each state have almost the exact same pattern. However, the algorithm’s line has a more pronounced jump at State 8;
- **Likelihoods at Every State-** The humans’ likelihood is greater than the algorithm’s likelihood at almost every state between 1 and 7, with the exception of State 4. However, in every case, the likelihoods between the two lines are very close at each of these states. After State 7, the algorithm’s likelihoods are consistently higher than the humans’;
- **Dramatic Increase in Likelihood-** For both the humans and the algorithm, the likelihood of the correct kit increases dramatically at State 8, where the percentages went up to 83% and 100%, respectively.



Humans' Likelihood Data for All Five Kits



Algorithm Likelihood Data for All Five Kits

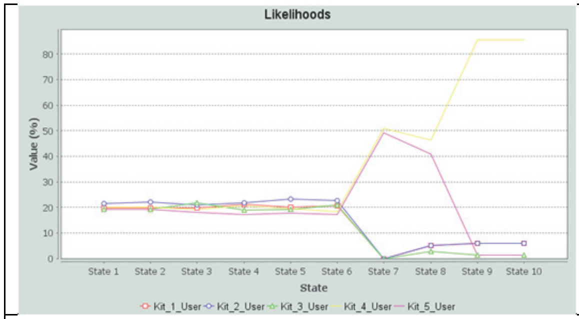


Comparison of Human and Algorithm Data for Kit 4 (Correct Kit)

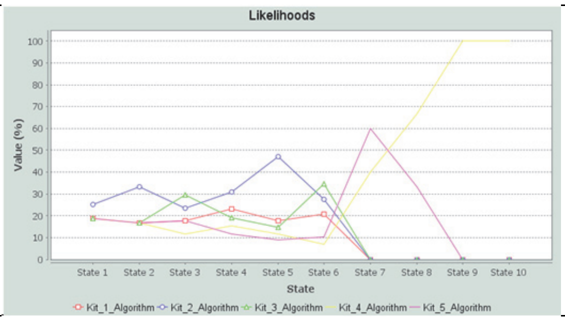
Figure 75: Human/Algorithm Comparison Data for Kit 4 Plan 2

Observations from Figure 75:

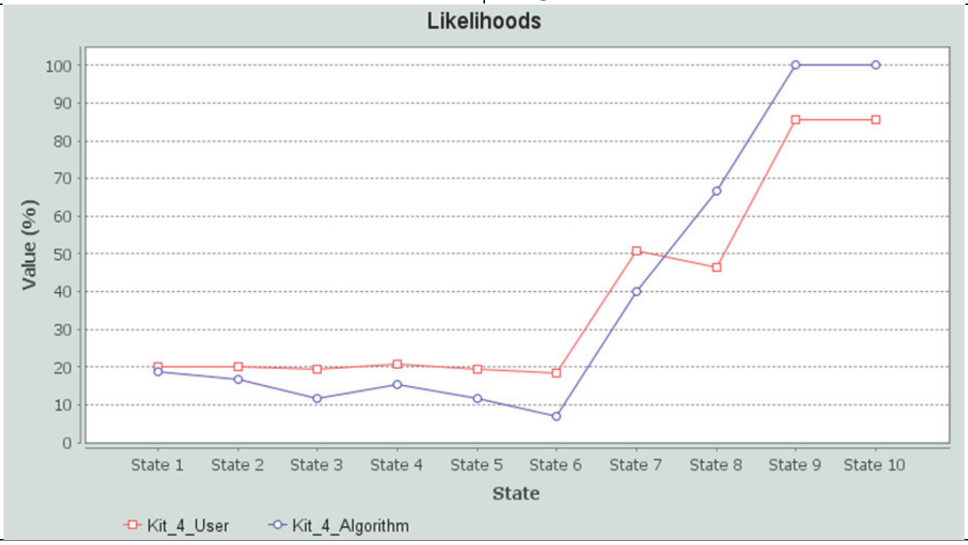
- **Pattern of the Lines**- Both the red line representing the humans' perceived likelihoods and the blue line representing the algorithm's likelihoods at each state have almost the exact same pattern. However, the algorithm's line has a more pronounced jump at States 5 and 8;
- **Likelihoods at Every State-** The algorithm's likelihood at each state (except for State 1) is greater than that of the humans', thus indicating better results by the algorithm;
- **Dramatic Increase in Likelihood-** There are two dramatic increases in the algorithm's likelihood at states 5 and 8, while in the humans' likelihoods, there is only a dramatic increase at State 8.



Humans' Likelihood Data for All Five Kits



Algorithm Likelihood Data for All Five Kits

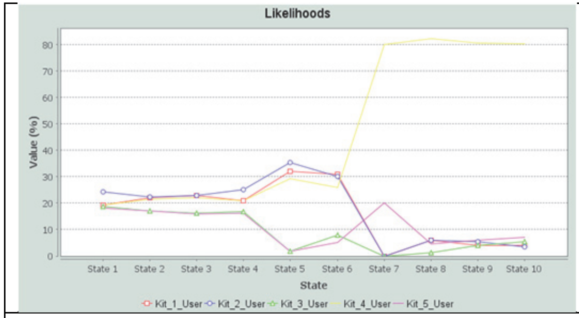


Comparison of Human and Algorithm Data for Kit 4 (Correct Kit)

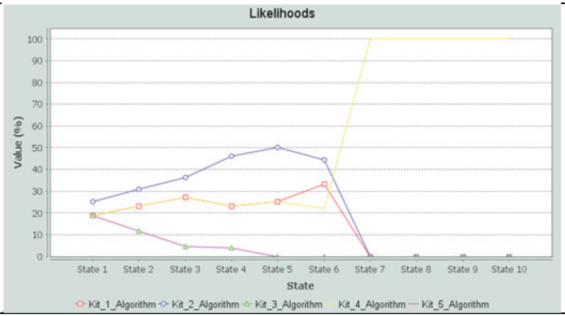
Figure 76: Human/Algorithm Comparison Data for Kit 4 Plan 3

Observations from Figure 76:

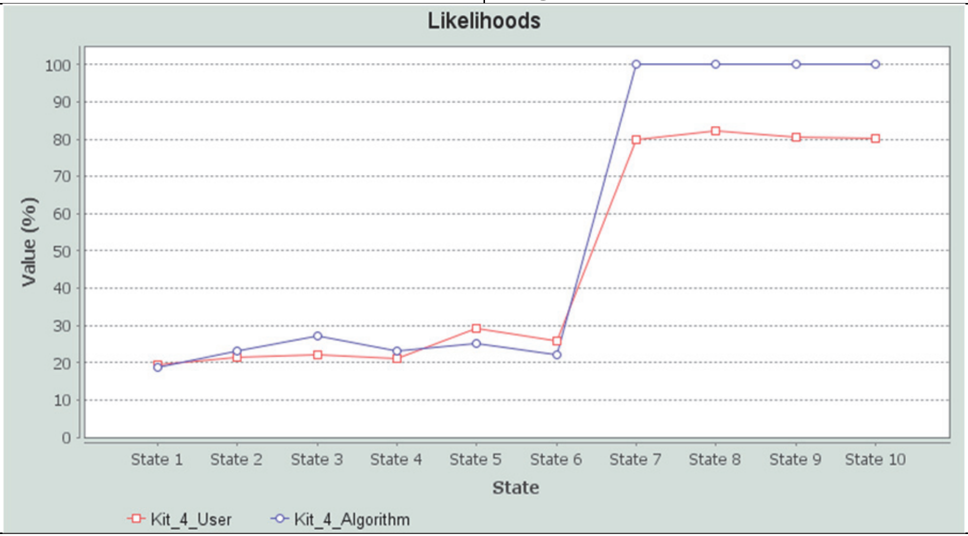
- **Pattern of the Lines**- Both the red line representing the humans' perceived likelihoods and the blue line representing the algorithm's likelihoods at each state have almost the exact same pattern. However, the humans' line has a dip at State 8 which does not exist in the algorithm's line;
- **Likelihoods at Every State-** The humans' likelihood at each state between States 1 and 7 is slightly greater than the algorithm's likelihood. Starting at State 8, the algorithm's likelihood is slightly better;
- **Dramatic Increase in Likelihood-** The algorithm's likelihoods start to increase at State 7 and then continues to increase through State 9. The humans' likelihoods also start to increase at State 7, but then take a dip at State 8 before increasing again at State 9.



Humans' Likelihood Data for All Five Kits



Algorithm Likelihood Data for All Five Kits

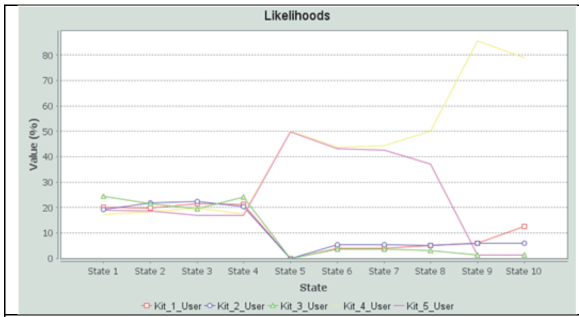


Comparison of Human and Algorithm Data for Kit 4 (Correct Kit)

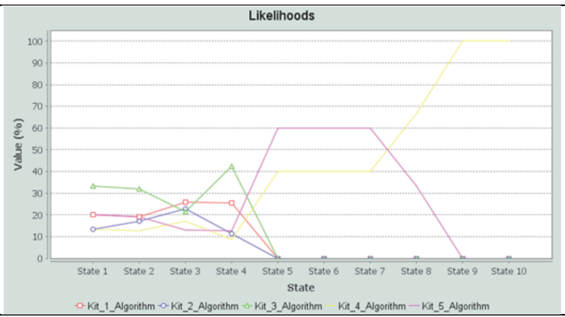
Figure 77: Human/Algorithm Comparison Data for Kit 4 Plan 4

Observations from Figure 77:

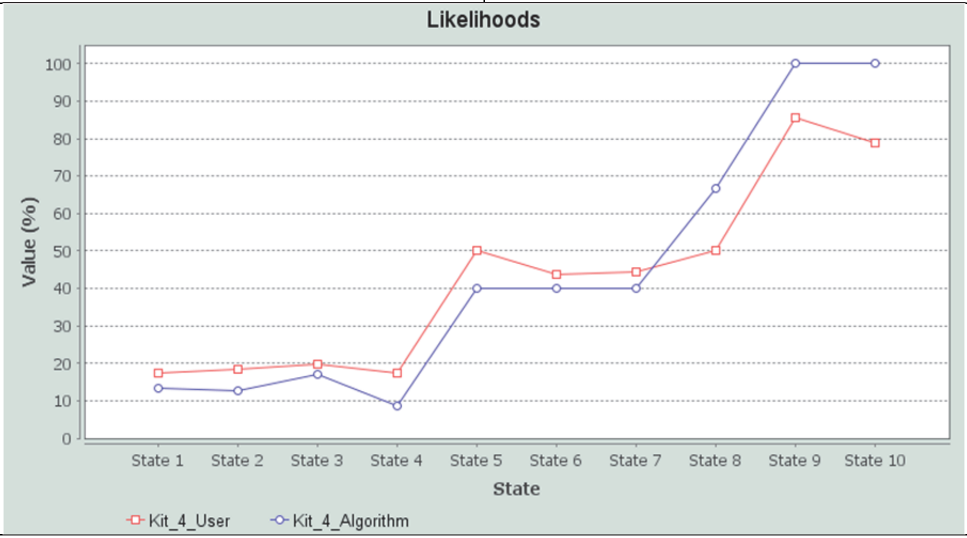
- **Pattern of the Lines**- Both the red line representing the humans' perceived likelihoods and the blue line representing the algorithm's likelihoods at each state have almost the exact same pattern. However, the algorithm's line has a more pronounced jump at State 7;
- **Likelihoods at Every State-** The algorithm's likelihood at each state (except for States 1, 5, and 6) is greater than that of the humans';
- **Dramatic Increase in Likelihood:** For both the humans and the algorithm, the likelihood of the correct kit increases dramatically at State 7, where the percentages went up to 80% and 100%, respectively.



Humans' Likelihood Data for All Five Kits



Algorithm Likelihood Data for All Five Kits

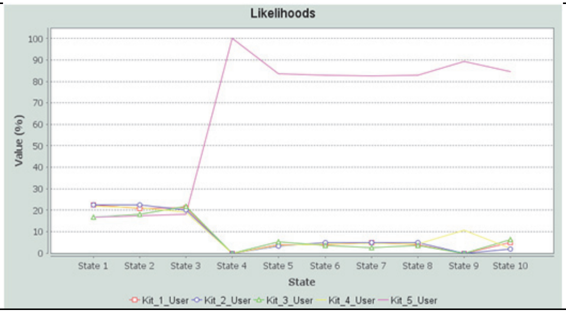
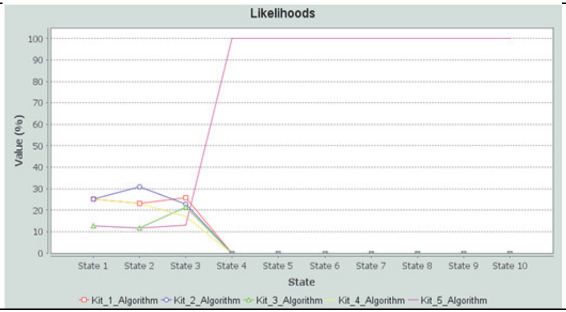


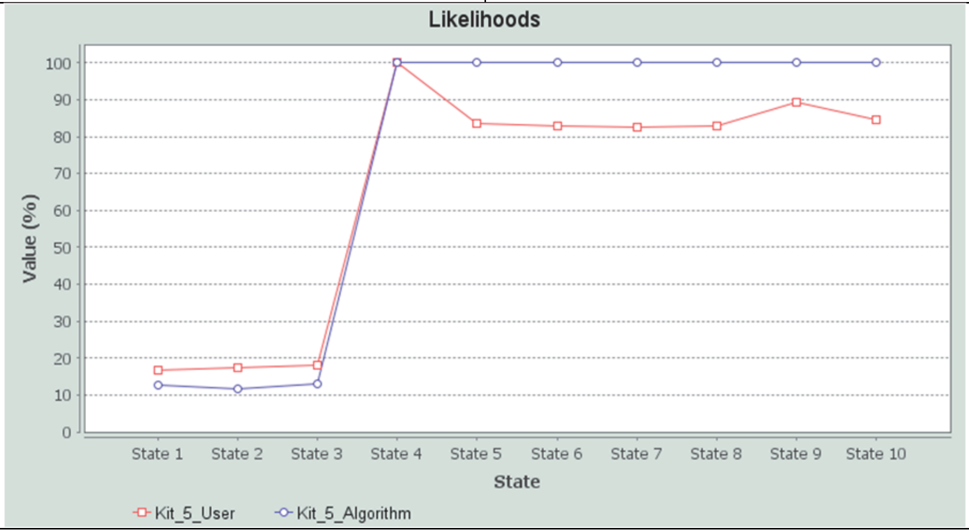
Comparison of Human and Algorithm Data for Kit 4 (Correct Kit)

Figure 78: Human/Algorithm Comparison Data for Kit 4 Plan 5

Observations from Figure 78:

- **Pattern of the Lines**- Both the red line representing the humans' perceived likelihoods and the blue line representing the algorithm's likelihoods at each state have almost the exact same pattern. However, the algorithms line has a more pronounced jump at State 8;
- **Likelihoods at Every State-** The humans' likelihood at each state until State 7 is greater than that of the algorithm's, often by a small margin. Starting at State 8, the algorithm's likelihood is greater than the humans';
- **Dramatic Increase in Likelihood-** For both the humans and the algorithm, there is a dramatic increase in likelihood at both States 5 and 9. However, at State 8, the algorithm's has a dramatic increase in likelihood that does not exist in the humans' line.

