

Doctoral Dissertations

Dissertations and Theses

July 2020

Integrating Recognition and Decision Making to Close the Interaction Loop for Autonomous Systems

Richard Freedman
University of Massachusetts Amherst

Follow this and additional works at: https://scholarworks.umass.edu/dissertations_2



Part of the Artificial Intelligence and Robotics Commons

Recommended Citation

Freedman, Richard, "Integrating Recognition and Decision Making to Close the Interaction Loop for Autonomous Systems" (2020). *Doctoral Dissertations*. 1924.
<https://doi.org/10.7275/jvms-nd30> https://scholarworks.umass.edu/dissertations_2/1924

This Open Access Dissertation is brought to you for free and open access by the Dissertations and Theses at ScholarWorks@UMass Amherst. It has been accepted for inclusion in Doctoral Dissertations by an authorized administrator of ScholarWorks@UMass Amherst. For more information, please contact scholarworks@library.umass.edu.

**INTEGRATING RECOGNITION AND DECISION
MAKING TO CLOSE THE INTERACTION LOOP FOR
AUTONOMOUS SYSTEMS**

A Dissertation Presented
by
RICHARD GABRIEL FREEDMAN

Submitted to the Graduate School of the
University of Massachusetts Amherst in partial fulfillment
of the requirements for the degree of

DOCTOR OF PHILOSOPHY

May 2020

College of Information and Computer Sciences

© Copyright by Richard Gabriel Freedman 2020
All Rights Reserved

**INTEGRATING RECOGNITION AND DECISION
MAKING TO CLOSE THE INTERACTION LOOP FOR
AUTONOMOUS SYSTEMS**

A Dissertation Presented
by
RICHARD GABRIEL FREEDMAN

Approved as to style and content by:

Shlomo Zilberstein, Chair

Roderic A. Grupen, Member

Benjamin M. Marlin, Member

Heather Pon-Barry, Outside Member

Ya'akov (Kobi) Gal, Member

James Allan, Chair of the Faculty
College of Information and Computer Sciences

DEDICATION

In loving memory of Chester Oxenhorn, Elijah Freedman, and Lilian Freedman.

*This dissertation is also dedicated to Doris Oxenhorn. Without their love, support,
and sacrifices, as well as the foundation they prepared, it would not have been
possible for the grandchildren (my generation) to discover and pursue our dreams.*

ACKNOWLEDGMENTS

Our lives are most often defined by small, rarely planned, moments and events—it is difficult to imagine where this work and I would be without mine and the roles that others played in them. It takes a village to raise a child, and my nearly eight-year journey maturing into a research scientist is no exception to this adage. Many people played a fundamental role in my education, career, and/or life during this time. Ranging from family and friends, to mentors and mentees, to colleagues, and even to people running everyday services that made it possible for me to survive and focus on what I had to do; they all deserve some degree of acknowledgment for their contributions to my growth into the person I am at the time of writing this dissertation.

Dissertation-Related

The first person who I must thank clearly played one of the largest roles in this entire rite of passage: my advisor, *Shlomo Zilberstein*. Since Shlomo first invited me to join the Resource Bounded Reasoning Lab under his mentorship, he has continuously looked out for me, provided deep guidance and feedback, trained me in academic discipline, allowed me the freedom to explore my passions, and helped me to make the most of various opportunities upon which I happened to stumble. It is difficult to imagine a more supportive advisor with as much patience and kindness, and I can only hope to provide similar warmth and insights to others in the future. If Shlomo did not introduce me to the area of automated planning research (I honestly was not aware of it when we first met), then the specific chain of events that

eventually led to studying decision making for interaction probably would not have happened.

No graduate student can talk about people who played a role during this time without also acknowledging their labmates. Working in the same space for years, doing research together, and providing feedback on life and ideas, it is quite obvious why people describe their lab as their ‘academic family’. My academic family in the Resource Bounded Reasoning Lab thus deserves a special thank you: ***Michele Roberts*** (the power and abilities of the grants and contracts coordinator should never be underestimated), ***Akshat Kumar, Xiaojian Wu, Yi Lu, Luis Pineda, Kyle Wray, Sandhya Saisubramanian, Justin Svegliato, Connor Basich, Shuwa Miura, John Peterson, and Abhinav Bhatia***—no matter how long or short our overlap was in the lab, I cannot completely express my appreciation for our time together and your support.

In my case, the Resource Bounded Reasoning Lab was a superset of my labmates. First, I had the opportunity to mentor three amazing undergraduate (and some later became masters) students under Shlomo’s blessing: ***Justin Purcell, Roman Ganchin, and Yi Ren Fung***. It was not a one-way experience; I learned a lot from interacting with them throughout their projects. The research with Justin appears in spirit through the discussion of integrating planning and recognition for videogame applications (such as Sections 1.3, 2.5, and 3.4), and Roman and Yi both played important roles in implementing multiple goal heuristic search based on just a journal paper’s description (see Section 6.5.1) while preparing for other research projects that have either been delayed or canceled. Furthermore, some lab members who graduated before I became a graduate student have taken the time to give me guidance and continue to stay in correspondence. For taking the time to do this, thank you ***Siddharth Srivastava, William Yeoh, and Christopher Amato***. I would also like to thank ***Felix Voelker*** and ***Haochong Zhang***, who both visited

our lab for a few months and might as well be members of my academic family with how many insights and experiences we shared together.

As a visitor to another lab myself, I absolutely must extend my thanks to my other academic family in the Fukunaga Lab at The University of Tokyo: **Masataro Asai**, **Yuu Jinnai**, **Ryoji Tanabe**, and **Shuwa Miura** (the same Shuwa Miura as above). We not only discussed research, but all of them kindly took time to help me better fit in and understand Japanese culture. Although **Claire Cheong** was not yet a member of the lab, we happened to meet on the campus and frequently discussed artificial intelligence research—her constant passion and infectious optimism deserve a thank you as well.

I owe my deepest gratitude to **Alex Fukunaga** himself for serving as my host professor; he kindly invited me to apply for the National Science Foundation’s East Asia and Pacific Summer Institutes (NSF EAPSI) grant, arranged all my accommodations, provided guidance as an additional advisor, and exposed me to many new insights (both regarding research and living in Japan). Despite Fukunaga-sensei’s modesty, I still give him the fundamental credit behind sparking my eventual realization of the notion of necessities, which inspired the research on responsive planning in Chapter 5. He introduced me to the recognition as planning class of algorithms and then asked me if it could work with the same planner for interaction. Without this pivotal discovery that the **NSF** generously sponsored, I cannot be certain that I would have conceived the later research in this dissertation because it all relies so heavily on the necessities work.

Besides my two official lab memberships, **Roderic Grupen** kindly treated me as a member of the Laboratory for Perceptual Robotics. As such, he provided further guidance and advising, encouraging additional research ideas and directions that I still wish there was time to have pursued while in graduate school. His students were just as welcoming, acting as a surrogate academic family: **Dirk Ruiken**, **Hee-**

Tae Jung, Michael Lanigan, Takeshi Takahashi, Tiffany Liu, Li Yang Ku, Jay Ming Wong, Mitchell Hebert, Khoshrav Doctor, Scott Jordan, James Kostas, and Hia Ghosh. Hee-Tae deserves an additional thank you for both collaborating with me on my early research in Chapter 4 and introducing me to the research area of human-robot interaction. Becoming aware of this field was a monumental moment in identifying my personal research interests.

As a part of finding this identity, I became a member of the *artificial intelligence for human-robot interaction (AI-HRI)* research community. The AI-HRI researchers are all amazing, and I greatly look forward to our yearly gatherings to introduce new findings, establish collaborations, and discuss future challenges for the field. They are the reasons that I understand and appreciate just how important it is to consider people themselves when designing artificial intelligence algorithms intended to work with humans (see Chapters 7 and 8). In particular, I wish to thank my fellow AI-HRI colleagues who have worked with me to organize the annual gathering at the AAAI Fall Symposium Series in 2016, 2018, and 2019: *Ross Mead, Dan Grollman, Bradley Hayes, Tom Williams, Nick DePalma, Tiago Ribeiro, Patrícia Alves-Oliveira, Gordon Briggs, Frank Broz, Katrin Lohan, Emmanuel Senft, Kalesha Bullard, Luca Iocchi, Justin Hart, Jivko Sinapov, and Elin Topp*. I have learned so much from these individuals, both working together to arrange the event as well as through discussions about our AI-HRI research.

Aside from the AI-HRI research community, my research colleagues within the *Plan, Activity, and Intent Recognition (PAIR), International Conference on Automated Planning and Scheduling (ICAPS)*, and the *Association for the Advancement of Artificial Intelligence (AAAI)* communities deserve a deep thank you for all their patience, discussions, feedback, and additional mentorship throughout the past eight years. I especially want to express my appreciation for *Steven Levine*'s support and frequent discussions as we developed our closed-loop

interactive intelligent systems—Section 5.3.1 mentions some of these. Additional discussions with *Sarah Keren*, *Reuth Mirsky*, *Karthik Talamadupula*, *Matthew Gombolay*, *Joseph Kim*, *Vaibhav Unhelkar*, *Ankit Shah*, *Claudia Pérez D'Arpino*, *Ramya Ramakrishnan*, *Tariq Iqbal*, *Tathagata Chakraborti*, *Sarath Sreedharan*, *Sailik Sengupta*, *Lydia Manikonda*, *Ping Hou*, *Christabel Wayllace*, *Maayan Shvo*, *Alberto Camacho*, *Christian Muise*, *Shirin Sohrabi*, and *Michael Katz* throughout the years on my research are worth acknowledgment as well.

Some members of these research communities have even taken time to provide additional advice and mentorship not just regarding my research, but also with respect to preparations for my academic future. Given how much these people already have to deal with regularly at their own jobs and institutions, it is vital that they receive a large thank you: *David Aha*, *Jeremy Frank*, *Christopher Geib*, *Laura Hiatt*, *Mak Roberts*, *Julie Shah*, *Sheila McIlraith*, *Matthias Scheutz*, *George Konidaris*, *Cynthia Matuszek*, and *J Benton*. Their advice helped me find new opportunities and perspectives, informed me enough to make some difficult decisions, and exposed me to some of the lesser-known sides of academia.

As additional mentors, I absolutely must extend my greatest appreciation to my dissertation committee: *Shlomo Zilberstein*, *Roderic Grupen*, *Benjamin Marlin*, *Heather Pon-Barry*, and *Ya'akov (Kobi) Gal*. Alongside the aforementioned advising roles, everyone took the time to meet with me throughout my research and provide critical feedback that greatly improved the quality of this dissertation. Though not on my dissertation committee, *Hanna Wallach* was similarly as attentive and insightful when she served on my masters thesis committee alongside Shlomo and Roderic. These people deemed my dissertation research worthy enough to complete the rite of passage into becoming a research scientist, and that is a privilege for which I am eternally grateful.

During the final stretch through which my committee patiently advised, I began working at Smart Information Flow Technologies (SIFT), LLC. Over these past one-and-a-half years, my co-workers provided further ideas, insights, and feedback that also greatly improved the quality of the dissertation research. So besides thanking ***Harry Funk*** and ***Christopher Miller*** for bringing such an amazing research and development consulting small business into existence, I want to provide a large thank you to all my SIFT co-workers for their support, fresh perspectives, and creating such a warm and friendly environment while I simultaneously worked on the completion of this dissertation. I am so happy to be part of this [yet another] academic family: ***Christopher Geib, Robert Goldman, Ugur Kuter, David Musliner, Scott Friedman, Jeremy Gottlieb, Dan Thomsen, Peter Keller, Jack Ladwig, Phillip Walker, David LaVergne, Joseph Mueller, Michael Pelican, Jeffrey Rye, Steven Johnston, Eric Engstrom, Ruta Wheelock, Nathan Ringo, Jesse Roland, Saumik Narayanan, Ian Kariniemi, Joan Zheng, Matthew DeHaven, Mark Valovage, Daniel Bryce, Mark Burstein, David McDonald, Laurel Bobrow, Josh Hamell, Sonja Schmer-Galunder, Pavan Kantharaju, Pooja Patel, Puja Trivedi, Phoebe Arthur Goldman, Michelle Ausman, Christopher Tran, Anthony Chen, Jack Maloney, Linda Holje, Kara Stegeman, and Cam Magnus.***

In addition to everyone who had some relation to me through my graduate studies and this work, there are also people in the present and future who are and will be connected through this dissertation. We game designers often thank the player because as important as others' support is during the production process, it is those who play the game afterwards that give that production effort purpose and recognition. Art is designed to express a message to others, and the greatest acknowledgement we can receive is that others experienced the art and contemplated its message. Like-

wise, taking the time to read and think about any extent of this dissertation deserves appreciation—so thank you, dear *reader*, for taking interest in this work.

Family and Friends

While Shlomo was the catalyst for my research direction and prepared me for life as an academic, my family was my strongest support group and prepared me for everything else in life. My parents, ***Barry and Daniella Freedman***, were, and are, always a single phone call away to answer the most mundane questions, provide feedback and advice, and have casual conversations whenever I was lonely or simply wanted to speak with someone. My sister, ***Riana Freedman***, has been working hard on her own endeavors while still taking the time to check in and share her own life's progress. My grandparents, ***Chester and Doris Oxenhorn*** and ***Elijah and Lilian Freedman***, have been some of my strongest supporters since I first realized that my dream job involved artificial intelligence research and teaching. All the other members of my family, from uncles and aunts to cousins at various degrees, also checked in and shared their support with my immediate family while I foolishly let myself get too carried away with work to reach out to them as frequently. My family's love and care for me has always kept me going.

In addition to my family's support, I am very grateful for my friends' unwavering support. Even with our separation across the East Coast since graduating from Wake Forest University together, ***Marvin Jones*** and I still keep in touch to the point that I have probably spoken with him more than any person outside my family. No matter the topic or reason, he has always been available to chat—sharing philosophies on teaching, going over research, talking about videogames, making jokes about mathematics and/or computer science, sending photographs and videos, getting feedback on artwork, etc. As we have both been enduring the graduate school journey with our

own hiccups along the way, it has been comforting to have someone who understands the trials and tribulations as we lean on each other to make it through.

More local to the Amherst area, I am very fortunate to have met new lifelong friends besides Marvin. From research interests to hobbies, ***Spencer Lane*** and I have a strong overlap such that there is always something to talk about. It is fun to have a random conversation or play a game as a diversion from the graduate school work. It has also been wonderful bouncing research ideas back-and-forth related to the areas of recognition and/or planning, and he knows robotics! Likewise, ***Rachel Lavery*** is always around the computer science building busily taking care of grants and contracts paperwork for someone, but somehow finds time to chat about lots of stuff as well. We have spent dozens of hours sharing music, talking about Japanese pop culture and media, getting overexcited about our hobbies, chatting about life in the area, and discussing our own challenges and accomplishments. Whenever I needed something while in Amherst, I knew that I could trust Spencer or Rachel for help.

On top of so many close friends, I have been further fortunate to get to know, learn from, and interact with many others through group activities and communities. Playing board games with ***Olga Gouliko***, ***Christopher Amey***, and their colleagues in the Physics Department was always a fantastic evening together. It more-or-less turned into a ritual to join ***Ira Male***, ***Trisha Zintel***, ***Korin Albert***, ***Liv Dedon***, and their colleagues in both the Molecular and Cellular Biology and Food Science Departments for weekly trivia (where the goal was to *not have the lowest score*). I further had the pleasure of so many wonderful interactions with various friends over meals when we were working late in our labs, including ***Weize Kong***, ***David Posner***, ***Jonaz Moreno***, ***Craig Greenberg***, ***Abram Handler***, ***Hananel Hazan***, and ***Jie Xiong***. When they were not available, ***James Boynton*** was often working at the on-campus dining facilities and open to deep discussions on a wide variety of

engaging topics. It is impossible to name everyone, but such interactions frequently provided inspiration for new ideas.

Additional Academic Acknowledgments

While some scientists view academia as a venue for more research freedom, I further see it as an institution for sharing experiences of the trade with others. The teaching and outreach aspects are just as important to me as the scientific research, which is likely through inspiration from so many fantastic instructors and mentors prior to attending the University of Massachusetts Amherst. Without them, I might not have received the initial experiences or exposures that helped me realize that this was the correct career path for me. Likewise, since pursuing this path, my amazing mentors and peers who share these passions have ensured that I found out about appropriate opportunities to improve my teaching and communication skills.

As far back as high school, I was fortunate to be a part of the Wake Forest Medical Center's *Center of Excellence for Research, Teaching, and Learning (CERTL)* summer program for two summers. During the first summer, I received exposure to scientific practices such as experimental design and learned about being a researcher while working in **Jill Harp**'s and **Morris Clarke**'s biochemistry labs. Over the second summer, I directly worked under **Edward Ip** in biostatistics where I learned about data analysis and visualization. This was before I knew anything about computer science, and he kindly invited me to work with him the following summer as an intern where he and **Anthony Pecorella** taught me the basics about programming in order to implement an interactive survey for data collection. Over the summer between high school and college, continuing into the following year, I interned under **Carl Langefeld** (also in biostatistics) to implement some additional plotting software. He made sure that I was exposed to the research behind the collected data as well. **Richard (Tommy) Guy** was a masters student at Wake Forest

University at the time who was also an intern there, and he served as a role model throughout my undergraduate degree by helping me prepare for graduate school as I delved deeper and deeper into the world of research.

Over my four years at Wake Forest University as an undergraduate student, the Mathematics and Computer Science Departments' faculty and staff truly made it possible for me to realize my passions for research through all their support and mentorship. Beginning with ***Stephen Robinson***'s encouragement to further investigate a simple trend that I noticed among some numbers he wrote on the whiteboard to introduce the Collatz Conjecture as a break from multivariable calculus, I became consumed with the world of research and began working on various other problems that the professors were willing to put in front of me. Even if I was not working on their research, many of the professors kindly met with me for hours to discuss my ideas, their research problems, how to apply to graduate school, and attending conferences to present my research findings. I hope to exercise their levels of support and passion to help future students identify their interests and make sure that they get every chance to pursue those interests. Many thank yous to Stephen, ***Errin Fulp, Sarah Raynor, Jeremy Rouse, Kenneth Berenhaut, Miaohua Jiang, Todd Torgersen, William Turkett, Frederick Chen, Daniel Cañas, David John, Stan Thomas, Pete Santiago, and Sriraam Natarajan*** for doing all that for me.

Sriraam invited me to work in his brand-new research laboratory where I did real artificial intelligence research. Along with my labmates ***Phillip Odom, Shuo Yang, Adam Edwards, Baidya Saha, and Jose Picado***, it was the first experience I had with an academic family. Sriraam taught me about artificial intelligence and machine learning, and he did so much to prepare me for the real graduate school experience. In hindsight, I cannot thank him enough when I realize just how many times I was prepared for something from his mentorship.

Alongside my exposure to research, ***Samuel Cho*** and ***Paúl Pauca*** introduced me to computer science outreach. As someone who did not know about computer science and programming until the end of high school, I found this mission too important and continued its pursuit at the University of Massachusetts Amherst. To those with whom I have worked at these outreach events, thank you so much for taking the time and care to ensure that people have the opportunity to know about this field and realize whether it is their cup of tea. I especially want to thank ***Benjamin Marlin***, ***Ivan Lee***, and ***Phillipa Gil*** for organizing the majority of these programs.