	
Humans’ Likelihood Data for All Five Kits	Algorithm Likelihood Data for All Five Kits

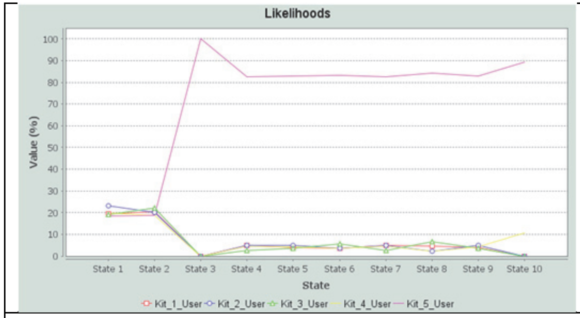


Comparison of Human and Algorithm Data for Kit 5 (Correct Kit)

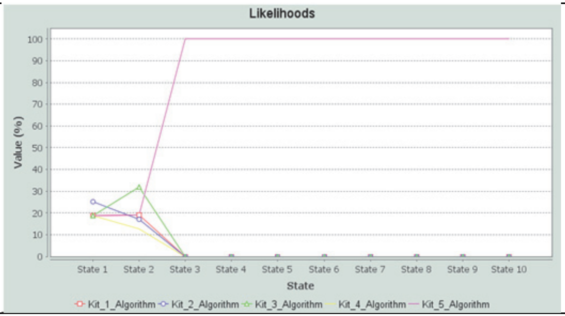
Figure 79: Human/Algorithm Comparison Data for Kit 5 Plan 1

Observations from Figure 79:

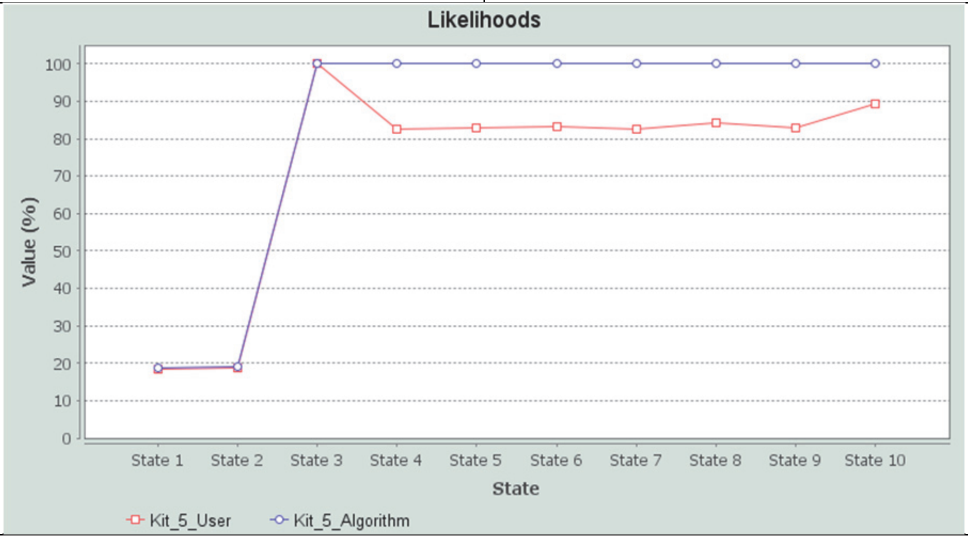
- **Pattern of the Lines**- Both the red line representing the humans’ likelihoods and the blue line representing the algorithm’s likelihoods at each state have almost identical patterns. However, the humans’ line has a pronounced dip at State 5 which does not exist in the algorithm’s line;
- **Likelihoods at Every State-** The humans’ likelihood at States 1 to 3 a greater than the algorithm’s by a small margin. They are about the same at State 4 and then the algorithm’s likelihoods are great than the humans’ from states 5-10;
- **Dramatic Increase in Likelihood-** For both the humans and the algorithm, the likelihood of the correct kit increase dramatically at State 4, where the percentages went up to about 100% each.



Humans’ Likelihood Data for All Five Kits



Algorithm Likelihood Data for All Five Kits

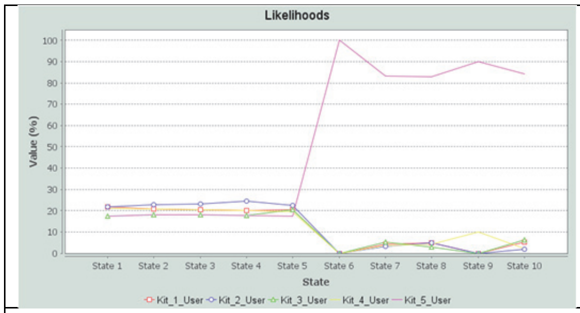


Comparison of Human and Algorithm Data for Kit 5 (Correct Kit)

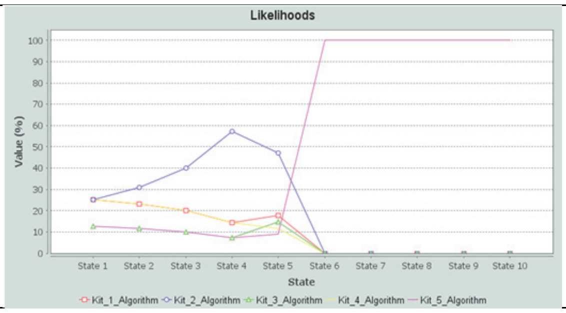
Figure 80: Human/Algorithm Comparison Data for Kit 5 Plan 2

Observations from Figure 80:

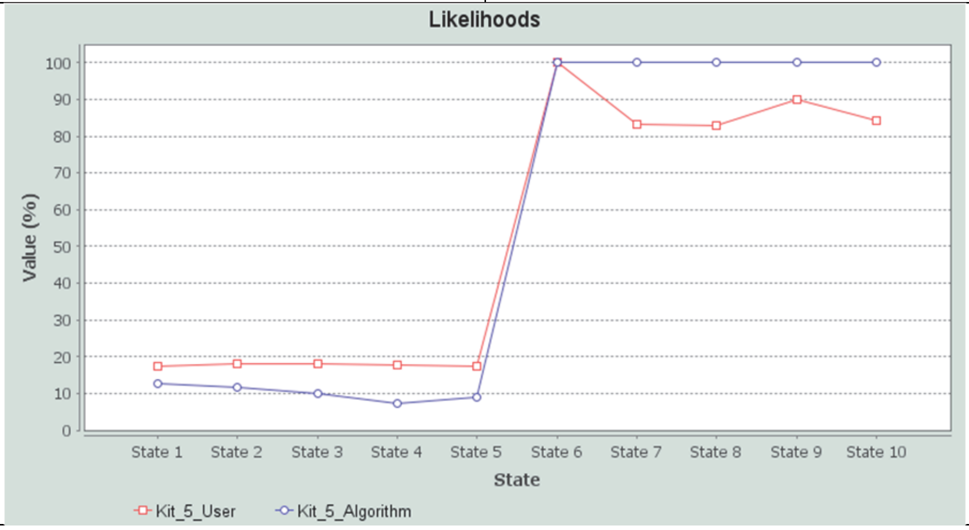
- **Pattern of the Lines**- Both the red line representing the humans’ likelihoods and the blue line representing the algorithm’s likelihoods at each state have almost identical patterns. However, the humans’ line has a slight dip at State 4 which does not exist in the algorithm’s line;
- **Likelihoods at Every State-** The algorithm’s likelihood at each state is greater than or equal to that of the humans’, thus indicating better results by the algorithm;
- **Dramatic Increase in Likelihood-** For both the humans and the algorithm, the likelihood of the correct kit increases dramatically at State 3, where the percentages increase to 100% each.



Humans' Likelihood Data for All Five Kits



Algorithm Likelihood Data for All Five Kits

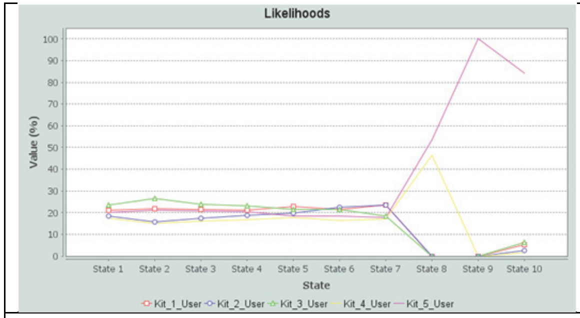


Comparison of Human and Algorithm Data for Kit 5 (Correct Kit)

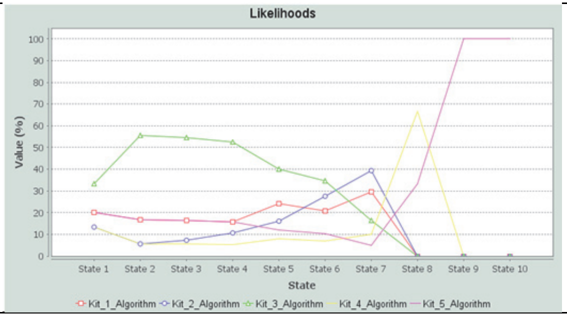
Figure 81: Human/Algorithm Comparison Data for Kit 5 Plan 3

Observations from Figure 81:

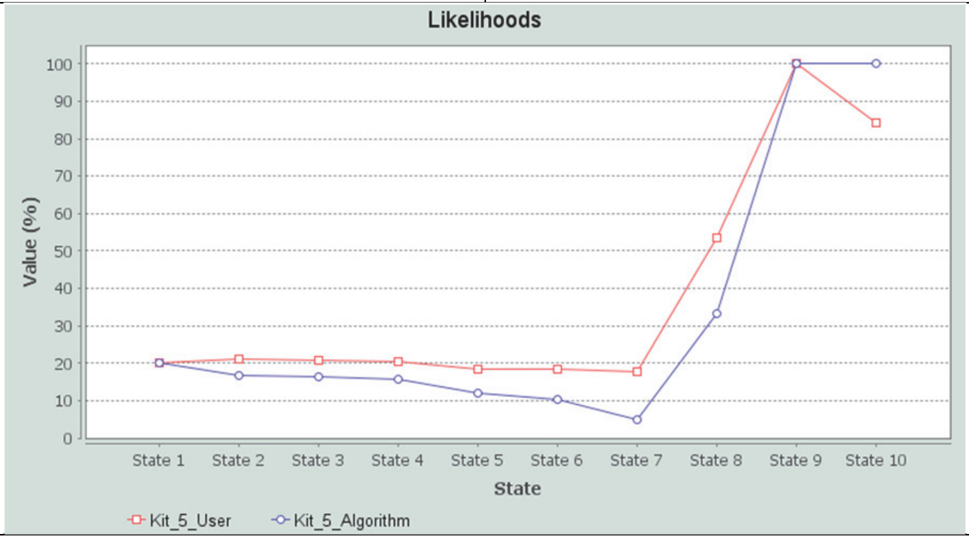
- **Pattern of the Lines**- Both the red line representing the humans' likelihoods and the blue line representing the algorithm's likelihoods at each state have almost identical patterns. However, the humans' line has a slight dip at State 7 which does not exist in the algorithm's line;
- **Likelihoods at Every State-** From States 1-5, the humans' likelihood is slightly greater than the algorithm's. Start at States 6 - 10, the algorithm's likelihood is slighter greater or equal to the humans';
- **Dramatic Increase in Likelihood-** For both the humans and the algorithm, the likelihood of the correct kit increases dramatically at State 6, where the percentages increase to 100% each.



Humans' Likelihood Data for All Five Kits



Algorithm Likelihood Data for All Five Kits

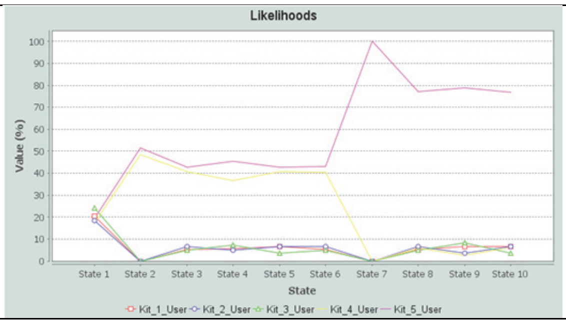


Comparison of Human and Algorithm Data for Kit 5 (Correct Kit)

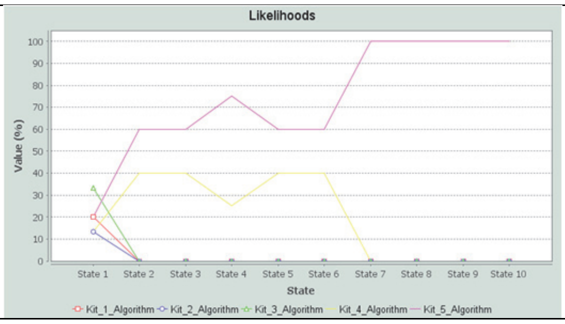
Figure 82: Human/Algorithm Comparison Data for Kit 5 Plan 4

Observations from Figure 82:

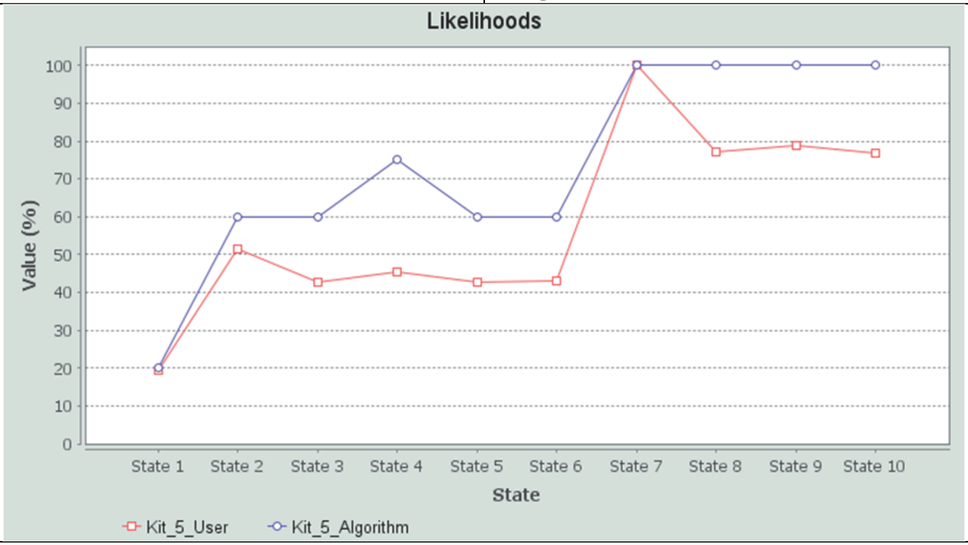
- **Pattern of the Lines**- Both the red line representing the humans' likelihoods and the blue line representing the algorithm's likelihoods at each state have almost identical patterns. However, the humans' line has a slight dip at State 10 which does not exist in the algorithm's line;
- **Likelihoods at Every State-** The humans' line has a slightly higher likelihood from States 1-8 as compared to the algorithm's line. Starting in State 9, the algorithm's line has a slightly greater or equal to likelihood as compared to the humans' line;
- **Dramatic Increase in Likelihood-** For both the humans and the algorithm, the likelihood of the correct kit increases dramatically from States 8-9, where the percentages increase to 100% each.