I must also thank the people I met who are doing computer science outreach beyond the scope of a program at the University of Massachusetts Amherst: ***William (Rick) Adriion***, ***Renee Fall***, ***Sarah Dunton***, and ***Deanna Roux***—they worked with and ran statewide and nationwide programs. I have been very fortunate to attend some of their local events and further meet with them for advice about organizing outreach programs on my own after graduate school. I did not consider such a scope until I first met and began to work with ***Melissa Zeitz***, brainstorming some projects to introduce programming to her elementary school students. If anyone ever has the opportunity to shadow a K-12 teacher for a day, I highly recommend it—visiting Melissa’s school and interacting with the students throughout the day was one of my most eye-opening experiences! ***Akanksha Atrey*** has since been working with Melissa as well as on her own outreach endeavors; thank you for both carrying the torch and your enthusiasm for educating others.

Beyond computer science, there are outreach efforts for many other areas of study. No matter what topic you are spreading with your passion, time, and resources: thank you. At the University of Massachusetts Amherst, I had the honor of co-organizing the first Outreach and Public Engagement Summit, which served as a gathering for graduate students all over campus to present their outreach efforts, share ideas, and find programs with which they could volunteer. There were many amazing efforts and

people at the event, and this was just within a small college town. I give my sincerest thanks to my co-organizers, *Evan Kuras*, *Laura Lanier*, *Laura Hancock*, and *Elena Sesma*, as well as to *Heidi Bauer-Clapp* in the Office of Professional Development for her guidance, support, and resources as we worked on this cross-disciplinary endeavor.

Both Heidi and *Shana Passonno* deserve my additional thanks for all their efforts to help us graduate students learn about making the most of our graduate school experience; ranging from helping us prepare funding proposals, to identifying alternative careers, and even to improving our scientific communication skills. Their services and dedication presented me with far too many fantastic opportunities and learning experiences to count, and they both helped me realize just how diverse life after graduate school could be compared to the often-mentioned industry vs. academic research paths.

In addition to their revelations on the diversity of jobs for people with advanced degrees, one person expanded my horizons over what an individual can do within the world: *Joshua A.C. Newman*. I was blessed to have met Joshua, for whom I originally served as a teaching assistant due to a very particular sequence of events (that involved recommendations from Shlomo). Joshua truly cares about the students' education and finds approaches to help them learn things in ways that work for and matter to them. He has similarly taken me under his wing and been mentoring me in the ways of becoming the kind of teacher who enables students to see education as an opportunity rather than an obstacle. Furthermore, he helped me realize that one's degrees are simply evidence of your abilities to others when there is no other evidence of experience—this means they are qualifications rather than restrictions on what we can do in life. As someone who was formerly brainwashed by the societal norm that our education dictated our job and future, I had assumed that my hobbies as a game designer and artist were nothing more than a form of passing time and not worthy of

professional standards. However, via experience through my dedication to the craft, Joshua showed me that I can be serious and taken seriously in these directions as much as I am with respect to researching artificial intelligence for interaction. As a teacher, game designer, science fiction writer, artist (over many mediums), and overall amazing person, Joshua has become my role model for working hard to be all the things that one cares to be—I will be sure to follow in his footsteps, pursuing my roles as a scientist, teacher, game designer, artist, and good human being.

ABSTRACT

INTEGRATING RECOGNITION AND DECISION MAKING TO CLOSE THE INTERACTION LOOP FOR AUTONOMOUS SYSTEMS

MAY 2020

RICHARD GABRIEL FREEDMAN

B.S., WAKE FOREST UNIVERSITY

M.S., UNIVERSITY OF MASSACHUSETTS AMHERST

Ph.D., UNIVERSITY OF MASSACHUSETTS AMHERST

Directed by: Professor Shlomo Zilberstein

Intelligent systems are becoming increasingly ubiquitous in daily life. Mobile devices are providing machine-generated support to users, robots are “coming out of their cages” in manufacturing to interact with co-workers, and cars with various degrees of self-driving capabilities operate amongst pedestrians and the driver. However, these interactive intelligent systems’ effectiveness depends on their understanding and recognition of human activities and goals, as well as their responses to people in a timely manner. The average person does not follow instructions step-by-step or act in a formulaic manner, but instead varies the order of actions and timing when performing a given task. People explore their surroundings, make mistakes, and may interrupt an activity to handle more urgent matters. The decisions that an autonomous intelli-

gent system makes should account for such noise and variance regardless of the form of interaction, which includes adapting action choices and possibly its own goals.

While most people take these aspects of interaction for granted, they are complex and involve many specific tasks that have primarily been studied independently within artificial intelligence. This results in open-loop interactive experiences where the user must perform a fixed input command or the intelligent system performs a hard-coded output response—one of the components of the interaction cannot adapt with respect to the other for longer-term back-and-forth interactions. This dissertation explores how developments in plan recognition, activity recognition, intent recognition, and autonomous planning can work together to develop more adaptive interactive experiences between autonomous intelligent systems and the people around them. In particular, we consider a unifying perspective of recognition algorithms that provides sufficient information to dynamically produce short-term automated planning problems, and we present ways to run these algorithms faster for the real-time needs of interaction. This exploration leads to the introduction of the Planning and Recognition Together Close the Interaction Loop (PRETCIL) framework that serves as a first step towards identifying how we can address the problem of closing the interaction loop, in addition to new questions that need to be considered.

PREFACE

As a scientist, I study what we do not understand about the world. As a teacher, I explain to others what we believe we understand about the world. As an artist, I take what we know we understand about the world and challenge it.

— Richard (Rick) G. Freedman

Whether with the world, other people, or a medium of expression, interaction is involved in everything I do. It does not just apply there; interaction is in the world all around us, from the simple things like opening doors to the complex phenomena of having complete conversations with people across a room using just gestures and signals. The goal of this work is to begin exploring this amazing feat as it applies to the complex interactions between intelligent agents: understanding others enough to make informed decisions about what to do next, and using expectations of consequences from our decisions to identify when things are going as planned or need to be revised.

TABLE OF CONTENTS

	Page
ACKNOWLEDGMENTS	v
ABSTRACT	xviii
PREFACE	xx
LIST OF TABLES	xxvi
LIST OF FIGURES	xxviii
 CHAPTER	
1. MOTIVATION: WHY INTERACTIVE ARTIFICIAL INTELLIGENCE MATTERS	1
1.1 Overview of Past Approaches	3
1.2 Complexity of Interaction	6
1.2.1 Increasing Complexity for Automated Planning	7
1.2.2 Increasing Complexity for Game-Theoretic Situations	7
1.2.3 Increasing Complexity for Interaction	8
1.2.4 The Potential to Reduce Complexity through Recognition	12
1.3 Videogames: A Motivating Interactive Domain	15
1.4 Overview of Introduced Framework	17
1.5 Dissertation Contributions	18
1.5.1 Primary Contributions	18
1.5.2 Additional Contributions	19
2. AUTOMATED PLANNING	21
2.1 Logic-Based Factored Representations	22
2.1.1 Knowledge Engineering: Expressive Factored Representations	26

2.1.2	Generalizing Factored Representations with Abstractions	31
2.2	The Planning Graph	34
2.3	Hierarchical Task Networks	37
2.3.1	Using HTNs with State-Space Search	39
2.4	Allowing Uncertainty in Automated Planning	40
2.5	Applications of Automated Planning in Videogames	44
2.6	Concluding Remarks: Evolution of Automated Planning	45
3.	PLAN, ACTIVITY, AND INTENT RECOGNITION	47
3.1	Plan Recognition	49
3.1.1	Abduction	50
3.1.2	Parsing	59
3.1.3	Recognition as Planning	64
3.2	Activity Recognition	68
3.2.1	Hidden Markov Models and Their Generalizations	69
3.2.2	Latent Semantic Analysis	72
3.2.3	Activity Recognition in the Wild	75
3.2.4	Activity Recognition for Assistive Technologies	77
3.3	Intent and Goal Recognition	78
3.3.1	Prediction of Upcoming Actions and Underlying Motives	80
3.3.2	Recognition as Planning	82
3.3.3	Goal Recognition Design	84
3.4	Applications of Recognition in Videogames	86
3.5	Concluding Remarks: A Unification of PAIR	86
4.	INTEGRATION OF ACTIVITY RECOGNITION AND APPROXIMATE PLAN RECOGNITION	91
4.1	Topic Model Approach and Analogy to Recognition	92
4.2	Domain-Independent Representations for Signal Data	95
4.2.1	Representing RGB-D Data with Joint-Angles	100
4.2.2	Interpretability of Representations	101
4.3	Using Temporal and Spatial Relations	109
4.3.1	Capturing Temporal Relations with the Composite Model	111

4.3.2	Capturing Object Relations with Parameterized LDA	115
4.3.3	The Parameterized Composite Model	117
4.3.4	Empirical Evaluation of Alternative Topic Models	118
4.3.4.1	Runtime Performance	120
4.3.4.2	Recognition Performance	121
4.3.4.3	Topic and State Investigation	122
4.4	Concluding Remarks: Recognizing Activities and Plans	126
5.	INTEGRATION OF RECOGNITION AND AUTOMATED PLANNING	128
5.1	Foresight Using Survival Analysis	130
5.2	Finding Intermediate Interaction Tasks with Necessities	133
5.3	Helpfulness as a Measure of Impact on the Interaction	137
5.3.1	Challenges with Computing Helpfulness	139
5.3.1.1	Helpfulness without Bias from Problem Instances	140
5.3.1.2	The Roles of Cost in Computing Helpfulness	142
5.3.1.3	Helpfulness at the Local and Global Perspectives	143
5.3.2	Helpfulness as a Heuristic	144
5.4	An Illustrated Example of Responsive Planning	147
5.4.1	Assistive Interaction Example	148
5.4.2	Adversarial Interaction Example	150
5.4.3	Independent Perspective	151
5.5	Concluding Remarks: Responding through Observation	152
6.	THE PRETCIL FRAMEWORK AND AN IMPLEMENTATION	155
6.1	Related Closed-Loop Interaction Approaches	157
6.1.1	Integrated Execution Monitoring and Automated Planning	157
6.1.2	Confirmation and Negotiation between Recognition and Planning	159
6.2	Overview of PRETCIL Framework	161
6.2.1	Example: Perception via Planning as Recognition	162