Humans’ Likelihood Data for All Five Kits



Algorithm Likelihood Data for All Five Kits



Comparison of Human and Algorithm Data for Kit 5 (Correct Kit)

Figure 83: Human/Algorithm Comparison Data for Kit 5 Plan 5

Observations from Figure 83:

- **Pattern of the Lines**- Both the red line representing the humans’ likelihoods and the blue line representing the algorithm’s likelihoods at each state have almost identical patterns. However, the algorithm’s line has a more pronounced jump at State 4 and the humans’ line has a dip at States 3 and 8;
- **Likelihoods at Every State-** The algorithm’s likelihood at each state is greater than or equal to that of the humans’, thus indicating better results by the algorithm;
- **Dramatic Increase in Likelihood-** For both the humans and the algorithm, the likelihood of the correct kit increases dramatically at State 2 and State 7. At State 7, the likelihoods from the humans’ and the algorithm increase to 100% each.

6.4. Experimental Results (Overall Observations)

The analysis in Sections 6.2 and 6.3 shows that the results of the Intention Recognition Algorithm closely replicated the performance of humans performing the same experiment and in some cases, exceeded their performance. This was determined by comparing the output from the experiment by looking at the following factors:

- **The state in which algorithm and the humans first identified the correct intention (and consistently identified it throughout the rest of the run)-** We showed that in over half of the runs (13/25), the Intention Recognition Algorithm determined the proper kit at the exact same state as the humans (as indicated by the zero in the difference column). In eight of the runs, the Intention Recognition Algorithm determined the proper kit earlier than the humans. In only four runs did the Intention Recognition Algorithm determine the proper kits later than the humans. In three cases, this was one state later and in the other it was three states later.
- **The state in which the algorithm and the humans were greater than 20% confident (as compared to the next most probable intention)-** In almost half of the runs (12/25), the Intention Recognition Algorithm determined the proper kit (with over 20% confidence compared to the next most probably kit) at the exact same state as the humans. In all of the remaining runs, the Intention Recognition Algorithm determined the proper kit earlier than the humans. This ranged from one to five states earlier than the humans.
- **The comparison of the pattern of lines representing the humans and the algorithm results-** In other words, do these lines have the same general pattern. In almost every one of the 25 runs, the general pattern of the lines representing the humans' and algorithm's results were almost identical. The exact values of the percentages often varied by a small amount, but the general path of the lines was extremely comparable. This shows that the algorithms exhibit the same pattern of intention recognition as a humans as the plan progresses, which indicates that the algorithms are mimicking the "thinking" of the humans.
- **The comparison of the likelihoods at every state for the lines representing the results of the humans and the algorithm for the correct kit-** Out of the 250 possible states (5 kits * 5 plans/kit * 10 states/plan), 189 times (75.6%) the algorithm performed better than the humans, as indicated by a higher percentage for the correct intention at each state. Twenty times (8%) the humans and the algorithm performed the same, and 41 times (16.4%) the humans performed better than the algorithm.
- **The comparison between the humans and the algorithm regarding the state in which the likelihood of the correct kit dramatically increased-** This represents the point in which the humans and algorithm became very confident in their guesses. As shown in Section 6.3, in 20 of the 25 plans, there was exactly one dramatic increase in likelihood and both the humans and the algorithm showed this increase at the exact same state. In one of the plans Kit 5,Plan 5) there were

two dramatic increases and both the humans and the algorithm showed this increase at the exact same state. In the remaining four plans, the algorithms showed a dramatic increase at the same state as the humans, but also introduced an additional dramatic increase that was not present in the humans' results. In all of these cases, this additional increase only brought the likelihood to close to 50%, while all of the others brought the likelihood into the 80-100% range. This indicates that the algorithm became much more confident that the intention was being performed, but was not "certain" as would be indicated by a likelihood in the 80-100% range.

From these five metrics, we can infer that the Intention Recognition Algorithm performed at least as well (often superior) as a humans in almost all cases and showed the same pattern of intention recognition as a humans as the plan progressed.

6.5. Applying Addition Intention Recognition Approaches

Overall, the Intention Recognition Algorithms performed superior to humans. Part of this could be due to natural human error in recognizing intentions. However, I was curious to see if, apart from human error, there were other factors that were affecting the difference between the algorithm and human results. Interestingly, the three other intention algorithms described in Section 4.2.4 seemed to play a large factor in accounting for the difference. The analysis below shows that we can mimic a human's behavior, so that we can better understand what a human takes into account when performing intention recognition. This will allow us to be better prepared to develop algorithms that leverage the best of these approaches.

As described in Section 4.2.4, in addition to the Bayesian approach, there were also three other intention recognition approaches that were explored, namely:

- Intention Recognition Approach 1: Number of observed state relations that are true in an intention at a given state (compared to other intentions). Every intention is composed of a set of state relations. The number of state relations that are true in one intention, as compared to others, should give an indication of the relative likelihood of that intention;
- Intention Recognition Approach 2: Percentage of an intention that is complete at a given state. As opposed to the number of true state relations, the percentage of state relations that are true in an intention could also provide a relative likelihood of that intention occurring;
- Intention Recognition Approach 3: Number of productive states that have occurred recently (within the past X states, where X is determined by the user). This approach looks at what intention is most probable based on what has happened recently. The supposition is that the more positive outcomes that

have happened in the past X states, the more likely that a given intention is occurring.

In this section, we apply these additional three approaches to determine how they affect the performance of the overall likelihood equation shown in Equation 46. Applying these additional three approaches to the experiment described earlier showed some interesting results. Each approach on its own did not provide results that were superior to that of the humans. However, the introduction of these approaches to Equation 46 with relatively small weights combined with the Bayesian approach containing a relatively high weight provided results that better mimicked the detailed performance of the humans. In all cases, the quantitative metrics (average state in which the kit was first detected and state which the likelihood was over 20 % greater) described in Section 6.1 did not change. However, by varying the weights assigned with the additional three approaches, we observed that the similarity of the line generated by Equation 46 as compared to the line generated by the humans increased. In this case, we assigned equal weights to all three additional approaches for simplicity purposes, but these approaches could have easily been assigned differing weights.

We analyzed the difference between the line generated by Equation 46 and the line generated by the humans. We measured the difference by looking at the sum of the absolute distance between corresponding points on each of the two lines at each of the ten states. This is shown in Table 19.

Table 19: Impact of Weights on First Three Approaches - Coarse Level

Weights Assigned to Approaches Format: (Approach 1 Weight – Approach 2 Weight – Approach 3 Weight – Bayesian Approach Weight)	**Sum of Differences** (measured in likelihood points)	**Standard Deviation**
0-0-0-100	83.03	6.16
10-10-10-100	42.97	3.71
20-20-20-100	56.88	7.83
30-30-30-100	66.41	11.36
40-40-40-100	79.09	13.58
50-50-50-100	85.58	15.06
60-60-60-100	94.33	16.23
70-70-70-100	103.86	17.32
80-80-80-100	111.28	18.04
90-90-90-100	114.99	18.37
100-100-100-100	118.12	19.01

Based on Table 19, there appears to be a significant reduction in the difference between the two lines when applying weights from zero to ten to the first three approaches, with a gradual and continuous increase when weights greater than ten are applied. Exploring the weights between zero and ten provided the data in Table 20. Table 19 and Table 20 combined are shown graphically in Figure 84.

Table 20: Impact of Weights on First Three Approaches: Fine-Grained Level

Weights Assigned to Approaches **Format: (Approach 1 Weight – Approach 2 Weight – Approach 3 Weight – Bayesian Approach Weight)**	**Sum of Differences** **(measured in likelihood points)**	**Standard Deviation**
1-1-1-100	77.14	5.40
2-2-2-100	65.01	5.32
3-3-3-100	55.84	4.74
4-4-4-100	48.03	4.85
5-5-5-100	37.42	4.57
6-6-6-100	20.48	2.50
7-7-7-100	38.13	3.94
8-8-8-100	41.21	3.75
9-9-9-100	41.90	3.88

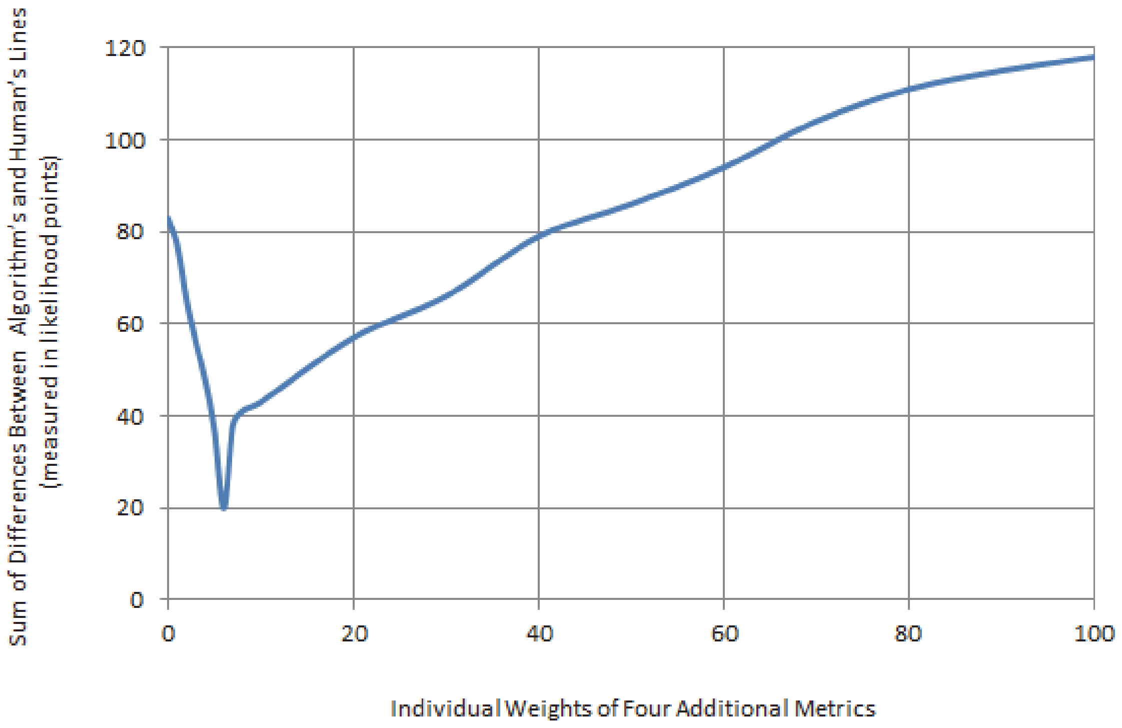


Figure 84: Individual Approaches Weights Effect on Sum of Difference

Figure 84 shows that there is a clear pattern in the weight assigned to these three additional approaches as compared to the “closeness” of the resulting Equation 46 line with that of the humans’. At a weight of six, the two lines most closely align and then consistently deviate as the weight move further away in both directions. The comparison of the humans' performance, the Bayesian approach by itself, and the addition of the three approaches at a weight of six are shown in Figure 85. As can be seen in the figure, the humans' performance (represented by the blue line) and the addition of the three approaches (represented by the green line), while not exactly the same, align quite well.