6.2.2	Example: Decision Making via Responsive Planning	163
6.2.3	Example: Simple Execution Monitoring	164
6.3	Overview of Implementation	164
6.3.1	Library Structure	165
6.3.2	State-Space Planning	166
6.3.3	Probabilistic Recognition as Planning	167
6.3.4	Responsive Planning Agent	167
6.4	Recognition as Planning using Heuristic Search	168
6.4.1	Notation	169
6.4.2	Approach 1: Sequence of Heuristic Searches	172
6.4.3	Approach 2: Extended Heuristic for Traditional Search	174
6.4.4	Experiments	181
6.4.4.1	BlockWords Domain Implementation	181
6.4.4.2	Procedure	182
6.4.4.3	Results	183
6.5	Algorithms for Faster Recognition as Planning	184
6.5.1	Multiple Goal Heuristic Search	186
6.5.1.1	Empirical Evaluation	192
6.5.2	Continuing the Search Process	197
6.5.2.1	Variations of Implementation	204
6.5.2.2	Connection to Plan Libraries	205
6.6	Concluding Remarks: Influences of Automated Planning and Recognition on Each Other	207
7.	HUMAN PERCEPTIONS OF ASSISTIVE AGENTS USING THE PRETCIL FRAMEWORK	210
7.1	Human-Subjects Studies with Intelligent Systems	211
7.2	Experiment Platform	214
7.2.1	BlockWords: Good for Human-Computer/Robot Interaction?	216
7.2.2	Revisions to Platform	216
7.3	Participant Recruitment	220
7.4	Experiment Procedure	222

7.5	Experiment Results	226
7.5.1	Human Performance Change	227
7.5.2	Human Emotional Evaluation	231
7.6	Concluding Remarks: Can PRETCIL Play with People?	236
8.	THE NEXT STEPS FOR DECISION MAKING FOR INTERACTION	237
8.1	Challenge Topic: Sufficient Information to Interact	237
8.2	Challenge Topic: (Un)Intentional Communication	239
8.3	Challenge Topic: What Information Actually Matters?	241
8.4	Challenge Topic: Integrated Architectures of Algorithms	243
8.5	Challenge Topic: Understanding People, Not Just AI	245
APPENDIX: FIRE EMBLEM HEROESTM IS NP-COMPLETE		249
BIBLIOGRAPHY		281

LIST OF TABLES

Table	Page
4.1 Elapsed Runtimes for Optimally-Trained Models (P_1)	121
4.2 Log-Evidence of Test Sets with Optimally-Trained Models	121
4.3 Number of Topics/Activities T and HMM States C in the Optimally-Trained Models per Partition of CAD-120.	123
6.1 Resources Consumed (BOOM and FIREWORKS)	184
6.2 Resources Consumed (DOPLR Heuristics)	184
6.3 Features of MGHS Variations	192
6.4 Resources Consumed (RaP with Single-Goal A*)	195
6.5 Resources Consumed (RaP with MGHS SUM_A*_UNW)	195
6.6 Resources Consumed (RaP with MGHS SUM_A*_W_NU)	195
6.7 Resources Consumed (RaP with Single-Goal Best-First)	195
6.8 Resources Consumed (RaP with MGHS SUM_BF_UNW)	195
6.9 Resources Consumed (RaP with MGHS SUM_BF_W_NU)	195
6.10 Resources Consumed (5-Block \rightarrow 6-Block Domain)	196
6.11 Resources Consumed (RaP with MGHS SUM_A*_UNW, Goal Pairs)	197
6.12 Resources Consumed (RaP with MGHS SUM_A*_W_NU, Goal Pairs)	197
6.13 Resources Consumed (RaP with MGHS SUM_BF_UNW, Goal Pairs)	198

6.14 Resources Consumed (RaP with MGHS SUM_BF_W_NU, Goal Pairs)	198
7.1 Human Subjects Survey: Response Means	232
7.2 Human Subjects Survey: Response Standard Deviations	232
7.3 Human Subject Survey: Absolute Value of Response Means	234

LIST OF FIGURES

Figure	Page
1.1 An illustration of how automated planning and recognition work together in the PRETCIL framework.....	18
3.1 A graphical model of a HMM with global transition matrix T . Shaded nodes represent observed random variables and edges $u \rightarrow v$ contain a conditional probability table $P(v u)$	70
3.2 A graphical model of a BOW with global distribution D . Shaded nodes represent observed random variables and edges $u \rightarrow v$ contain a conditional probability table $P(v u)$	73
3.3 LDA's generative process, graphical model, and mathematical representation.	74
3.4 A visualization of R_{ed} 's decision making and action execution processes. Directed edges indicate a dependency between two components, but dependencies are not necessarily probabilistic. The dashed boxes indicate the same variable under two different names to bridge the HMMs: $s_{k,n_k} = s_{k+1,1}$. Shaded nodes indicate observations. The partially shaded action nodes a_1, a_2, \dots, a_i are the executed actions according to π , which <i>might</i> be observed depending on the observation inputs. Plan recognition typically identifies π and intent recognition typically identifies G or some a_{i+j} where $j \geq 1$. \mathcal{A} , \mathcal{D} , and I are not shaded nodes because they are assumed, rather than observed.....	89
3.5 A focus on a single action execution HMM from Figure 3.4, broken down into its steps. Activity recognition typically identifies a_1 given $o_{1,1}, \dots, o_{1,n_1} \in \mathcal{O}$. These observations can range from raw signal data to discrete actions.....	89
4.1 A plot of unique word tokens in a collection of forty recorded plan executions at various granularities.	101

4.2	These most likely postures for some activity appear to share relations for ‘sitting’ such as legs bent and perpendicular to the torso, but how do we know this for certain?	102
4.3	Some human-interpretable features for a posture.	104
4.4	Proposed models enhancing LDA, based on their additional information.	110
4.5	Graphical model representation of the parameterized composite model with LDA, the composite model, and parameterized LDA as subgraphs.	111
4.6	Visualization of each model’s inferred topic and HMM state assignments for an execution of ‘having a meal’ which breaks down into ‘moving’ (1-83; 101-134; 156-194; 241-276; 346-370; 385-444), ‘eating’ (84-100; 371-384), ‘reaching’ (135-155; 333-345), ‘drinking’ (195-240), and ‘placing’ (277-332).	124
4.7	A heat map illustrating the distributions over objects Ω_t for each activity t in the optimally-trained parameterized LDA with respect to $P1$. Each column is single topic where red indicates the greatest probability mass and dark blue indicates the least probability mass.	124
5.1	Initial state and goals for our example problem with tuple $(cost_{\pi_G^*}, cost_{\pi_{R_{ed}+R_{ing}}^*})$. The specific problem instance’s goal word is bold-faced.	148
5.2	Recognized distribution over \mathcal{H} as each action of π_G^* is taken. The value of foresight parameter ϵ increases from left to right, top to bottom. The most costly goal, spelling BOTHER, becomes more likely as ϵ increases to look further ahead.	149
5.3	The state after R_{ed} performs the sixth action in its plan. The identified necessities are shown in red (satisfied) and green (unsatisfied) overlaid lines.	150
5.4	The state after R_{ing} performs its assistive responsive plan alongside R_{ed} ’s plan. All the necessities are satisfied so that R_{ed} just needs to stack the final block and complete its spelling task.	150

5.5	The state in Figure 5.3 two actions later, modified with respect to R_{ing} 's adversarial responsive plan $\pi_{R_{ing}}$. Stacking S on top of M undid two satisfied necessities while R_{ed} began to uncover the A block.	151
5.6	The state in Figure 5.3 four actions later, modified with respect to R_{ing} 's independent responsive plan $\pi_{R_{ing}}$ with a personal goal of spelling HOWL. R_{ed} does not use responsive planning and thus covers H with F, but R_{ing} also moves M instead of helping R_{ed} uncover A as they did during assistive interaction.	152
6.1	The PRETCIL framework's planning and recognition interaction loop. Components pass information forward for processing (gold) or backwards to improve performance in later iterations (green).	156
6.2	Current components in the library and their relations to each other. The arrows show the flow of information and/or use of an instance of the class. Extended dotted-border boxes contain the current functions that the developer needs to define.	166
6.3	An illustration of the FIREWORKS algorithm overlaid on the original search space. The prefix path ($I \rightarrow I'$) solving \mathcal{P}^O requires a single search (solid shaded), but each goal check must search (dotted borders) for its own suffix ($I' \rightarrow G_{1 \leq i \leq K= \mathcal{H} }$) solving $\mathcal{P}_O^{G'}$	174
6.4	An observation space (right, original state space is in the center based on the grid on the left) that illustrates how to find a non-optimal path from $(I, 0)$ to $(g, 1)$ first using heuristic h_G^O when h_G is the oracle heuristic.	178
6.5	An observation space (right, original state space is on the left) that illustrates how to find a non-optimal path from $(I, 0)$ to $(g, i \neq 2)$ first using heuristic $h_G^{\neg O}$ when h_G is the oracle heuristic.	180
6.6	KL Divergence comparing the distributions from our approaches to the original.	185