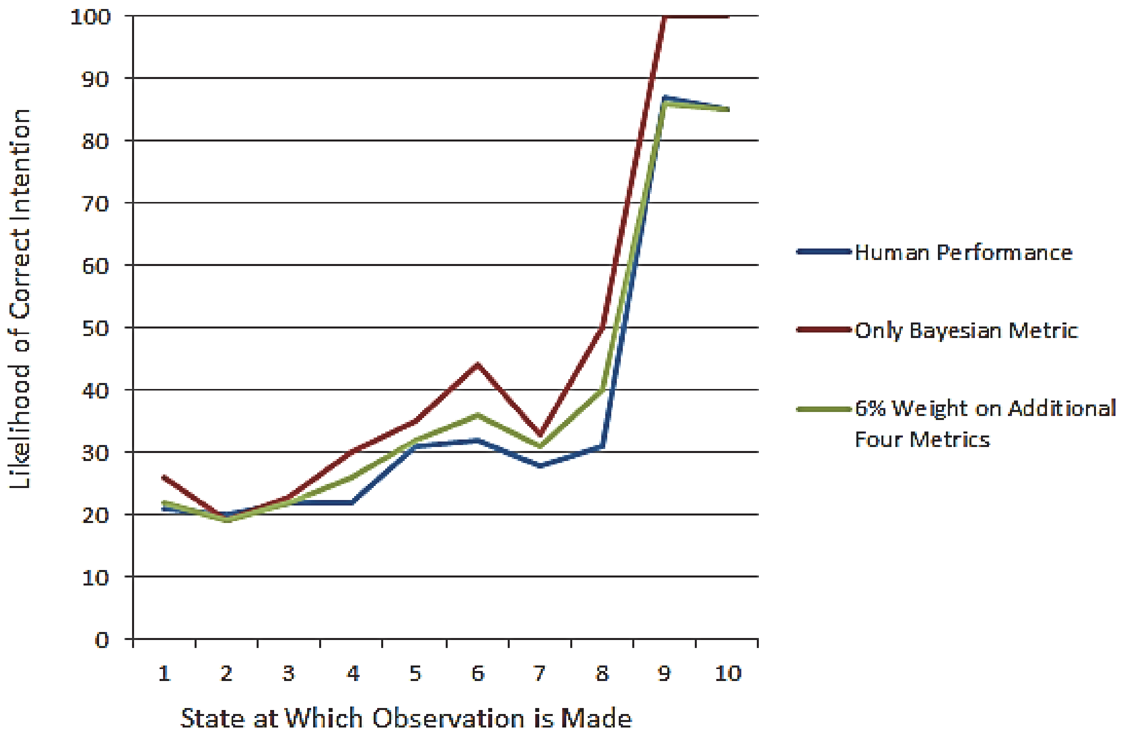


Figure 85: Comparison of Human Bayesian and Additional Approaches for Kit 2, Plan 4

The results are interesting for the following reasons. From the original experiment, it was clear that humans' performance followed that of a traditional Bayesian probability algorithm in the sense that the forms of the lines were very similar, though the actual location of the lines on the graph did not align (the algorithm line was generally higher on the graph than the humans' line). The addition of the three new approaches started to explain this difference. When these additional approaches were introduced at a relatively small weight as compared to the Bayesian approach, the resulting algorithm line greatly approached that of the humans. This may indicate that the humans were consciously or unconsciously considering the other three approaches when anticipating which kit was being developed, namely, the number of true observed states, the percentage of the intention complete, and the number of productive states that have occurred recently. Based on the data, a relative weight of six percent for these additional three approaches (as compared to the Bayesian approach) appear to best represent the thought process of the humans.

This was meant to be an initial exploration of these additional three approaches. More experiments would be necessary to determine if the six percent relative weight holds for other situations and for different sets of humans. Additional exploration would also be necessary to determine how independently varying the weights of these three additional approaches would change the results and which of the approaches, if any, contribute to the best correspondence with the humans' performance. The goal of this analysis was to determine if there appears to be a benefit in introducing these additional three approaches, which appears to be the case.

7. Conclusion and Future Work

7.1. Achievement

In this thesis, we presented a novel approach for inferring the intention of an agent in the environment via the recognition and representation of state information. This approach to intention recognition is different than many ontology-based intention recognition approaches in the literature as they primarily focus on activity (as opposed to state) recognition and then use a form of abduction to provide explanations for observations. We inferred detailed state relationships using observations based on Region Connection Calculus 8 (RCC-8) and then combined the RCC-8 relations to define the overall state relationships that are true at a given time. Once a sequence of state relationships was determined, we used a Bayesian approach to associate those states with likely overall intentions to determine the next possible action (and associated state) that were likely to occur.

The five main contributions of this thesis are as follows:

- State Recognition and Representation Algorithms: These algorithms recognize relevant qualitative state relations in the environment and represent them using a combination of extended RCC-8 relations and cardinal direction information. These low-level state relations are combined to form more abstract state relations which are used as the basis to represent intentions. In this work, a sequence of state relations constitutes an intention.
- Intention Recognition Algorithms: These algorithms compare observation of qualitative state information against predefined intentions which are possible in the environment. Based on these comparisons, the algorithms assign likelihoods to each relevant intention. Likelihoods are determined through a weighted average among various intention recognition approaches, with the primary approach being Bayesian.
- Manufacturing Kitting Ontology: This ontology represents the key concepts relevant to the manufacturing kitting domain. It serves as the basis for the intention recognition work by defining the relevant concepts and explicitly defining the relationships between the concepts.
- State Representation Ontology: This ontology representation the various levels of qualitative state information need for the intention recognition work. At the lowest level, the RCC8 relations are represented along each axial plane. A set of logical expressions are represented to abstract the RCC8 relations into intermediate state relations and ultimately to predicates that can be used as the basis for intention recognition.
- Intention Ontology: This ontology leverages and modifies the OWL-S ontology to construct intentions based on the combination and sequencing of state

information. Sequencing constructs are captured in this ontology along with constructs to represent the relevant intentions for the domains being explored.

An experiment was developed to assess the performance of the state-based Intention Recognition Algorithm described above. This was done by comparing the output of our algorithm to the performance of several humans watching agents performing the same intentions. Five kits were used that each contained ten blocks, but each contained a unique combination of differently-shaped blocks. For this experiment, we randomly chose five of those orders for each kit, resulting in 25 total runs.

The results of the experiment showed very promising results:

- In over half of the runs (13/25), the Intention Recognition Algorithm determined the proper kit at the exact same state as the humans. In eight of the runs, the Intention Recognition Algorithm determined the proper kit earlier than the humans. In only four runs did the Intention Recognition Algorithm determine the proper kits later than the humans. In three cases, this was one state later and in the other it was three states later;
- In almost half of the runs (12/25), the Intention Recognition Algorithm determined the proper kit (with over 20% confidence compared to the next most probably kit) at the exact same state as the humans. In all of the remaining runs, the Intention Recognition Algorithm determined the proper kit earlier than the humans. This ranged from one to five states earlier than the humans;
- In almost all of the 25 runs, the general pattern of the lines representing the humans' and algorithm's results were almost identical. The exact values of the percentages often varied by usually a small amount, but the general form of the lines was extremely comparable. This showed that the algorithms mimicked the same pattern of intention recognition as a human as the plan progressed, thus showing it was "thinking" like a human.
- In more than three-quarters of the 250 possible states (5 kits * 5 plans/kit * 10 states/plan), the algorithm performed better that the humans (189 times, 75.6%), as indicated by a higher percentage for the correct kit at each state. The humans and the algorithm performed the same 20 times (8%), and the humans performed better than the algorithm 41 times (16.4%).

7.2. Future Work

Although substantial work has been performed, there are extensions of this work that will be the focus of future efforts. First, the State-Based Intention Recognition approach will be applied to areas outside of the manufacturing kitting domain. The State Recognition Algorithm as well as the Intention Recognition Algorithm have been developed to be generic in nature, but because they have only been applied to the manufacturing kitting domain, there are undoubtedly additional extensions that will be needed. The next area to which they will be applied to is manufacturing assembly, which is an extension of manufacturing kitting. Additional high-level state relations that will need to be developed include concepts such as "*attached_to.*" Additional intermediate state relations may also needed. For example, in the case of a screw being inserted into a hole, there may need to be different levels of "Partially_In" to differentiate when a screw is partially screwed in vs. when it is completely screwed in. Lastly, additional intentions will need to be developed that align to the product that is being assembled.

Second, the algorithms have currently only been testing in simulation where we assume that sensor processing is perfect and relevant state relationships can be determined without any specified uncertainty. We have already begun establishing the infrastructure to transition this to a real environment as described in Chapter 5, but more work needs to be done to make this a reality. This includes developing the interfaces needed to task the sensory processing system to return location and orientation information about objects in the environment. Considering the results of the sensory processing system will not be perfect, we need a way to represent uncertainty in the output of these systems and to apply this uncertainty to the results of the State Recognition Algorithms. Uncertainty can come from the recognition of the object, the identification of the position of the object, and the identification of the orientation of the object. The Bayesian approach can still be used even when the input state relations have an associated uncertainty. For example, if the observation states that there is a 70% chance that a blue part is in the tray, a 20% probability that a red part is in the tray, and a 10% probability that nothing is in the tray, the Bayesian probability can still be used for each scenario and then a weighted average can be applied to the three scenarios to determine an overall probability.

Third, in Chapter 4, we described additional approaches to intention recognition. This included approaches such as: the number of observed state relations that were true in an intention; the percentage of an intention that were complete; and the number of productive states that occurred recently. Initial exploration applied these approaches to the experiment, but more work is needed to determine the impact that each approach has on the performance of the overall intention recognition system by varying the weights of each independently and determining the optimal set of weights for all of the approaches. Additional experiments can also try to find the optimal weights to maximize the difference between the algorithms' and humans' results, ensuring that the algorithms' results are always greater.

Fourth, though learning was not specifically addressed in this thesis, there are a number of ways that it can be incorporated. They include:

1. In the Bayesian probability equation (Equation 47), part of the equation focused on the proportion of all kits that are a specific type of kit. This number can be "learned" over time by look at the historical proportion of

kits that are different types and modifying this number in real time as new kits are being developed.

2. In the overall likelihood equation (Equation 46), there are multiple intention recognition algorithms proposed, each with a separate weight. These weights can be “learned” over time by looking at the kits that have been developed and modifying the weights to provide optimal values based on the development of previous kits.

Bibliography

[1] S. Szabo, R. Norcross, and W. Shackleford. (2011). *Safety of Human-Robot Collaboration Systems Project*. Available: http://www.nist.gov/el/isd/ps/safhumrobcollsys.cfm

[2] M. Shneier, "Safety of Human-Robot Collaboration in Manufacturing," presented at the 8th Safety Across High-Consequence Industries Conference, Saint Louis, MO, 2013.

[3] J. Chabrol, "Industrial robot standardization at ISO," *Robotics,* vol. 3, pp. 229-233, 1987.

[4] F. Sadri, "Logic-Based Approaches to Intention Recognition," in *Handbook of Research on Ambient Intelligence and Smart Environments: Trends and Perspectives*, N.-Y. Chong and F. Mastrogiovanni, Eds., ed, 2011, pp. 346-375.

[5] C. Heinze, "Modeling intention recognition for intelligent agent systems," Australia2003.

[6] C. Schlenoff, A. Pietromartire, Z. Kootbally, S. Balakirsky, T. Kramer, and S. Foufou, "Inferring Intention Through State Representation in Cooperative Human-Robot Environments," in *Engineering Creative Design in Robotics and Mechatronics*, M. Habib and P. Davim, Eds., ed, 2013.

[7] D. Randell, Z. Cui, and C. A., "A spatial logic based on regions and connection," presented at the 3rd International Conference on Representation and Reasoning, San Mateo, CA, 1992.

[8] D. Martin, M. Burstein, J. Hobbs, O. Lassila, D. McDermott, S. McIlrath*, et al.* (2004). *OWL-S: Semantic Markup of Web Services*. Available: http://www.w3.org/Submission/OWL-S/

[9] H. Jeon, T. Kim, and J. Choi, "Ontology-based User Intention Recognition for Proactive Planning of Intelligent Robot Behavior," presented at the International Conference on Multimedia and Ubiquitous Engineering Busan, Korea, 2008.

[10] R. Kelley, A. Tavakkoli, C. King, M. Nicolescu, M. Nicolescu, and G. Bebis, "Understanding Human Intentions Via Hidden Markov Models in Autonomous Mobile Robots," presented at the 3rd ACM/IEEE International Conference on Human Robot Interaction Amsterdam, 2008.

[11] O. Schrempf and U. Hanebeck, "A Generic Model for Estimating User-Intentions in Human-Robot Cooperation," presented at the 2nd International Conference on Informatics in Control, Automation, and Robotics ICINCO 05 Barcelona, 2005.

[12] W. Mao and J. Gratch, "A Utility-Based Approach to Intention Recognition," presented at the AAMAS Workshop on Agent Tracking: Modeling Other Agents from Observations New York, 2004.

[13] S.-J. Youn and K.-W. Oh, "Intention Recognition using a Graph Representation," *World Academy of Science, Engineering and Technology,* vol. 25, 2007.

[14] C. Schlenoff, S. Foufou, and S. Balakirsky, "Performance Evaluation of Robotic Knowledge Representation (PERK)," presented at the Performance Metrics for Intelligent Systems (PerMIS), College Park, MD, 2012.
[15] E. Charniak and D. McDermott, *Introduction to artificial intelligence*. Reading, MA: Addison Wesley, 1985.
[16] P. Roy, B. Bouchard, A. Bouzouane, and S. Giroux, "a hybrid plan recognition model for Alzheimer's patients: interleaved-erroneous dilemma," presented at the IEEE/WIC/ACM International Conference on Intelligent Agent Technology, 2007.
[17] G. Sukthanker and K. Sycara, "Team-aware robotic demining agents for military simulation," presented at the Innovative Applications of Artificial Intelligence (IAAI), 2001.
[18] L. M. Pereira and H. T. Ahn, "Elder care via intention recognition and evolution prospection," presented at the 18th International Conference on Applications of Declarative Programming and Knowledge Management (INAP'09), Evora, Portugal, 2009.
[19] F. Mulder and F. Voorbraak, "A formal description of tactical plan recognition," *Information Fusion,* vol. 4, 2003.
[20] P. A. Jarvis, T. F. Lunt, and K. L. Myers, "Identifying terrorist activity with AI plan-recognition technology," *AI Magazine,* vol. 26, p. 9, 2005.
[21] R. Demolombe, A. Mara, and O. Fern, "Intention recognition in the situation calculus and probability theory frameworks," presented at the Computational Logic in Multi-Agent Systems (CLIMA) Conference, 2006.
[22] M. Philipose, K. Fishkin, M. Perkowitz, D. Patterson, D. Hahnel, D. Fox*, et al.*, "Inferring ADLs from interactions with objects," *IEEE Pervasive Computing,* 2005.
[23] P. R. Cohen, C. R. Perrault, and J. F. Allen, "Beyond question answering," in *Strategies for Natural Language Processing*, W. Lenhart and M. Ringle, Eds., ed Hillsdale, NJ: Lawrence Erlbaum Associates, 1981, pp. 245-274.
[24] C. W. Geib and R. P. Goldman, "Plan recognition in intrusion detection systems," presented at the DARPA Information Survivability Conference and Exposition (DISCEX), 2001.
[25] L. M. Pereira and T. A. Han, "Intention recognition via causal Bayes networks plus plan generation," presented at the Progress in Artificial Intelligence, Proceedings of the 14th Portugese International Conference on Artificial Intelligence (EPIA'09), 2009.
[26] C. S. Pierce, *Collected Papers, Band VII (Hrsg.)*: Arthur W. Burks, 1958.
[27] P. Michalak, "Intention Recognition for Task Learning," 2005.
[28] E. Charniak and R. P. Goldman, "A bayesian model for plan recognition," *Artificial Intelligence,* vol. 64, pp. 53-79, 1993.
[29] T. A. Han and L. M. Pereira, "State-of-the-Art of Intention Recognition and its use in Decision Making," *Studies in Applied Philosophy, Epistemology and Rational Ethics,* vol. 2, pp. 263-287, 2012.
[30] K. Knonlige and M. Pollack, "Ascribing plans to agents: preliminary report," presented at the International Joint Conference on Artificial Intelligence, Detroit, MI, 1989.