6.7 An illustration of MGHS’s search progression (dotted borders) with the sum and progress heuristics compared to individual heuristic searches per goal condition set G_i (solid borders). Overlapping solid-bordered regions indicate redundant exploration that MGHS saves. MGHS unnecessarily explores regions exclusively within the dotted border and not within a solid-bordered region (shaded).	190
6.8 An observation space (right, original state space is in the center based on the grid on the left) that illustrates the lack of any solution from $(I, 0)$ to $(g, 0)$. The red edges in the observation space show the bidirectional edges in the state space that would transition to the goal state if the <i>up</i> action was not observed.	199
7.1 Screenshot of the BlockWords two-player game.	215
7.2 Temporal resources consumed per non-tutorial level during the human subjects experiment. Levels with no results are not shown—all Group 1 participants skipped them, and they were removed for Group 2.	229
7.3 Survey responses from participants in the human subjects experiment. Participants completed the survey after finishing the game or choosing to terminate their play session.	233
A.1 The three types of layers used to construct a FEH gadget given some S and z . The board joins these layers along the horizontal edges, with one (a) Enemy Layer on the top, one (b) Set Element Layer per $s \in S$ in the middle, and one (c) Player Layer on the bottom. This sets the vertical tile length of the board, and the horizontal tile length of the board pads columns of tall object obstacle (TOO) tiles to make a square board shape.	270

CHAPTER 1

MOTIVATION: WHY INTERACTIVE ARTIFICIAL INTELLIGENCE MATTERS

Interaction is something we do all the time, from things as simple as opening a doorknob to things as complex as living our daily lives amongst all the people around us. ... We do it so much that we take the ability to interact with others for granted, but for machines, such as our computers, mobile devices, and robots, this is actually really hard. ... Hopefully one day, machines can interact with us, just like how we interact with each other.

— Richard (Rick) G. Freedman,
excerpts from his Three-Minute Thesis finalist speech

As artificial intelligence (AI) technology develops, *interactive intelligent systems* have become more personalized and context-aware, serving people beyond the role of instrumental tools towards ‘smart’ and cooperative assistants/companions [1, 201, 210, 243] in our daily lives. More complex intelligent systems are thus increasingly interacting with people at work [145, 247], at home [109, 251], and on the road [28, 287]. For these systems to be effective, it is crucial to develop a far better understanding of *how to recognize human activities and respond appropriately in a timely manner*. People engage in the world via *dialogue* and by taking *goal-directed* actions. A wide array of sensors can provide computers some insights into what a person is doing [10, 262, 291]. Ideally, the sensors’ insights can inform automated decision-making algorithms about what is happening so that it has sufficient understanding of the situation to determine in which ways to respond or engage in an interaction.

In e-learning, for example, intelligent tutoring systems are becoming increasingly interactive, tracking students’ skill levels, forming hypotheses about their solution

strategies, and providing machine-generated support when necessary to increase learning gains [7, 163, 234]. In manufacturing, robots are “coming out of their cages” and interacting closely with coworkers in the same space [54, 174, 209]. Self-driving cars with varying degrees of autonomy interact with both pedestrians [278] and the humans in the driver’s seat [136, 288] to transport passengers while handling traffic situations. The rapidly growing elderly population has motivated the development of robots that can keep them company [34, 181], assist with therapy [72, 139], and keep them safe [62, 241]. Other examples of interactive intelligent systems include text prediction in mobile phones [261] and augmentative and alternative communication (AAC) devices [78, 274, 275, 118], personal desktop assistants [204], robotic assistants [13, 273], and robots that can help children learn [286, 155].

Furthermore, users can provide feedback to present-day intelligent systems, such as selecting suggestions from recommender systems [49, 212], that can affect future interactions. The recent revival of explainable AI [3, 76] and responsive push for interpretable AI [236] allow systems to provide human-understandable and acceptable¹ justifications for the actions that they take [138, 31]. These advances allow machines to operate with high levels of autonomy, yet still collaborate with human partners in joint human-machine teams [8, 270, 44]. The dynamic and long-term nature of these interactions raise the need for agents that *close the loop* between recognizing users’ activities and generating actions that sustain the collaboration [108, 142, 162, 290].

One major challenge for such interactive intelligent systems is that unlike machines that follow instructions step by step, *humans do not act in a predictable, formulaic manner*. There are many different ways to perform a task [285], and an individual may use whichever action sequence seems more convenient. Humans often explore their

¹The notion of acceptance is important because people can understand explanations of the form “I took this action because it got the highest score using my internally-programmed equations.” However, they often have similar validity to the blanket justification “I took this action just because I felt like it.”

surroundings, use trial-and-error, make mistakes, and learn from these extraneous actions [193]. They also intermittently *perform seemingly irrelevant actions*, such as answering the phone, adding noise to the observed activity. A person's exact goals are often unknown and subject to change from these actions, which adds to the complexity of understanding what is happening.

A second challenge is the *need to interpret the human's activity and determine whether and how to respond under tight time constraints*. Recognition algorithms need to infer what people's goals are and how they are achieving their goals. A robot can infer that a person is going to the kitchen, but it also needs to know which route the person is likely to take to avoid colliding with the person. Pausing for a long period of time to compute not only seems unnatural in an interactive situation, but could render the system impractical. Humans are willing to give a robot control of a situation if it raises efficiency [99, 42], whereas lack of efficiency lowers the likelihood that a person will consider future collaboration. Hence the machine's computed responses need to be *optimized given the scenario's time constraints*. Solution techniques must therefore rely on *anytime algorithms* [298], which offer adjustable runtime and a tradeoff between time and solution quality. Finally, it is crucial to *establish a degree of confidence about the result*. For example, an interactive intelligent tutoring system should not offer help before it is fairly confident about the student's confusion.

1.1 Overview of Past Approaches

Several areas of AI; particularly automated planning, plan recognition, activity recognition, and intent recognition (the latter three are abbreviated together as PAIR); are developing component solutions to these challenges. Automated planning is devoted to methods and algorithms for autonomous, goal-driven action selection. It is usually considered together with decision making and problem solving; we discuss automated planning in more details in Chapter 2. Considered to be automated

planning’s inverse process, PAIR focuses on using observations to identify the driving goal; observations are commonly in the form of others’ actions and/or changes to the world. Though considered together, each specialization within PAIR identifies a different aspect of the goal. Plan recognition matches tasks and overall plans as explanations that complement an agent’s recent activity. Activity recognition finds higher-level representations of what is observed, serving as the gateway between the real world and the plan recognition model when sensor data are involved. Intent/goal recognition predicts future actions and/or how the environment will change upon accomplishing the goal. We discuss PAIR in more details in Chapter 3.

There has been tremendous progress in these areas as PAIR evolved into its own field of research, spurring a wave of dedicated professional meetings (e.g. [96]), published monographs (e.g. [263]), and the development of numerous applications. Unifying principles behind the developed approaches have emerged, leading to the introduction of generalized algorithms [35, 74, 202, 227]. However, these algorithms typically process the observed activity and output recognized information [128, 283, 296]. The system computes a response to the user, if at all, using a separate planning algorithm. Automated planning algorithms have been studied for a wide variety of problem classes using approaches such as heuristic search [113, 27], decision-theoretic methods for nondeterministic domains [21, 61, 141], and decomposition methods that recursively break down tasks into subtasks [70, 11, 257]. Despite all this work, currently *automated planning and each research area in PAIR mostly studies its own specific problem(s) independently of the others*.

In the few exceptional cases where these research areas have been combined, researchers simply *pipelined* the information from one algorithm to another. This paragraph provides a brief overview of a few examples for motivation, and some of them will be discussed in more detail when relevant to later chapters. Kelley et al.’s intent recognition algorithm [151, 152] combines multiple hidden Markov models (HMMs)

to identify what action a user is trying to perform based on point-cloud information from a red, green, blue, depth (RGB-D) sensor. However, their interactive system acts using a table of response actions for each recognized intention, and the response is only selected from the table once the intention is recognized. In another example, Levine and Williams [174, 176] use a planner’s precomputed sequence of actions as instructions that a user must follow; branching points identify where the user may make a choice. An observing robot monitors the user’s execution of the plan and identifies the action selected at each branch point. Again, the robot is not able to plan or act until the user’s choice is determined. Geib et al.’s [90] recent work introduces collaborative behavior between a robot and human that instead acts independently on subtasks. After the observing agent is able to make a confident prediction using the plan recognition component, it can identify the remaining subtasks from a library of plans and “negotiate” them with the human. The plan library is again pre-calculated and the robot is unable to help without permission, but it can preform these negotiations at any time. It is important to note that *the independent subtasks prevent direct interaction between the human and robot*, but Levine and Williams’s [174, 176, 175] approach does allow direct interaction between the agents.

Reinforcement learning (RL) and inverse reinforcement learning (IRL) have been used to determine how to act in each world state [265, 104] and identify the user’s preferred states [169, 268], respectively. However, these approaches also have their limitations, particularly generalizing to large state spaces. Research on transfer learning can handle simple changes to the state space [203], but not to varying user preferences for which little training data may be available. Unfortunately, this is not just a limitation in RL/IRL, but also in automated planning and PAIR methods. As pointed out by Zhang and Parker for their highly accurate activity recognition algorithm [296], any small change to the environment degrades performance because the *input representation is not generalizable outside the trained space*.

Consequently, existing methods often restrict the user’s input actions. This is perhaps reasonable in software applications that use fixed buttons, but not in systems handling more open-ended human interactions. At the present moment, they often restrict the number of gestures [67, 294], phrases [5], or buttons/cards representing the available inputs [56, 297]. Imagine a manufacturing robot that can only hand a human a tool if she outstretches her hand, or a robot serving beverages that requires the client to place the cup in just the right location to trigger pouring [221]. This places *severe restrictions on possible interactions*.