[31] D. E. Appelt and M. E. Pollack, "Weighted abduction for plan ascription," *User Modeling and User-Adapted Interaction,* vol. 2, pp. 1-25, 1991.

[32] A. Waern, "Plan inference for a purpose," presented at the Fourth International Conference on User Modeling, Hyannis, MA, 1994.

[33] M. Bauer, "A Demster-Shafer approach to modeling agent preferences for plan recognition," *User Modeling and User-Adapted Interaction,* vol. 5, pp. 317-348, 1996.

[34] J. Pearl, *Probabilistic Reasoning in Intelligent Systems*: Morgan Kaufman, 1988.

[35] B. Raskutti and I. Zukerman, "Generation and selection of likely interpretations during plan recognition in task-oriented consultation systems," *User Modeling and User-Adapted Interaction,* vol. 1, pp. 323-353, 1991.

[36] D. W. Albrecht, I. Zukerman, and A. E. Nicholson, "Bayesian models for keyhold plan recognition in an adventure game," *User Modeling and User-Adapted Interaction,* vol. 8, pp. 5-47, 1998.

[37] T. Dean and T. Wellman, *Planning and Control*. San Mateo, California: Morgan Kaufman, 1991.

[38] D. V. Pynadath and M. P. Wellman, "Accounting for context in plan recognition with applications to traffic monitoring," presented at the Conference on Uncertainty in Artificial Intelligence (UAI'95), 1995.

[39] J. Forbes, T. Huang, K. Kanazawa, and S. Russell, "The batmobile: towards a Bayesian automated taxi," presented at the The 24th International Joint Conference on Artificial Intelligence, 1995.

[40] S. M. Brown, "A decision theoretic approach for interface agent development," USA1998.

[41] R. P. Goldman, C. W. Geib, and C. Miller, "A new model of plan recognition," presented at the Conference on Uncertainty in Artificial Intelligence (UAI'99), 1999.

[42] H. Bui, S. Venkatesh, and G. West, "Policy recognition in the abstract hidden markov model," *Journal of Artificial Intelligence Research,* vol. 17, pp. 451-499, 2002.

[43] N. Blaylock and J. Allen, "Corpus-based statistical goal recognition," presented at the 18th International Joint Conference on Artificial Intelligence (IJCAI'03), 2003.

[44] H. Bui, "A general model for online probabilistic plan recognition," presented at the 8th International Joint Conference on Artificial Intelligence (IJCAI'03), 2003.

[45] M. Huber and R. Simpson, "Recognizing the plans of screen reader users," presented at the Workshop on Modeling Other Agents from Observations (MOO2004), 2004.

[46] C. W. Geib, "Assessing the complexity of plan recognition," presented at the Conference of the American Association of Artificial Intelligence (AAAI'2004), 2004.

[47] K. A. Tahboub, "Intelligent human-machine interaction based on dynamic Bayesian networks providing probablistic intention recognition," *Journal of Intelligent Robotic Systems,* vol. 45, pp. 31-52, 2006.
[48] O. C. Schrempf, D. Albrecht, and E. D. Hanebeck, "Tractable probabilistic models for intention recognition based on expert knowledge," presented at the International Conference on Intelligent Robots and Systems, 2007.
[49] C. W. Geib and R. P. Goldman, "A probabilistic plan recognition algorithm based on plan tree grammars," *Artificial Intelligence,* vol. 173, pp. 1101-1132, 2009.
[50] M. G. Armentano and A. Amandi, "Goal recognition with variable-order markov model," presented at the 21st International Joint Conference on Artificial Intelligence, 2009.
[51] T. A. Han and L. M. Pereira, "Context-dependent and incremental intention recognition via Bayesian network model construction," *Journal of Artificial Intelligence Research (submitted),* 2011.
[52] M. T. Cox and B. Kerkez, "Case-based Plan Recognition with Novel Imputs," *International Journal of Control and Intelligent Systems,* vol. 34, pp. 96-104, 2006.
[53] J. L. Kolodner, *Case-based reasoning*. San Mateo, CA: Morgan Kaufman, 1993.
[54] B. Kerkez and M. T. Cox, "Incremental case-based plan recognition with local predictions," *International Journal on Artificial Intelligence Tools: Architectures, language, algorithms,* vol. 12, pp. 413-463, 2003.
[55] C. K. Riesbeck and R. C. Schank, *Inside case-based reasoning*. Hillsdale, NJ: Lawrence Erlbaum Associates, 1989.
[56] D. Leake, *Case-based reasoning: Experiences, lessons, & future directions*. Menlo Park, CA: AAAI Press / MIT Press, 1996.
[57] P. Taillandier, O. Therond, and B. Gaudou, "A new BDI agent architecture based on the belief theory. Application to teh modelling of cropping plan decision-making," presented at the International Congress on Environmental Modelling and Software : Managing Resources of a Lmited Planet, Leopzig, Germany, 2012.
[58] F. Dignum, D. Morley, E. A. Sonenberg, and L. Cavedon, "Towards socially sophisticated BDI agents," presented at the Fourth International Conference on MultiAgent Systems, 2000.
[59] A. Pokahr, L. Braubach, and W. Lamersdorf, "Jadex: A BDI Reasining Engine," in *Multi-Agent Programming: Languages, Platforms, and Applications*. vol. 15, ed, 2005, pp. 149-174.
[60] A. Guerra-Hernandez, A. E. Fallah-Seghrouchni, and H. Soldano, *Learning in BDI Multi-Agent Systems*, 2004.
[61] M. Rao and P. Georgeff, "Formal models and decision procedures for multi-agent systems," 1995.
[62] M. Rao and P. Georgeff, "BDI-agents: From Theory to Practice," presented at the First INternational Conference on Multiagent Systems (ICMAS), 1995.
[63] M. Georgeff, B. Pell, M. Pollack, M. Tambe, and M. Wooldridge, *The Belief-Desire-Intention Model of Agency*, 1999.

[64] A. Pokahr, L. Braubach, and W. Lamersdorf, *Jadex: A BDI Reasoning Engine*, 2005.
[65] S. Sardina, L. d. Silva, and L. Padgham, "Hierarchical planning in BDI agent programming langauges: a formal approach," presented at the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems, 2006.
[66] H. Kautz and J. Allen, "Generalized plan recognition," presented at the The Conference of the American Association of Artificial Intelligence (AAAI'1986), 1986.
[67] B. Karlsson, A. E. M. Ciarlini, B. Feijo, and A. L. Furtado, "Applying a plan-recognition / plan-generation paradigm to interactive storytelling, the LOGTELL case study," presented at the ICAPS06 Workshop on AI Planning for Computer Games and Synthetic Characters, Lake District, UK, 2006.
[68] B. A. Goodman and D. J. Litman, "On the interaction between plan recognition and intelligent interfaces," *User Modeling and User-Adapted Interactions,* vol. 2, pp. 83-115, 1992.
[69] N. Lesh, C. Rich, and C. L. Sidner, "Using plan recognition in human-computer collaboration," presented at the Seventh International Conference on User Modelling, Canada, 1999.
[70] M. H. Nguyen and W. Wobcke, "A Flexinble Framework for SharedPlan," presented at the Advances in Artificial Intelligence19th Australian Joint Conference on Artificial Intelligence, Hobart, Australia, 2006.
[71] S. Giroux, J. Bauchet, H. Pigot, D. Lusser-Desrochers, and Y. Lachappelle, "Pervasive behavior tracking for cognitive assistance," presented at the Third International Conference on Pervasive Technologies Related to Assistive Environments (PETRA'08), Greece, 2008.
[72] C. Baum and D. Edwards, "Cognitive Performance in Senile Dementia of the Alzheimer's Type: The Kitchen Task Assessment," *American Journal of Occupational Therapy,* vol. 1, 1993.
[73] S. Carberry and S. Elzer, *Exploiting evidence analysis in plan recognition*. Berlin Heidelberg: Springer-Verlag, 2007.
[74] Z. Wang, M. Deisenroth, H. B. Amor, D. Vogt, N. Scholkopf, and J. Peters, "Probabilistic Modeling of Human Movements for Intention Inference," in *Robotics: Science and Systems*, Sydney, Australia, 2012.
[75] H. Goto, J. Miura, and J. Sugiyama, "Human-Robot Collaborative Assembly by On-line Human Action Recognition Based on an FSM Task Model," presented at the Human-Robot Interaction 2013 Workshop on Collaborative Manipulation, Tokyo, 2013.
[76] D. C. Cheng and R. Thowanmas, "Case-based plan recognition for real-time strategy games," presented at the 5th Game-On International Conference (CGAIDE'04), Reading, UK, 2004.
[77] M. Buro and T. Furtak, "RTS Games as Testbeds for Real-Time Research," presented at the Workshop on Game AI, JCIS, 2003.
[78] J. Mayfield, "Evaluating Plan Recognition Systems: Three Properties of a Good Explanation," *Artificial Intelligence Review,* vol. 14, pp. 351-376, 2000.
[79] DARPA, "Information Innovation Office: Mind's Eye Program," 2012.

[80] J. Marvel, T.-H. Hong, and E. Messina, "2011 Solutions in Perception Challenge Performance Metrics and Results," presented at the Performance Metrics for Intelligent Systems (PerMIS) Conference, College Park, Maryland, 2012.
[81] M. Newman and S. Balakirsky. (2011) Contests in China Put Next-Generation Robot Technology to the Test. *IEEE Robotics and Automation Magazine, DOI 10.1109/MRA.2011.942540*.
[82] T. Choudhury and G. Borriello. (2008) The Mobile Sensing Platform: An Embedded System for Activity Recognition. *IEEE Pervasive Magazine- Special Issue on Activity-Based Computing*.
[83] N. Ravi, N. Dandekar, P. Mysore, and M. Littman, "Activity Recognition from Accelerometer Data," presented at the Seventeenth Conference on Innovative Applications of Artificial Intelligence (IAAI/AAAI), 2005.
[84] R. Bodor, B. Jackson, and N. Papanikolopoulos, "Vision-Based Human Tracking and Activity Recognition," presented at the 11th Mediterranean Conference on Control and Automation, 2003.
[85] A. Hoogs and A. G. A. Perera, "Video Activity Recognition in the Real World," presented at the American Association of Artificial Intelligence (AAAI) Conference, 2008.
[86] J. Bateman, "Situating Spatial Langauge and the Role of Ontology: Issues and Outlook," *Langauge and Linguistic Compass,* vol. 4, pp. 639-664, 2010.
[87] B. Smith and P. Grenon, "The cornucopia of formal ontological relations," *Dialectica,* vol. 58, pp. 279-296, 2004.
[88] C. Masolo, S. Borgo, A. Gangemi, N. Guarino, and A. Oltramari, "Ontologies library (final)," Padova, Italy2003.
[89] B. Heller and H. Herre, "General ontological language GOL: a formal framework for building and representing ontologies," Leipzig, Germany2004.
[90] N. Asher and L. Vieu, "Towards a geometry of common sense: a semantics and a complete axiomatisation of mereotopology," presented at the 15th International Joint Conference on Atrificial Intelligence (IJCAI), San Mateo, CA, 1995.
[91] D. Lenat and R. Guha, *Building Large Scale Knowledge-Based Systems: Representation and Inference in the Cyc Project*. Addison-Wesley, 1990.
[92] A. Philpot, E. Hovy, and P. Pantel, "The omega ontology," presented at the ONTOLEX Workshop at IJCNLP, 2005.
[93] I. Niles and A. Pease, "Towards a Standard Upper Ontology," 2001.
[94] D. Oberle, A. Ankolekar, P. Hitzler, P. Cimiano, M. Sintek, M. Kiesel*, et al.*, "DOLCE ergo SUMO: on foundational and domain models in the SmartWeb Integrated Ontology (SWIntO)," *Journal of Web Semantics,* vol. 5, pp. 156-174, 2007.
[95] R. Casati and A. C. Varzi, *Holes and other superficialities*. Cambridge, MA and London: MIT Press (Bradford Books), 1994.
[96] Cycorp, "OpenCyc 0.7.0 Technical Report," 2004.
[97] J. Bateman and S. Farrar, "Spatial ontology baseline," University of Bremen, Germany: Collaborative Research Center for Spatial Cognition2004.

[98] P. Buitelaar, "Semantic Lexicons: Between Terminology and Ontology," presented at the OntoLex Workshop on Ontologies and Lexical Knowledge Bases, Bulgaria, 2000.

[99] N. Guarino and C. Welty, "An overview of OntoClean," in *Handbook on Ontologies*, S. Staab and R. Studer, Eds., ed Heidelberg and Berlin: Springer-Verlag, 2004, pp. 151-171.

[100] C. Welty and W. Anderson, "Towards OntoClean 2.0: a framework for rigidity," *Applied Ontology,* vol. 1, pp. 107-116, 2005.

[101] J. Albrecht, "Towards interoperable geo-information standards: a comparison of reference models for geo-spatial information," *Annals of Regional Science,* vol. 33, pp. 151-169, 1999.

[102] P. Vossen, *Euro WordNet: a multilingual database with lexical semantics networks*. Dordrecht: Kluwer Academic Publishers, 1998.

[103] C. F. Baker, C. Fillmore, and J. Lowe, "The Berkeley FrameNet Project," presented at the ACL/COLING, Montreal, Quebec, 1998.

[104] G. Miller, "WordNet: an online lexical database," *International Journal of Lexicography,* vol. 3, 1990.

[105] K. Kipper-Schuler, "VerbNet: a broad-coverage, comp[rehensive verb lexicon," Philadelphia, PA2005.

[106] J. d. Kleer and J. S. Brown, *A qualitative physics based on confluence. Formal theories of the commonsense world*. Norwood, NJ: Ablex Publishing Corporation, 1985.

[107] P. Simon, *Parts: a study of ontology*. Oxford: Clarendon Press, 1987.

[108] D. Mark, D. Comas, M. Egenhofer, S. Freundschuh, J. Gould, and J. Nunes, "Evaluating and refining computational models of spatial relations through cross-linguistic human-subjects testing.," in *Spatialinformation theory: a theoretical basis for GIS*, A. Frank, Ed., ed Berlin: Springer-Verlag, 1995, pp. 553-568.

[109] J. Renz, R. Rauh, and M. Knauff, "Towards cognitive adequancy of topological spatial relations," in *Spatial cognition II - integrating abstract theories, emperical studies, formal methods, and practical applications*, C. Freksa, W. Brauer, C. Habel, and K. F. Wender, Eds., ed Berlin: Springer, 2000, pp. 184-197.

[110] R. Rauh, C. Hagen, M. Knauff, T. Kuss, C. Schlieder, and G. Strube, "Preferred and alternative mental models in spatial reasoning," *Spatial Cognition and Computation* vol. 5, pp. 239-269, 2005.

[111] A. Klippel, R. Kai-Florian, T. Barkowsky, and C. Freksa, "The cognitive reality of schematic maps," in *Mp-based mobile services - theories, methods, and implementations*, L. Meng, A. Zipf, and T. Reichenberger, Eds., ed Berlin: Springer, 2005, pp. 96-100.

[112] S. Kuehne and K. Forbus, "Qualitative physics as a component in natural langauge semantics: a progress report," presented at the 24th Annual Meeting of the Cognitive Science Society, Fairfax, VA, 2002.

[113] M. Knauff, "Stop using introspection to gather data for the design of computational modeling and spatial assistance," presented at the AAAI

Spring Symposium on Reasoning with Mental and External Diagrams: Computational Modeling anf Spatial Assistance, Menlo Park, CA, 2005.

[114] F. Wolter and M. Zakharyaschev, "Spatio-temporal representation and reasoning based on RCC-8," presented at the 7th Conference on Principles of Knowledge Representation and Reasoning (KR2000), Breckenridge, CO, 2000.

[115] J. Albath, J. Leopold, C. Sabharwal, and A. Maglia, "RCC-3D: Qualitative Spatial Reasoning in 3D," presented at the 23rd International Conference on Computer Applications in Industry and Engineering (CAINE), Las Vegas, NV, 2010.

[116] G. Ligozat, "Qualitative triangulation for spatial reasoning," in *CPSIT 1993*. vol. 716, I. Campari and A. U. Frank, Eds., ed Heidelberg: Springer, 1993, pp. 54-68.

[117] C. Freksa, "Using orientation information for Qualitative spatial reasoning," in *Theories and methods of spatio-temporal reasoning in geographic space* A. U. Frank, I. Campari, and U. Formentini, Eds., ed Heidelberg: Springer, 1992, pp. 162-178.

[118] C. Schlieder, "Reasoning about ordering," in *COSIT*. vol. 988, W. Kuhn and A. U. Frank, Eds., ed Heidelberg: Springer, 1995, pp. 341-349.

[119] R. Moratz, F. Dylla, and J. Frommberger, "A relative orientation algebra with adjustable granularity," presented at the Proceedings of the Workshop on Agents in Real-Time and Dynamic Environments, IJCAI Edinburgh, Scotland, 2005.

[120] J. O. Wallgrun, L. Frommberger, D. Wolter, F. Dylla, and C. Freksa, "Qualitative spatial representation and reasoning in the SparQ-Toolbox," in *Spatial Cognition V*, T. Barkowsky, M. Knauff, G. Ligozat, and D. R. Montello, Eds., ed: Springer, 2006, pp. 39-58.

[121] J. Renz and B. Nebel, "On the complexity of qualitative spatial reasoning; a maximal tractable fragment of the region connection calculus," *Artificial Intelligence,* vol. 108, pp. 69-123, 1999.

[122] N. V. d. Weghe, A. G. Cohn, P. D. Maeyer, and F. Witlox, "Representing moving objects in computer-based expert systems that overtake event examples," *Expert Systems with Applications,* vol. 29, pp. 977-983, 2005.

[123] A. G. Cohn and S. M. Hazarika, "Qualitative spatial representation and reasoning: An overview," *Fundamenta Informaticae,* vol. 43, pp. 2-32, 2001.

[124] R. Moratz, T. Tenbrink, J. Bateman, and K. Fischer, "Spatial Knowledge Representation for Human-Robot Interaction," in *Spatial Cognition III - Lecture Nores in Computer Science*. vol. 2685, ed, 2003, pp. 263-285.

[125] J. Bateman and S. Farrar, "Spatial Ontology Baseline Version 2.0," University of Bremen2006.

[126] J. Bateman and S. Farrar, "Towards a generic foundation for spatial ontology," presented at the Formal Ontology in Information Systems (FOIS), 2004.

[127] F. Jackendoff, "The architecture of the linguistic-spatial interface," in *Language and space*, P. Bloom, M. A. Peterson, L. Nadel, and M. F. Garrett, Eds., ed Cambridge, MA: MIT Press, 1999, pp. 1-30.

[128] E. Stopp, K. Gapp, G. Herzog, T. Laengle, and T. Lueth, "Utilizing spatial relations for natural language access to an autonomous mobile robot," presented at the KI-94, 1994.
[129] P. Olivier and H. Tsujii, "A computational view of the cognitive semantics of spatial prepositions," presented at the 32nd Annual Meeting of the Association for Commputational Linguistics, Las Cruces, NM, 1994.
[130] C. Kray and A. Blocher, "Modeling the basic meaning of path relations," presented at the 16th INternational Joint Conference on Artificial Intelligence (IJCAI), Los Altos, CA, 1999.
[131] M. Levit and D. Roy, "Interpretation of spatial language in a map navigation task," *IEEE Transactions on Systems, Man, and Cybernetics,* vol. 37, pp. 667-679, 2006.
[132] N. Mavridis and D. Roy, "Grounded situation models for robots: Where words and percepts meet," presented at the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2006.
[133] L. Steels and M. Loetzsch, "Perspective alignment in spatial language," in *Spatial language and dialogue*, K. Coventry, T. Tenbrink, and J. Bateman, Eds., ed Oxford: Oxford University Press, 2009, pp. 70-88.
[134] A. Rashid, N. M. Sharif, M. J. Egenhofer, and D. M. Mark, "Natutal-language spatial relations between linear and areal objects: the topology and metric of English-language terms," *International Journal of Geographical Information Science,* vol. 12, pp. 215-246, 1998.
[135] M. Egenhofer, A. Rashid, and B. M. Shariff, "Metric details for natural-language spatial relations," *ACM Transactions on Information Systems,* vol. 16, pp. 295-321, 1998.
[136] D. M. Mark, W. Kuhn, B. Smith, and A. G. Turk, "Ontology, natiural language, and information systems: implications of cross-linguistic studies on geographic terms," presented at the 6th AGILE conference on geographic information science, Lyon, France, 2003.
[137] D. M. Mark and A. G. Turk, "Landscape categories in Yindjibarndi: ontology, environment, and language," in *Spatial information theory: foundations of geographic information science*, W. Kuhn, M. Worboys, and S. Timpf, Eds., ed Berlin & Heidelberg: Springer-Verlag, 2003, pp. 31-49.
[138] C. Schlenoff, A. Pietromartire, S. Foufou, and S. Balakirsky, "Ontology-Based State Representation for Robot Intention Recognition in Ubiquitous Environments," presented at the UBICOMP 2012 Workshop on "Smart Gadgets Meet Ubiquitous and Social Robots on the Web (UbiRobs)", Pittsburgh, PA, 2012.
[139] C. Schlenoff, A. Pietromartire, Z. Kootbally, S. Balakirsky, and S. Foufou, "Ontology-Based State Representation for Intention Recognition in Cooperative Human-Robot Environments," *Robotics and Autonomous Systems,* vol. 61, pp. 1224-1234, 2013.
[140] J. Carbonera and C. Schlenoff, "Defining Position in a Core Ontology for Robotics," presented at the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Tokyo, Japan, 2013.

[141] J. Tan, F. Duan, Y. Zhang, K. Watandbe, R. Kato, and T. Arai, "Hman-Robot Collaboration in Cellular Manufacturing Design and Development," presented at the International Conference on Information Systems (ICIS), Phoenix, Arizona, 2009.

[142] J. Shi, G. Jimmerson, T. Pearson, and R. Menassa, "Levels of Human and Robot Collaboration for Automotive Manufacturing," presented at the Performance Metrics for Intelligent Systems (PerMIS), College Park, MD, 2012.

[143] S. Balakirsky, Z. Kootbally, C. Schlenoff, T. Kramer, and S. Gupta, "An Industrial Robotic Knowledge Representation for Kit Building Applications," presented at the International Robots and Systems (IROS) Conference Vilamoura, Algarve Portugal, 2012.

[144] C. Schlenoff, E. Prestes, R. Madhavan, P. Goncalves, H. Li, S. Balakirsky*, et al.*, "An IEEE Standard Ontology for Robotics and Automation," presented at the International Conference on Intelligent Robots and Systems (IROS), Vilamoura, Algarve (Portugal), 2012.

[145] F. Harmelen and D. McGuiness. (2004). *OWL Web Ontology Language Overview, W3C web site: http://www.w3.org/TR/2004/REC-owl-features-20040210/*.

[146] S. Balakirsky, Z. Kootbally, T. Kramer, A. Pietromartire, C. Schlenoff, and S. K. Gupta, "Knowledge Driven Robotics for Kitting Applications," *Robotics and Autonomous Systems,* vol. 61, pp. 1205-1214, 2013.

[147] D. Nau, M. Ghallab, and P. Traverso, *Automated Planning: Theory and Practice*. San Francisco, CA: Morgan Kaufmann Publishers Inc., 2004.

[148] C. Schlenoff, S. Foufou, and S. Balakirsky, "An Approach to Ontology-Based Intention Recognition Using State Representations," presented at the Fourth International Conference on Knowledge Engineering and Ontology Development, Barcelona, Spain, 2012.

[149] C. Schlenoff, T. Hong, C. Liu, R. Eastman, and S. Foufou, "A Literature Review of Sensor Ontologies for Manufacturing Applications," presented at the IEEE International Symposium on Robotic and Sensors Environments (ROSE), Washington DC, 2013.

[150] C. Schlenoff, A. Pietromartire, and S. Foufou, "Performance Evaluation of Intention Recognition in Human-Robot Collaborative Environments," *The ITEA Journal,* vol. 34, 2013.

[151] S. Carpin, M. Lewis, J. Wang, S. Balakirsky, and C. Scrapper, "USARSim: a Robot Simulator for Research and Education," presented at the 2007 IEEE International Conference on Robotics and Automation (ICRA), Roma, Italy, 2007.

[152] C. Scrapper, S. Balakirsky, and E. Messina, "MOAST and USARSim: A Combined Framework for the Development and Testing of Autonomous Systems," presented at the 2006 SPIE Defense and Security Symposium, Orlando, FL, 2006.

[153] A. Jacoff, E. Messina, and J. Evans, "A Reference Test Course for Urban Search and Rescue Robots," presented at the 14th International Florida Artificial Intelligence Research Society Conference, Key West, FL, 2001.

[154] S. Balakirsky, T. Kramer, Z. Kootbally, and A. Pietromartire, "Metrics and Test Methods for Industrial Kit Building," Gaithersburg, MD2013.

[155] I. Astrova, N. Korda, and A. Kalja, "Storing OWL ontologies in SQL Relational Databases," *World Academy of Science, Engineering and Technology,* vol. 29, pp. 167-172, 2007.

[156] C. Schlenoff and S. Foufou, "Evaluating State-Based Intention Recognition Algorithms Against Human Performance," in *Robot Intelligence Technology and Applications 2*. vol. XIV, J.-H. Kim, E. Matson, H. Myung, P. Xu, and F. Karray, Eds., ed Denver, Colorado: Springer, 2014, pp. 219-232.

[157] S. Carpin, M. Lewis, J. Wang, S. Balakirsky, and C. Scrapper, "USARSim: a robot simulator for research and education," presented at the IEEE International Conference on Robotics and Automation (ICRA), 2007.

[158] C. Schlenoff and S. Foufou, "Comparing State-Based Intention Recognition to Human Performance," *IEEE Transactions on Human-Machines Systems (submitted),* 2013.

Appendix A: Kitting Workstation Ontology Details

This section provides descriptions of each of the classes in the kitting workstation ontology, in alphabetical order. The names used below are those in the OWL version of the ontology. Each description lists the data members of the class and explains what the class and its data members mean. The terms *inherited*, *optional*, and *multiple* that follow many data member names mean the obvious things. The term *inverse* means that the property (*hasThis_That*) for the data member has an inverse. This occurs whenever the value of a data member is an object rather than a datatype and the data member is not inherited. In this case a *hadByThat_This* property for the data member is also defined in the OWL version of the ontology. If the data member is inherited, some ancestor of the class has the inverse.

BoxVolume is derived from DataThing.

An instance of BoxVolume has the following: MaximumPoint (inverse)

MinimumPoint (inverse).

The MaximumPoint and MinimumPoint are diagonally opposite corner points of a box shaped volume whose edges are aligned with the coordinate system in which the BoxVolume is located. The MinimumPoint has the minimum values of X, Y, and Z. The MaximumPoint has the maximum values of X, Y, and Z.

BoxyShape is derived from InternalShape.

An instance of BoxyShape has the following:

Description (inherited)

GraspPose (inherited, optional)

Length

Width

Height

HasTop.

A BoxyShape is box shaped. The Length is larger of the two dimensions that are not the Height. The Width is smaller of the two dimensions that are not the Height. The coordinate system of a BoxyShape (i.e. the thing that is located and oriented by a Pose) has its origin in the middle of the bottom, its z-axis parallel to the height sides and pointing into the box, and its x-axis parallel to the length sides. If HasTop is true, the top of the box (i.e. the side through which the +Z axis passes) exists and is closed. If HasTop is false, the box has no top.

CylindricalShape is derived from InternalShape.

An instance of CylindricalShape has the following:

Description (inherited)

GraspPose (inherited, optional)

Diameter

Height

HasTop.

The cylinder is a right circular cylinder with a circular base having the given Diameter. The axis is perpendicular to the base. The base is always a surface that is part of the cylinder. The sides of the cylinder stop at the given Height as if cut by a plane perpendicular to the axis. The coordinate system of a CylindricalShape (i.e. the thing that is located and oriented by a pose) has its origin in the middle of the bottom, and its z-axis on the axis of the cylinder. If HasTop is true, the top of the cylinder (i.e. the side through which the +Z axis passes) exists and is closed. If HasTop is false, the cylinder has no top.

DataThing is an abstract type from which more specific types of data thing are derived. That includes all complex data types such as Vector, PhysicalLocation, etc.

EndEffector is derived from NoSkuObject.

An instance of EndEffector has the following:

PrimaryLocation (inherited)

SecondaryLocation (inherited, optional, multiple)

InternalShape (inherited, optional)

ExternalShape (inherited, optional)

Description

Weight

MaximumLoadWeight

HeldObject (optional, inverse).

EndEffector is an abstract type from which more specific types of end effector are derived. An EndEffector is an end effector for a robot. The optional HeldObject is for the object being held by the end effector, if the end effector is holding an object. Every EndEffector is either a GripperEffector or a VacuumEffector. Every EndEffector in a KittingWorkstation is either attached to the end of a robot arm or sitting in an EndEffectorHolder at an EndEffectorChangingStation.