1.2 Complexity of Interaction

Computational complexity measures the difficulty of solving problem types with respect to the resources (time and memory) necessary for finding a solution as the problem instance’s size increases. The simplest problem types have instantaneous solutions no matter how large the problem instances become, and increasing difficulty amounts to eventually checking every possible solution (though confirming a solution can require significantly different amounts of resources depending on the complexity). We can similarly compare the computational complexity of different types of interaction scenarios, where instances scale in domain size (rather than number of agents). Research on complexity for single-agent decision making has generally found that the amount of uncertainty plays a role in distinguishing difficulty, and we will discuss how this also applies to decision making for interaction. Most importantly, we explain how this insight provides further motivation for integrating recognition with decision making as a means of reducing uncertainty.

Collective interactions, such as centralized multi-agent systems, are often compiled into single-agent problems with multiple actuators (like a single organism operating multiple limbs). Thus we strictly discuss decentralized multi-agent systems where the

notion of a “hive mind” is absent and each agent acts autonomously and independently.

1.2.1 Increasing Complexity for Automated Planning

Within automated planning for single-agent tasks, *complexity increases as predictability decreases* for various elements. Classical planning enforced many assumptions to avoid these uncertainties [237], and the standard logical representations (such as STRIPS) have polynomial space (PSPACE)-complete complexity [37]. However, including non-determinism (unpredictability of action outcomes) using logic-based factored representations can dramatically increase the complexity [179]. Likewise, partial observability (unpredictability from hidden information) increases the complexity [182]. These are the most common forms of uncertainty currently studied in automated planning, but others have been considered such as generalizations (unpredictable quantities of things in the world and task specifications) [256, 127, 126, 180, 242, 20] and incompleteness (unpredictability of how actions work) [207]. We are unaware of any research on their complexity.

1.2.2 Increasing Complexity for Game-Theoretic Situations

The game-theoretic perspective does not need to consider how the agents will interact because each agent has its own personal payoffs/rewards based on everyone’s action decisions, and solutions that maximize everyone’s payoff with respect to *rational play* are called Nash Equilibrium. Finding Mixed Nash Equilibrium (the most generalized Nash Equilibrium) for any number of agents is in PPAD-complete, a special complexity class where the worst-case approach is trying every option because at least one of them is guaranteed to be the solution [58]. However, our interaction scenarios require agents to complete tasks rather than optimize payoffs. This is a different class of problems because solutions, in this case instructions for what each agent should do, are not guaranteed to exist.

For any type of interaction with complete unpredictability, simply giving every agent its specific tasks privately and no prior knowledge of other agents in the world is a partially-observable stochastic game (POSG). When all the POSG agents have the same reward function (taking an action in a certain moment receives the same payoff), the worst-case complexity is solving a problem represented by a decentralized partially-observable Markov decision process (DEC-POMDP) [23]. Finding a solution for such a problem is in NEXP-complete, which is far more complex than PSPACE-complete for classical planning problems. If the POSG agents' reward functions are designed to be competitive with each other, then finding solutions is NEXP^{NP} -hard [98], which is even more complex than NEXP-complete. All these problems generally take a long time to solve even if we could check every possible solution simultaneously.

1.2.3 Increasing Complexity for Interaction

We respectively consider interactive scenarios by *how predictable the interactive partner is*. As the simplest case, suppose that each interactive partner R_k for $k \in \{1, 2, \dots, K\}$ follows a fixed policy $\pi_k : S \rightarrow A_k$ where S is the set of possible states in the world and A_k is the set of actions R_k can take. This means that R_k will always perform the same action in a given state of the world. If the interactive intelligent system R_0 knows all these policies, then R_0 can perfectly predict what each R_k will do at all times. Thus R_0 can complete the interactive scenario like an automated planning problem; the interactive partners are simply modeled as part of the environment. In particular, the interactive intelligent system composites all the agents' actions with its own action a for the state transition function $T : S \times \bigcup_{i=0}^K A_k \rightarrow S$ from state s_0 :

$$s' = s_{K+1} \text{ such that } s_i = T(\pi_{\sigma(i-1)}(s_{i-1}), s_{i-1}) \text{ and } \pi_{\sigma(0)}(s_0) = a$$

where σ denotes the permutation of agents under turn-order. If agents can act simultaneously, then any conflicts between their actions' effects will need resolution; this is

represented with a transition function that instead allows the cross-product of chosen actions as inputs $T^\times : S \times (A_0 \times A_1 \times \dots \times A_K) \rightarrow S$:

$$s' = T^\times ((a, \pi_1(s_0), \dots, \pi_K(s_0)), s_0).$$

Upon introducing any degree of unpredictability to the interactive partners, we need to consider the types of interaction between agents because *their motivations will impact the heuristics available*. For our discussion, we present three primary classes of interaction:

Assistive Agents have tasks that are intended to synergize. This can be in the form of (1) all agents sharing a common goal and receiving mutual benefits, such as a competition team, or (2) some agents being assigned a task in the form of “help other agents accomplish their goals”, such as collaboration tasks and helping.

Adversarial Agents have tasks that are intended to conflict. This can be in the form of (1) all agents sharing a common goal with exclusive benefits to a subset of them, such as a race, or (2) some agents being assigned a task in the form of “prevent other agents from accomplishing their goals”, such as interception tasks and bullying.

Independent Agents have individual tasks that are related by coincidence. This means that the agents can work on their own simultaneously; there is no expectation of aiding or hindering other agents. However, the completion of their tasks might overlap beneficially or require the use of more resources than are available. Though these interactions are assistive or adversarial by nature, the agent’s motives make this nature unintentional. Taking turns, waiting, sharing, and other such strategies might be necessary for agents to accomplish their otherwise independent goals when these intersections occur.

One aspect of unpredictability will be over a collection of J fixed policies per interactive partner $\pi_k^j : S \rightarrow A_k$ for $j \in \{1, 2, \dots, J\}$. Multiple policies can represent different approaches to doing a single task, a set of possible goal conditions to satisfy, hypotheses regarding how the agent behaves, etc. If the interactive intelligent system R_0 knows all these policies in advance, then it must reason over which policy each agent will follow at a particular state. That is, R_k has a set of actions that it may perform at each state s : $\pi_k^1(s), \dots, \pi_k^J(s)$. For turn-based adversarial situations between exactly two agents, it might be possible to use the minimax algorithm if the notion of progress for one agent corresponds to preventing progress for the other agent. Then the interactive intelligent system wants to perform actions that generate states that *hinder progress for the interactive partner regardless of which policy it follows*. Unfortunately, these tree-based algorithms are not as reliable when there are more than two agents because the heuristics for multiple agents' progress are not easy to compare with trichotomy relations. Aggregate measurements are also difficult because of opportunities for ad-hoc team formation and betrayal.

When the interaction is assistive or independent, an intelligent interactive system wants to perform actions that generate states that *enable progress for the interactive partners regardless of which policies they follow*. However, doing this at the price of making its own progress is not ideal (especially for independent interactions). Thus some form of agreement between agents is sometimes necessary. As briefly mentioned above, many interactive interfaces currently restrict the set of user inputs and/or set of system outputs. This is an enforced social cue that is inherently followed by the design of the system, but it places restrictions on the interactive partners' policies that facilitate its own decision making for its interactive tasks [244]. When social norms are not assigned to the world for resolution, communication protocols will be necessary to inform and restrict policy choices. This can be done silently via

legible actions [65], verbally via negotiations [90, 55], or directly via explanations and requests [271, 235].

Another form of unpredictability, which will be the primary one addressed in this work, is the lack of knowing each interactive partner’s goal. If the interactive intelligent system knows a list of possible goals a priori, then observing the other agents can assist in narrowing down the options [227]. If the environment is designed in such a way that each task’s solution has unique features or a fixed policy [156], then this might have similar complexity to knowing a set of fixed policies because the agents’ actions are limited. However, we cannot design everything in such a way due to reconstruction costs or constraints on functionality/purpose, and then it becomes more difficult than simply knowing which actions must be taken at each state. In particular, all applicable actions are possible, and performing a small number of actions in a few states can appear to be task-ambiguous. As of the time writing this, Shvo and McIlraith recently considered dynamically acting in the world to affect the environment for similar consequences [246].

We address decision making for interaction when this ambiguity occurs in Chapter 5. For adversarial interaction, identifying the goal is important in order to make sure that actions actually hinder the correct task; one does not want to wait at the end of a hallway to intercept someone who can crawl out a window to reach their destination. In assistive interaction, it is equally important because R_0 will not be considered ‘helpful’ if the agent does not do anything relevant to the task. In independent interaction, this is not as critical, but it can be practical to determine whether there is a chance of overlapping tasks or resources—following the adage “an ounce of prevention is worth a pound of cure” in the case that the overlap is a conflict with another agent’s goal and the adage “hitting two birds with one stone” in the case that the overlap benefits all the agents’ goals at once.

1.2.4 The Potential to Reduce Complexity through Recognition

The sections above presented the names of multiple computational complexity classes without any background other than their relative comparisons of difficulty. Although we will not discuss all of the them, we will present some details for two that are relevant to motivating the approach in this dissertation. As a disclaimer, we use the term ‘-complete’ while describing ‘-hard’. The difference is that the problems C in the ‘-complete’ version of the complexity class are a subset of the ‘-hard’ version of the complexity class where any problem in the ‘-hard’ class H can be reduced to (a.k.a. turned into) C using a polynomial-time algorithm with respect to H ’s size. Effectively, this reduction means that we can solve any H if we can already solve C , and the conversion used for this property is computationally efficient compared to the efforts needed to solve H (and thus C).

PSPACE-complete complexity requires a solution program whose tape-encoding is polynomial in size with respect to the tape-encoding of the problem instance’s input. If $\text{PSPACE} \neq \text{NP}$ ², then PSPACE-complete problems conceptually require nothing simpler than an Alternating Turing Machine in order to verify a solution. The alternation is between existential and universal quantification.

“For all accepting states s_I^{\forall} , there exists a state s_I^{\exists} transitioning to s_I^{\forall} such that, for all *possible* states s_{I-1}^{\forall} transitioning to s_I^{\exists} , there exists a state s_{I-1}^{\exists} transitioning to s_{I-1}^{\forall} such that, … for all *possible* states s_0^{\forall} transitioning to s_1^{\exists} , there exists a state s_0^{\exists} transitioning to s_0^{\forall} such that the problem instance’s input can also transition to s_0^{\exists} .”

The s_i^{\exists} states reveal a solution in the form of a collection of paths in the Alternating Turing Machine that lead to the accepting states.

As a more concrete example, the game of Geography/Shiritori in its abstract form is PSPACE-complete. In this game, two players have to name unique locations/words that start with the last letter of the previous word—a player who cannot think of a

²The author believes this is likely true, but the equality is still an unsolved problem.

novel word under these constraints loses. The typical problem decides whether the player who goes first can win³ with a specified starting word and list of words that each player knows. Thus s^{\exists} are the words the first player can choose to say and s^{\forall} are the *possible* responses the second player can say; the latter states are accepting when they contain no possible responses.

However, one can technically solve games with a finite set of possible states such as Geography/Shiritori through brute force and exhausting all possible states. This is considered to be constant runtime with respect to the input size, even if that constant is very large, because there is no scaling of the game to alter the problem's difficulty. Thus the computational complexity version of the question abstracts the game to become *generalized* (not to be confused with generalized planning mentioned above) with respect to arbitrarily long vocabulary lists and number of letters available in the language; this creates a directed graph where edges denote what words may be used based on the most-recently chosen word [178]. Scaling the problem size now scales the graph size via the number of nodes and/or number of edges.

NP-complete complexity requires a solution program whose runtime is nondeterministic polynomial with respect to the size of the tape-encoding of the problem instance's input. If $P \neq NP$ ⁴, then NP-complete problems conceptually require nothing simpler than a Turing Machine running a polynomial-time tape in order to verify a solution. The nondeterminism relates to running the verification on all possible solutions in parallel simultaneously. The solution program can return the first solution that the verification program accepts, but the program realizes there is no solution in the worst case once all the verification program rejects all possible solutions.

³The potential to win is important because it allows inspection before the game starts, regardless of what the players actually choose to do.

⁴The author believes this is likely true, but the equality is still an unsolved problem.

As a more concrete example, the k -subset sum puzzle is NP-complete. In this puzzle, a set of non-negative numbers S is given along with a desired sum k —the challenge is to find a subset of numbers in S whose sum is k . The typical problem decides whether a solution exists to this puzzle given some S and k . Thus the set of possible solutions is the powerset 2^S , and each element of 2^S can be verified within $c|S|$ steps (for some constant $c \in \mathbb{N}$) by adding its contents and comparing the sum to k . Scaling this problem is therefore agnostic of k , but increasing the size of S will increase the verification efforts.

From a layperson’s perspective, NP-complete problems are solved using “guess-and-check” methods while PSPACE-complete problems are solved using an “I’m thinking that you’re thinking that I’m thinking...” approach. It is hopefully clear that this makes solving PSPACE-complete problems much more difficult than solving NP-complete problems. The concrete examples also start to expose the nature of problems in each complexity class: PSPACE-complete problems often involve games between two decentralized agents and NP-complete problems often involve puzzles for a single agent. This is an allusion to Section 1.2.3’s increase in complexity as well: *when the policies that other agents follow are known, then the interaction can be solved like a puzzle rather than a game*. Therefore, employing recognition in the process of deciding how to interact ideally sheds light on what the other agents are doing to reveal their motives and upcoming actions. If R_0 recognizes enough to make each R_i predictable, then we can effectively *decrease the complexity of deciding how to respond to others*.

As a proof-of-concept for complexity reduction, we prove in Appendix A that the currently popular mobile videogame Fire Emblem HeroesTM in its generalized form is NP-complete because a deterministic expert system controls the in-game opponent. As far as we are aware, this is the first time that a two-player game of any form, even if one player’s actions are constrained without the ability to reason, has been proven

to be NP-complete. However, this is *not* intended to claim that integrating recognition with planning will reduce the complexity of decision making for interaction all the way down to NP-complete. Instead, this provides evidence that complete predictability of other agents essentially transforms the decision-making-for-interaction problem into a single-agent decision-making problem, which means that the *computational complexity of the single-agent decision-making problem is the lower-bound complexity to which the decision-making-for-interaction problem can be simplified*. Recall above that classical planning for a single agent alone is PSPACE-complete [37]; thus the potential complexity for deciding how to interact through something like a DEC-POMDP (whose complexity is far worse than PSPACE-complete [23]) might be simplified to PSPACE-complete in the best case using recognition to predict other agents' behavior.

1.3 Videogames: A Motivating Interactive Domain

Although we initially discussed motivations in household robotics, elder care, robotic coworkers in the factory, children's education, vehicles with varying levels of autonomy, and personal assistants within devices, it is important to select one as a focus for this dissertation and running examples. The methods introduced will be applicable to many of the situations described above, and thus choosing one to which we apply the approach will not limit how things work. However, it will serve as a simpler testbed for us to understand the ideas and examine them in a more controlled environment.

Videogames have been a form of entertainment with its rises and falls in interest. Following the increased ubiquity of mobile devices and games being designed on them for short-term play sessions that can appeal to casual players as much as hardcore players, they are now a major source of interaction between users and their electronic devices [29]. They are not just ubiquitous and a clear example of interaction, but

their design is very convenient for research testing. For example, unless relying on the screen’s image like the Angry Birds AI Competition [229], the game’s internal data structures yield a fully observed state without any noise. Likewise, a user’s interactions via buttons and touch screens have little-to-no noise so that the player’s actions are more easily available. That is, we can *avoid many forms of uncertainty found in autonomous planning* to reduce the computational complexity to no more than what is necessary, *focusing on the forms of uncertainty specific to decision making for interaction*. Lastly, the virtual nature of videogames means that integrating computer-controlled players is often as simple as a socket connection to a program, and the human players might not be aware of who is controlled by a human (to avoid bias between playing against a computational agent and another person).

In the majority of the research discussed in this dissertation, the ongoing video-game environment is based on a common toy problem in the autonomous planning community called Block Words (also called Blocks World). A table contains stacks of blocks that each have a letter inscribed on them, and the agent is tasked with picking-and-placing blocks until there exists a stack of blocks whose inscribed letters spell a specific goal word when read from top-to-bottom. Actions either pick up a block that is on top of any stack or put down a held block on top of any stack or the table, but an agent can only hold up to one block at a time. Our extension includes two agents, each able to hold up to one block at a time, that take turns performing actions. Either agent may pass their turn with a no-op action. Though we will discuss adversarial and independent interaction during the theory and concepts, our current implementation for the computer-controlled agent is exclusively assistive for experimental evaluation.

1.4 Overview of Introduced Framework

In order to close the interaction loop, we will first provide background and recent developments on the relevant areas of study. Chapter 2 covers automated planning, primarily the classical representations since our initial implementation will use them with some discussed extensions. Chapter 3 discusses plan, activity, and intent recognition both individually and as an overall concept. Chapter 6 introduces the Planning and Recognition Together Close the Interaction Loop (PRETCIL) framework and shows how to use or extend the work in the previous chapters to create an initial implementation of the framework. This introduction to the PRETCIL framework builds upon research in Chapters 4 and 5, which respectively discuss integrating individual components and their challenges. Lastly, we discuss experimental setups, tests, and results for the interactive aspects of our implementation of the PRETCIL framework in Chapter 7. We also provide a postmortem regarding this implementation of the PRETCIL framework because many concepts that made sense in theory encountered issues in practice. Although negative results are rarely published, we prefer the reader be aware of these missteps to avoid repeating them. Inspired by the PRETCIL framework and work in this dissertation, Chapter 8 concludes this primarily hypothesis-generating research with a discussion about new research directions concerning *decision making for interaction*.

The goals of the PRETCIL framework address the fact that, though part of the environment, interactive agents have their own decision making processes that do not necessarily follow the natural physics and rules of the environment itself. This removes the patterns that are usually assumed in classical automated planning, and planning under uncertainty can easily become intractable considering what to do for every possible decision an interactive agent might make. Thus we propose recognizing these interactive agents' actions, plans, and intents to help guide the decision making process. Likewise, the decision making process can inform a recognition component

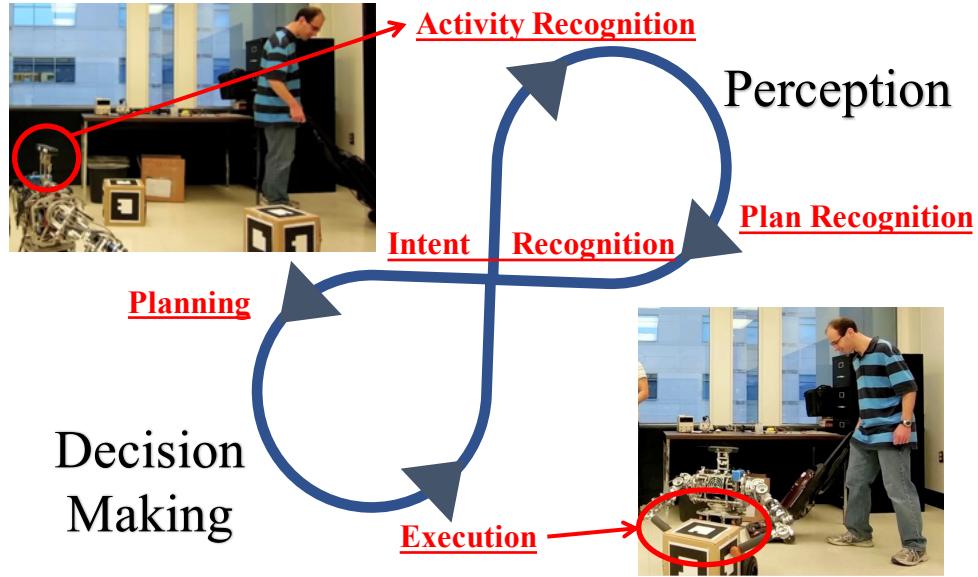


Figure 1.1. An illustration of how automated planning and recognition work together in the PRETCIL framework.

about what to expect so that interaction quality and anomalies may be better evaluated. Therefore, integrating both planning and recognition algorithms into *processes that work simultaneously and share information* will introduce ways to make decisions in the environment while accounting for interactive agents and their interactions. Figure 1.1 previews the PRETCIL framework.