EndEffectorChangingStation is derived from NoSkuObject.

An instance of EndEffectorChangingStation has the following:

PrimaryLocation (inherited)

SecondaryLocation (inherited, optional, multiple)

InternalShape (inherited, optional)

ExternalShape (inherited, optional)

Base (inverse)

EndEffectorHolder (multiple, inverse).

An EndEffectorChangingStation is a place where end effectors are stored and where the robot can change end effectors. The coordinate system of an EndEffectorChangingStation is in the same place as the coordinate system of its Base. The shape of an EndEffectorChangingStation may also be found from the shapes of the Base and the EndEffectorHolders and their relative positions.

EndEffectorHolder is derived from NoSkuObject.

An instance of EndEffectorHolder has the following:

PrimaryLocation (inherited)

SecondaryLocation (inherited, optional, multiple)

InternalShape (inherited, optional)

ExternalShape (inherited, optional)

EndEffector (optional, inverse).

An EndEffectorHolder holds zero or one end effector and is part of an EndEffectorChangingStation.

ExternalShape is derived from ShapeDesign.

An instance of ExternalShape has the following:

Description (inherited)

GraspPose (inherited, optional)

ModelFormatName

ModelFileName

ModelName (optional).

An ExternalShape is a shape defined in an external file. The ModelFormatName is the name of the format of model (for example, 'STEP Advanced Brep' or 'USARSim'). The ModelFileName is the name of the file containing the model and may include a path (for example 'partFiles/STEP/ANC101.stp'). The model file may contain more than one shape model. The ModelName is optional and is the name of a model within the model file. The ModelName is necessary if the model file contains more than one model.

GripperEffector is derived from EndEffector.

An instance of GripperEffector has the following:

PrimaryLocation (inherited)

SecondaryLocation (inherited, optional, multiple)

InternalShape (inherited, optional)

ExternalShape (inherited, optional)

Description (inherited)

Weight (inherited)

MaximumLoadWeight (inherited)

HeldObject (inherited, optional).

A GripperEffector holds an object by gripping it with fingers or claws or by suction.

Human is derived from NoSkuObject.

An instance of Human has the following:

PrimaryLocation (inherited)

SecondaryLocation (inherited, optional, multiple)

InternalShape (inherited, optional)

ExternalShape (inherited, optional).

A Human is a type representing a human being. An internal shape used with a human should be a bounding box or cylinder that encloses the human completely.

InternalShape is derived from ShapeDesign.

An instance of InternalShape has the following:

Description (inherited)

GraspPose (inherited, optional).

InternalShape is an abstract type from which more specific types of shape are derived. Instances of InternalShape in a instance file contain information about the appearance of the shape without referring to another file.

Kit is derived from NoSkuObject.

An instance of Kit has the following:

PrimaryLocation (inherited)

SecondaryLocation (inherited, optional, multiple)

InternalShape (inherited, optional)

ExternalShape (inherited, optional)

Design (inverse)

KitTray (inverse)

Finished

Part (optional, multiple, inverse)

Slot (optional, multiple).

Finished is a boolean indicator of whether the Kit is finished. Part may occur several times (once for each part in the kit). The optional Slots may be used to keep track of whether each place in the kit that should have a part on it does have a part on it. The PartRefAndPose of each Slot should indicate a PartRefAndPose in the design of the kit (different for each slot). The locating point of the Tray in the kit should be (0,0,0), and its X and Z axes should be (1,0,0) and (0,0,1), respectively.

KitDesign is derived from DataThing.

An instance of KitDesign has the following:

KitTraySku (inverse)

PartRefAndPose (multiple, inverse).

The KitTraySku identifies a type of kit tray. The Pose in a PartRefAndPose is the location of the part relative to the coordinate system of the ShapeDesign of the tray.

KittingWorkstation is derived from NoSkuObject.

An instance of KittingWorkstation has the following:

PrimaryLocation (inherited)

SecondaryLocation (inherited, optional, multiple)

InternalShape (inherited, optional)

ExternalShape (inherited, optional)

AngleUnit

LengthUnit

ChangingStation (inverse)

Object (multiple, inverse)

OtherObstacle (optional, multiple, inverse)

Robot (inverse)

KitDesign (multiple, inverse)

Sku (multiple, inverse)

WeightUnit.

All angle, length, and weight values related to the workstation use the units implicitly. The workstation includes one robot and one end effector changing station. There may be many instances of Object in the workstation, including such things as work tables, large boxes with kits, large boxes with empty kit trays, and parts trays. The collection of instances of KitDesign is a library of all kit designs known to the workstation. The collection of instances of Sku is a library of all stock keeping units known to the workstation. The OtherObstacles are obstacles to robot motion of unspecified type. Containers of various sorts enter and leave the workstation. The robot builds kits of parts by executing kitting plans as directed by a kitting plan execution system. The location of each instance of KittingWorkstation should be given relative to itself in order to end the chain of relative locations.

KitTray is derived from SkuObject.

An instance of KitTray has the following:

PrimaryLocation (inherited)

SecondaryLocation (inherited, optional, multiple)

Sku (inherited)

SerialNumber.

The Sku specifies the SKU of the kit tray. A KitTray is designed to hold Parts with various SKUs in known positions.

LargeBoxWithEmptyKitTrays is derived from NoSkuObject.

An instance of LargeBoxWithEmptyKitTrays has the following:

PrimaryLocation (inherited)

SecondaryLocation (inherited, optional, multiple)

InternalShape (inherited, optional)

ExternalShape (inherited, optional)

LargeContainer (inverse)

KitTray (optional, multiple, inverse).

The location point of the LargeContainer should be (0,0,0), its Z axis should be (0,0,1), and its X axis should be (1,0,0). The PrimaryLocation of a KitTray in a LargeBoxWithEmptyKitTrays should be given by a PoseLocationIn or RelativeLocationIn that is relative to the LargeContainer. The KitTrays in a LargeBoxWithEmptyKitTrays are intended to all be of the same SKU, although there is currently no formal requirement for that.

LargeBoxWithKits is derived from NoSkuObject.

An instance of LargeBoxWithKits has the following:

PrimaryLocation (inherited)

SecondaryLocation (inherited, optional, multiple)

InternalShape (inherited, optional)

ExternalShape (inherited, optional)

LargeContainer (inverse)

Kit (optional, multiple, inverse)

KitDesign (inverse)

Capacity.

The coordinate system of a LargeBoxWithKits is in the same place as the coordinate system of its LargeContainer. The PrimaryLocation of the LargeContainer should be relative to the LargeBoxWithKits. The KitDesign is an identifier for a KitDesign. The PrimaryLocation of a Kit in a LargeBoxWithKits should be given by a PoseLocationIn or RelativeLocationIn that is relative to the LargeContainer. The Capacity is an xs:positiveInteger giving the maximum number of kits of the given design that can be held in the box. The Kits in a LargeBoxWithKits are intended to all be of the given design, but there is currently no formal constraint requiring that.

LargeContainer is derived from SkuObject.

An instance of LargeContainer has the following:

PrimaryLocation (inherited)

SecondaryLocation (inherited, optional, multiple)

Sku (inherited)

SerialNumber.

The Sku specifies the SKU of the LargeContainer. A LargeContainer can hold one or more instances of a single type of tray, bin, or kit.

MechanicalComponent is derived from NoSkuObject.

An instance of MechanicalComponent has the following:

PrimaryLocation (inherited)

SecondaryLocation (inherited, optional, multiple)

InternalShape (inherited, optional)

ExternalShape (inherited, optional).

A MechanicalComponent is a component of kitting workstation device such as a robot or an end effector changing station.

NoSkuObject is derived from SolidObject.

An instance of NoSkuObject has the following:

PrimaryLocation (inherited)

SecondaryLocations (inherited, optional, multiple)

InternalShape (optional, inverse)

ExternalShape (optional, inverse).

A NoSkuObject is an abstract type from which more specific types of solid object are derived. The InternalShape and ExternalShape are not required to represent the same shape, but they should not be inconsistent. If a NoSkuObject consists of components it may also get its shape from the shape of the components and their relative positions.

Part is derived from SkuObject.

An instance of Part has the following:

PrimaryLocation (inherited)

SecondaryLocation (inherited, optional, multiple)

Sku (inherited)

SerialNumber.

The Part represents a part. The Sku specifies the SKU for the part.

PartRefAndPose is derived from DataThing.

An instance of PartRefAndPose has the following:

Sku (inverse)

Point (inverse)

ZAxis (inverse)

XAxis (inverse).

The Sku identifies a type of part. The Point specifies the location of the origin of the part in the coordinate system of the tray of the KitDesign to which the PartRefAndPose belongs. The ZAxis and XAxis specify the orientation of the part relative to that coordinate system.

PartsBin is derived from PartsVessel.

An instance of PartsBin has the following:

PrimaryLocation (inherited)

SecondaryLocation (inherited, optional, multiple)

Sku (inherited)

SerialNumber (inherited)

PartSku (inherited)

PartQuantity (inherited)

Part (inherited, optional, multiple).

The Sku specifies the SKU for the PartsBin. A PartsBin holds a number of Parts (PartQuantity) with the same SKU (PartSku) in unknown random positions. Each Part in the tray should be listed explictly and have a RelativeLocationIn with the bin as its RefObject.

PartsVessel is derived from SkuObject.

An instance of PartsVessel has the following:

PrimaryLocation (inherited)

SecondaryLocation (inherited, optional, multiple)

Sku (inherited)

SerialNumber

PartSku (inverse)

PartQuantity

Part (optional, multiple, inverse)

PartsVessel is an abstract type from which more specific types of things that supply parts are derived. The Sku specifies the SKU for the PartsVessel. The shape of a PartsVessel is as specified in its Sku. The PartSku specifies the SKU for the Parts in the PartsVessel. The value of PartQuantity should be the number of instances of Part.

PartsTray is derived from PartsVessel.

An instance of PartsTray has the following:

PrimaryLocation (inherited)

SecondaryLocation (inherited, optional, multiple)

Sku (inherited)

SerialNumber (inherited)

PartSku (inherited)

PartQuantity (inherited)

Part (inherited, optional, multiple).

The Sku specifies the SKU of the PartsTray. A PartsTray holds a number of Parts (PartQuantity) with the same SKU (PartSku) in known positions. Each Part in the tray should be listed explictly and have a PoseLocation with the parts tray as its RefObject.

PhysicalLocation is derived from DataThing.

An instance of PhysicalLocation has the following:

RefObject (inverse)

Timestamp (optional).

PhysicalLocation is an abstract type from which more specific types of physical location are derived. A PhysicalLocation says where a SolidObject is relative to its reference object. Timestamp represents the most recent date and time when the location was updated.

Point is derived from DataThing.

An instance of Point has the following:

X

Y

Z.

X, Y, and Z are the Cartesian coordinates of the Point.

PoseLocation is derived from PhysicalLocation.

An instance of PoseLocation has the following:

RefObject (inherited)

Timestamp (inherited, optional).

Point (inverse)

XAxis (inverse)

ZAxis (inverse)

PositionStandardDeviation (optional)

OrientationStandardDeviation (optional).

PoseLocation is an abstract type from which more specific types of pose location are derived. The Point locates the origin of a coordinate system. The XAxis and ZAxis give the orientation of the coordinate system. The data for the Point, the ZAxis and the XAxis are expressed relative to the coordinate system of the reference object.

The PositionStandardDeviation is based on a normal distribution of actual position about its given value. Thus, for example, the actual position is expected to be within the given PositionStandardDeviation amount 68% of the time and within twice the given amount 95% of the time. The PositionStandardDeviation is measured in the length units being used.

The OrientationStandardDeviation is based on a normal distribution of orientation about its given value. The error is to be measured as the angle of rotation about a single axis needed to rotate a solid object from its stated orientation to its actual orientation. The OrientationStandardDeviation is measured in the angle units being used.

PoseLocationIn is derived from PoseLocation.

An instance of PoseLocationIn has the following:

RefObject (inherited)

Timestamp (inherited, optional).

Point (inherited)

XAxis (inherited)

ZAxis (inherited)

PositionStandardDeviation (inherited, optional)

OrientationStandardDeviation (inherited, optional).

A PoseLocationIn indicates that the object is inside the RefObject. The notion of 'inside' is vague and might be made more precise.

PoseLocationOn is derived from PoseLocation.

An instance of PoseLocationOn has the following:

RefObject (inherited)

Timestamp (inherited, optional).

Point (inherited)

XAxis (inherited)

ZAxis (inherited)

PositionStandardDeviation (inherited, optional)

OrientationStandardDeviation (inherited, optional).

A PoseLocationOn indicates that the Object is on top of the RefObject. The notion of 'on top of' is vague and might be made more precise.

PoseOnlyLocation is derived from PoseLocation.

An instance of PoseOnlyLocation has the following:

RefObject (inherited)

Timestamp (inherited, optional).

Point (inherited)

XAxis (inherited)

ZAxis (inherited)

PositionStandardDeviation (inherited, optional)

OrientationStandardDeviation (inherited, optional).

An object located by a PoseOnlyLocation may or may not be inside or on top of the reference object of the PoseOnlyLocation.

RelativeLocation is derived from PhysicalLocation.

An instance of RelativeLocation has a the following:

RefObject (inherited)

Timestamp (inherited, optional)

Description.

RelativeLocation is an abstract type from which more specific types of relative location are derived. A RelativeLocation indicates that the SolidObject that has the RelativeLocation is on or in the RefObject. The Description may be used to describe the relative positions of the object and its reference object.

RelativeLocationIn is derived from RelativeLocation.

An instance of RelativeLocationIn has the following:

RefObject (inherited)

Timestamp (inherited, optional)

Description (inherited).

A RelativeLocationIn indicates that the SolidObject that has the RelativeLocation is in the RefObject. The notion of 'in' is vague and might be made more precise.

RelativeLocationOn is derived from RelativeLocation.

An instance of RelativeLocationOn has the following:

RefObject (inherited)

Timestamp (inherited, optional)

Description (inherited).

A RelativeLocationOn indicates that the SolidObject that has the RelativeLocation is on top of the the RefObject. The notion of 'on top of' is vague and might be made more precise.

Robot is derived from NoSkuObject.

An instance of Robot has the following:

PrimaryLocation (inherited)

SecondaryLocation (inherited, optional, multiple)

InternalShape (inherited, optional)

ExternalShape (inherited, optional)

Description

EndEffector (optional, inverse)

MaximumLoadWeight

WorkVolume (multiple, inverse).

The Robot ontology given here might be expanded greatly to include, for example, its kinematic description, the values of joint angles, arm lengths of variable length arms, gripper actuation (open, closed, etc.), ranges, velocities, and accelerations of each joint, etc.

ShapeDesign is derived from DataThing.

An instance of ShapeDesign has the following:

Description (optional)

GraspPose (optional, inverse).

ShapeDesign is an abstract type from which more specific types of shape design are derived. Each ShapeDesign has a coordinate system that is expected to be specified explicitly or implicitly. A shape defined using coordinate values has an implicit coordinate system. The GraspPose is relative to the coordinate system of the ShapeDesign. The Point in the pose is the point at which a gripper should make contact with the shape. The ZAxis of the pose may be used to indicate a direction for aligning the ZAxis of the gripper (parallel or antiparallel) and is usually normal to the the object having the shape and pointing away from the object. The GraspPose should not use the optional Timestamp.

SkuObject is derived from SolidObject.

An instance of SkuObject has the following:

PrimaryLocation (inherited)

SecondaryLocations (inherited, optional, multiple)

Sku (inverse)

A SkuObject is an abstract type from which more specific types of solid object are derived. A SkuObject is an instance of a stockkeeping unit. The shape of a SkuObject is specified by its stockkeeping unit.

Slot is derived from DataThing.

An instance of Slot has the following:

PartRefAndPose (inverse)

Part (optional, inverse).

A Slot identifies whether or not a particular PartRefAndPose from the design of a Kit is occupied in an instance of a Kit. The PartRefAndPose identifies a PartRefAndPose from the Design of the Kit. The Part identifies a Part that occupies the PartRefAndPose. The Sku of the PartRefAndPose should be the Sku of the Part, the PartRefAndPose should be in the Kit design, and the Part should be in the Kit. The location described by the Pose of the Part in the Kit may differ from the location described by the Pose in the PartRefAndPose, but will usually be very close to it. If the Part is not used for a slot, that means the slot is empty.

SolidObject is an abstract type from which more specific types of solid thing are derived.

An instance of SolidObject has the following:

PrimaryLocation (inverse)

SecondaryLocations (optional, multiple, inverse)

The secondary locations are required to be logically and mathematically consistent with the value of the PrimaryLocation so that all locations of a SolidObject describe (or are consistent with) a single place in space. No SolidObject except the Workstation may be located with respect to itself, and all chains of primary location must end at the Workstation.

StockKeepingUnit is derived from DataThing.

An instance of StockKeepingUnit has the following:

Description

InternalShape (optional, inverse)

ExternalShape (optional, inverse)

Weight

EndEffector (optional, multiple, inverse).

A StockKeepingUnit is an object type description. SKU is an abbreviation for Stock Keeping Unit. Each EndEffector identifies an instance of EndEffector that can handle the SKU. One or both of InternalShape and ExternalShape must be given. The shapes are not required to represent the same shape, but they should not be inconsistent.

VacuumEffector is derived from EndEffector.

An instance of VacuumEffector has the following:

PrimaryLocation (inherited)

SecondaryLocation (inherited, optional, multiple)

InternalShape (inherited, optional)

ExternalShape (inherited, optional)

Description (inherited)

Weight (inherited)

MaximumLoadWeight (inherited)

CupDiameter

Length.

VacuumEffector is an abstract type from which more specific types of VacuumEffector are derived. A VacuumEffector holds an object by putting a cup or cups against the object and applying a vacuum.

VacuumEffectorMultiCup is derived from VacuumEffector.

An instance of VacuumEffectorMultiCup has the following:

PrimaryLocation (inherited)

SecondaryLocation (inherited, optional, multiple)

InternalShape (inherited, optional)

ExternalShape (inherited, optional)

Description (inherited)

Weight (inherited)

MaximumLoadWeight (inherited)

CupDiameter (inherited)

Length (inherited)

ArrayNumber

ArrayRadius.

The ArrayNumber is the number of cups, which must be at least 2. The cups are arranged in a circular array spaced evenly apart. The center of the wide end of one cup is on the x-axis of the coordinate system of the VacuumEffectorMultiCup. The center of the circular array is at the origin of the coordinate system. The axis of the array circle is the z-axis of the coordinate system, and the length of the VacuumEffector is measured along that axis. The wide ends of the cups lie on the xy plane of the coordinate system. Note that a square array can be represented easily as circular array.

VacuumEffectorSingleCup is derived from VacuumEffector.

An instance of KitTray has the following:

PrimaryLocation (inherited)

SecondaryLocation (inherited, optional, multiple)

InternalShape (inherited, optional)

ExternalShape (inherited, optional)

Description (inherited)

Weight (inherited)

MaximumLoadWeight (inherited)

CupDiameter (inherited)

Length (inherited).

A VacuumEffectorSingleCup has one cup. The center of the wide end of the cup (which is a circle) is at the origin of the coordinate system of the VacuumEffectorSingleCup. The Z axis of the coordinate system is the axis of that circle, and the length of the VacuumEffector is measured along that axis.

Vector is derived from DataThing.

An instance of Vector has the following:

I

J

K.

I, J, and K represent the usual i, j, and k components of a 3D vector.")

WorkTable is derived from NoSkuObject.

An instance of WorkTable has the following:

PrimaryLocation (inherited)

SecondaryLocation (inherited, optional, multiple)

InternalShape (inherited, optional)

ExternalShape (inherited, optional)

ObjectOnTable (optional, multiple, inverse).

Each ObjectOnTable is a SolidObject located with respect to the WorkTable. The reference object of each ObjectOnTable should be the WorkTable. Typically, those objects will be on top of the WorkTable. Typically, the shape of a WorkTable will be a BoxyShape, so that the table has Length, Width, and Height.

Appendix B: Kit Plans for Experiment

Kit 1, Plan 1
1. Red object in the kit tray.
2. Green object in the kit tray.
3. Green object in the kit tray.
4. Blue object in the kit tray.
5. Green object in the kit tray.
6. Blue object in the kit tray.
7. Red object in the kit tray.
8. Red object in the kit tray.
9. Red object in the kit tray.
10. Blue object in the kit tray.

Kit 2, Plan 2
1. Blue object in the kit tray.
2. Red object in the kit tray.
3. Red object in the kit tray.
4. Blue object in the kit tray.
5. Blue object in the kit tray.
6. Green object in the kit tray.
7. Red object in the kit tray.
8. Green object in the kit tray.
9. Green object in the kit tray.
10. Red object in the kit tray.

Kit 1, Plan 3
1. Red object in the kit tray.
2. Blue object in the kit tray.
3. Green object in the kit tray.
4. Green object in the kit tray.
5. Blue object in the kit tray.
6. Red object in the kit tray.
7. Green object in the kit tray.
8. Red object in the kit tray.
9. Red object in the kit tray.
10. Blue object in the kit tray.

Kit 1, Plan 4
1. Red object in the kit tray.
2. Blue object in the kit tray.
3. Red object in the kit tray.
4. Red object in the kit tray.
5. Blue object in the kit tray.
6. Blue object in the kit tray.
7. Green object in the kit tray.
8. Red object in the kit tray.
9. Green object in the kit tray.
10. Green object in the kit tray.

Kit 1, Plan 5
1. Red object in the kit tray.
2. Red object in the kit tray.
3. Blue object in the kit tray.
4. Blue object in the kit tray.
5. Red object in the kit tray.
6. Green object in the kit tray.
7. Green object in the kit tray.
8. Blue object in the kit tray.
9. Red object in the kit tray.
10. Green object in the kit tray.

Kit 2, Plan 1
1. Green object in the kit tray.
2. Red object in the kit tray.
3. Blue object in the kit tray.
4. Green object in the kit tray.
5. Green object in the kit tray.
6. Red object in the kit tray.
7. Green object in the kit tray.
8. Red object in the kit tray.
9. Red object in the kit tray.
10. Blue object in the kit tray.

Kit 2, Plan 2

1. Green object in the kit tray.
2. Green object in the kit tray.
3. Red object in the kit tray.
4. Blue object in the kit tray.
5. Green object in the kit tray.
6. Red object in the kit tray.
7. Green object in the kit tray.
8. Blue object in the kit tray.
9. Red object in the kit tray.
10. Red object in the kit tray.

Kit 2, Plan 3

1. Red object in the kit tray.
2. Red object in the kit tray.
3. Green object in the kit tray.
4. Red object in the kit tray.
5. Blue object in the kit tray.
6. Green object in the kit tray.
7. Red object in the kit tray.
8. Green object in the kit tray.
9. Green object in the kit tray.
10. Blue object in the kit tray.

Kit 2, Plan 4

1. Red object in the kit tray.
2. Blue object in the kit tray.
3. Green object in the kit tray.
4. Red object in the kit tray.
5. Red object in the kit tray.
6. Green object in the kit tray.
7. Blue object in the kit tray.
8. Green object in the kit tray.
9. Green object in the kit tray.
10. Red object in the kit tray.

Kit 2, Plan 5
1. Green object in the kit tray.
2. Green object in the kit tray.
3. Red object in the kit tray.
4. Red object in the kit tray.
5. Green object in the kit tray.
6. Blue object in the kit tray.
7. Green object in the kit tray.
8. Red object in the kit tray.
9. Blue object in the kit tray.
10. Red object in the kit tray.

Kit 3, Plan 1
1. Green object in the kit tray.
2. Red object in the kit tray.
3. Blue object in the kit tray.
4. Blue object in the kit tray.
5. Green object in the kit tray.
6. Blue object in the kit tray.
7. Blue object in the kit tray.
8. Red object in the kit tray.
9. Green object in the kit tray.
10. Blue object in the kit tray.

Kit 3, Plan 2
1. Blue object in the kit tray.
2. Blue object in the kit tray.
3. Red object in the kit tray.
4. Blue object in the kit tray.
5. Blue object in the kit tray.
6. Red object in the kit tray.
7. Green object in the kit tray.
8. Green object in the kit tray.
9. Green object in the kit tray.
10. Blue object in the kit tray.

Kit 3, Plan 3
1. Green object in the kit tray.
2. Red object in the kit tray.
3. Blue object in the kit tray.
4. Blue object in the kit tray.
5. Blue object in the kit tray.
6. Green object in the kit tray.
7. Blue object in the kit tray.
8. Red object in the kit tray.
9. Blue object in the kit tray.
10. Green object in the kit tray.

Kit 3, Plan 4
1. Blue object in the kit tray.
2. Green object in the kit tray.
3. Blue object in the kit tray.
4. Green object in the kit tray.
5. Blue object in the kit tray.
6. Blue object in the kit tray.
7. Blue object in the kit tray.
8. Red object in the kit tray.
9. Red object in the kit tray.
10. Green object in the kit tray.

Kit 3, Plan 5
1. Blue object in the kit tray.
2. Red object in the kit tray.
3. Blue object in the kit tray.
4. Green object in the kit tray.
5. Green object in the kit tray.
6. Blue object in the kit tray.
7. Red object in the kit tray.
8. Blue object in the kit tray.
9. Blue object in the kit tray.
10. Green object in the kit tray.

Kit 4, Plan 1
1. Blue object in the kit tray.
2. Red object in the kit tray.
3. Green object in the kit tray.
4. Red object in the kit tray.
5. Green object in the kit tray.
6. Green object in the kit tray.
7. Red object in the kit tray.
8. Orange object in the kit tray.
9. Blue object in the kit tray.
10. Red object in the kit tray.

Kit 4, Plan 2
1. Green object in the kit tray.
2. Red object in the kit tray.
3. Red object in the kit tray.
4. Blue object in the kit tray.
5. Orange object in the kit tray.
6. Green object in the kit tray.
7. Blue object in the kit tray.
8. Red object in the kit tray.
9. Green object in the kit tray.
10. Red object in the kit tray.

Kit 4, Plan 3
1. Green object in the kit tray.
2. Green object in the kit tray.
3. Blue object in the kit tray.
4. Red object in the kit tray.
5. Green object in the kit tray.
6. Blue object in the kit tray.
7. Orange object in the kit tray.
8. Red object in the kit tray.
9. Red object in the kit tray.
10. Red object in the kit tray.

Kit 4, Plan 4
1. Green object in the kit tray.
1. Red object in the kit tray.
3. Red object in the kit tray.
4. Green object in the kit tray.
5. Red object in the kit tray.
6. Blue object in the kit tray.
7. Orange object in the kit tray.
8. Red object in the kit tray.
9. Green object in the kit tray.
10. Blue object in the kit tray.

Kit 4, Plan 5
1. Blue object in the kit tray.
2. Green object in the kit tray.
3. Red object in the kit tray.
4. Blue object in the kit tray.
5. Orange object in the kit tray.
6. Green object in the kit tray.
7. Green object in the kit tray.
8. Red object in the kit tray.
9. Red object in the kit tray.
10. Red object in the kit tray.

Kit 5, Plan 1
1. Red object in the kit tray.
2. Green object in the kit tray.
3. Blue object in the kit tray.
4. Yellow object in the kit tray.
5. Blue object in the kit tray.
6. Green object in the kit tray.
7. Red object in the kit tray.
8. Green object in the kit tray.
9. Orange object in the kit tray.
10. Blue object in the kit tray.

Kit 5, Plan 2
1. Green object in the kit tray.
2. Blue object in the kit tray.
3. Yellow object in the kit tray.
4. Red object in the kit tray.
5. Green object in the kit tray.
6. Blue object in the kit tray.
7. Red object in the kit tray.
8. Blue object in the kit tray.
9. Green object in the kit tray.
10. Orange object in the kit tray.

Kit 5, Plan 3
1. Red object in the kit tray.
2. Green object in the kit tray.
3. Green object in the kit tray.
4. Green object in the kit tray.
5. Blue object in the kit tray.
6. Yellow object in the kit tray.
7. Blue object in the kit tray.
8. Red object in the kit tray.
9. Orange object in the kit tray.
10. Blue object in the kit tray.

Kit 5, Plan 4
1. Blue object in the kit tray.
2. Blue object in the kit tray.
3. Green object in the kit tray.
4. Green object in the kit tray.
5. Red object in the kit tray.
6. Green object in the kit tray.
7. Red object in the kit tray.
8. Orange object in the kit tray.
9. Yellow object in the kit tray.
10. Blue object in the kit tray.

Kit 5, Plan 5

1. Blue object in the kit tray.
2. Orange object in the kit tray.
3. Green object in the kit tray.
4. Blue object in the kit tray.
5. Red object in the kit tray.
6. Green object in the kit tray.
7. Yellow object in the kit tray.
8. Green object in the kit tray.
9. Blue object in the kit tray.
10. Red object in the kit tray.