1.5 Dissertation Contributions

Following the overview and summary in the previous section, we briefly denote the research contributions of this dissertation. Each one also notes the section in which it is located for quick reference.

1.5.1 Primary Contributions

- The unification of plan, activity, and intent recognition as multiple perspectives of an overarching problem: Section 3.5.

- Introducing plans as bags, rather than strictly sequences, of actions to enable simultaneous activity recognition and approximate plan recognition: Chapter 4. This includes investigations of representation (Section 4.2) and model designs (Section 4.3) for such algorithms.
- Formalizing the responsive planning problem, which introduces decision making that accounts for what others are doing in the environment, and providing initial approaches to solve it: Chapter 5.
- Integrating automated planning and recognition for interaction in the PRET-CIL framework, including a library for developers to use with their own code: Chapter 6.

1.5.2 Additional Contributions

- A formal proof that the generalized form of Fire Emblem HeroesTM is NP-complete, which contributes to the motivations discussed in this chapter: Appendix A.
- Introducing a metric to evaluate interaction based on an agent's helpfulness: Section 5.3.
- Identifying how to run Recognition as Planning (RaP) algorithms using heuristic search in a general state space, rather than just a STRIPS-represented state space: Section 6.4.
- Investigating various approaches to improve the run-time performance of RaP algorithms: Section 6.5.
- An explanation about how plan libraries and planning domains, two separate approaches to recognition tasks, are related under commonly used conditions and assumptions 6.5.2.2.

- Experiments exploring how actual people perceive the interactive experience with an agent using our implementation of the PRETCIL framework: Chapter 7.
- A list of new open problems related to decision making for interaction, including additional motivation for more human-aware design of human-aware AI: Chapter 8.

CHAPTER 2

AUTOMATED PLANNING

So remember to work hard, be healthy, and play safe. Now that is a PLAN [points at brain] that YOU [points at the audience] should always put into ACTION [changes pointing hand into a thumbs-up gesture]!

— Richard (Rick) G. Freedman,
concluding the public service announcement in an educational video
submitted to the AAAI Video Competition

One of the earliest challenges posed to the AI community involved machines being capable of making decisions autonomously at or above the level of *human experts* (a person with sufficient knowledge to make an informed decision). This led to the establishment of the automated planning and scheduling community that particularly studies representation of tasks, problem solving under various conditions ranging from uncertainty to resource constraints, and higher-level decision making processes such as metareasoning.

Throughout this chapter, we will investigate some common automated planning frameworks and see their specific perspectives on the following generalized formal definitions:

Definition 1. A planning problem is a tuple $\mathcal{P} = (\mathcal{D}, I, G)$ where \mathcal{D} is a domain that models the world, I provides the initial setup of the world, and G lists the problem's completion conditions.

Definition 2. A planning domain models the world in which the agents act. The contents of its tuple should describe the set of states S and the set of actions A that transition between states.

Definition 3. *The state space \mathcal{D}_{enum} is a graph derived from planning domain \mathcal{D} with vertices S that are joined by edges from instances of elements in A . The direction of the edges show the before-and-after relationship (a state transition) for performing the action associated with the edge, and bidirectional edges indicate that an action is reversible. This is like a roadmap navigating the manipulation of the world, illustrating how various actions change it.*

Many automated planning solvers are based on search techniques through \mathcal{D}_{enum} that find paths from I to some $s \in S$ that satisfies G . Each path is analogous to a list of actions that change the environment from the initial state (the ‘now’ moment) to some goal state that satisfies the problem’s completion conditions. Failing to find a path (with a complete search algorithm) means that there is no solution for \mathcal{P} using its current formulation, but it can also be the case that a solution is found that is difficult to perform in reality. Much like the programming adage, AI methods can only do what they are told, regardless of whether it is what the user wants.

2.1 Logic-Based Factored Representations

Classical approaches to automated planning applied various assumptions to simplify the problem, which aids the application of theoretical analysis. The representation that satisfied all the assumptions was named STRIPS after the STanford Research Instistute’s Problem Solver (STRIPS) [73], which was the representation language used for the solver’s inputs. It introduced a logic-based factored representation that has become the inspiration for most present-day representations, especially the Planning Domain Definition Language (PDDL) [191, 75]. Factored representations decompose a state into sets of features that are viewed and manipulated independently, similar to how pieces of a jigsaw puzzle represent a small aspect of a complete picture.

Definition 4. A fluent f is a logic statement, evaluating to either true or false, that can describe the world. The set of all defined fluents will be called F , and each STRIPS state $s \in S$ is a set of defined fluents that are true in its corresponding world instance. That is, $S = 2^F$.

Definition 5. The closed-world assumption states that anything that is not acknowledged to be true must be false. That is, nothing can be unknown, and one naively assumes that the unknown simply cannot be true. Specifically, the fluents not listed in a state, $F - s$, are false in s .

Definition 6. A STRIPS action $a \in A$ is a tuple that describes a way to manipulate the world. The tuple contains a name name_a , preconditions $\text{pre}_a \subseteq F$, add effects $\text{add}_a \subseteq F$, and delete effects $\text{del}_a \subseteq F$. The action is applicable in state $s \in S$ when the preconditions are true $\text{pre}_a \subseteq s$, and performing an applicable action changes the state to $a(s) = (s - \text{del}_a) \cup \text{add}_a$. That is, the effects take place so that all the add effects become true and all the exclusive delete effects (in case a fluent appears in both sets) become false. It is also possible to assign each action a unique cost $\text{cost}_a \in \mathbb{R}^{\geq 0}$. Without such an assignment, it is assumed that all actions in A have a uniform cost.

Definition 7. A STRIPS domain is a tuple $\mathcal{D} = (F, A)$ where F is the set of fluents and A is the set of actions. $D_{\text{enum}} = (S, \{(s, a(s)) | s \in S, a \in A, \text{pre}_a \subseteq s\})$ is the resulting state space where directed edges connect state changes by their applicable actions.

Definition 8. A STRIPS problem is a tuple $\mathcal{P} = (\mathcal{D}, I, G)$ where $\mathcal{D} = (F, A)$ is a STRIPS domain, $I \in S = 2^F$ is the initial state, and $G \subseteq F$ is the set of goal conditions. A state satisfying the goal conditions happens to be any $s \in S$ such that $G \subseteq s$.

With this formulation, we can search for solutions to \mathcal{P} using traditional search methods (see Algorithm 1) in the search space beginning with I and stopping when

either (1) a state satisfying G is explored or (2) all reachable states from I are explored.

In the former case, we can trace the searched path back to I and recover the actions performed in order. The action sequence is the actual solution that we return.

```

Input: STRIPS problem  $\mathcal{P}$ 
Output: A path  $\pi$  from  $\mathcal{P}.I$  to  $\mathcal{P}.G$  or NIL

1 initialize frontier data structure;
2 initialize explored hashmap;
3 initialize  $\pi$  to an empty list;
   //Each pair is of the form (state, parent)
4 insert pair  $(\mathcal{P}.I, \text{NIL})$  into frontier;
5 while frontier is not empty do
6   insert frontier.top() into explored with key frontier.top().first;
7    $current \leftarrow \text{frontier.remove().first}$ ;
   //When the goal state is explored, recover its path to return
8   if current satisfies  $\mathcal{P}.G$  then
9      $ancestor \leftarrow \text{explored.get}(current)$ ;
10    repeat
11      | append ancestor.first to  $\pi$ ;
12      |  $ancestor \leftarrow \text{explored.get}(ancestor.second)$ ;
13      | until ancestor.second is NIL;
14    return  $\text{reverse}(\pi)$ 
15  end
   //Expand the state into all its children
16  foreach action  $a \in \mathcal{P}.D.A$  do
17    if a is applicable in current then
18      | insert pair  $(a(current), current)$  into frontier;
19    end
20  end
21 end
22 //If no return yet, then no goal was found in the entire state space
23 return NIL

```

Algorithm 1: Traditional Search for Automated Planning

Definition 9. A plan π is a sequence of actions $a_1, a_2, \dots, a_{|\pi|} \in A$ such that each action is applicable when performed in order from the initial state

$$pre_{a_{i+1}} \subseteq a_i(a_{i-1}(\dots a_1(I)\dots)) \text{ for all } i \text{ from } 1 \text{ to } |\pi|$$

and the resulting state after completing all the actions in order satisfies the goal conditions

$$G \subseteq a_{|\pi|} (a_{|\pi|-1} (\dots a_1 (I) \dots)) .$$

If the actions have uniform cost, then the cost of π is $\text{cost}_\pi = |\pi|$. If the actions have assigned costs, then the cost of π is the total action cost $\text{cost}_\pi = \sum_{i=1}^{|\pi|} \text{cost}_{a_i}$.

Definition 10. A plan π^* is optimal for STRIPS problem \mathcal{P} if there does not exist an alternative plan π' that also solves \mathcal{P} and has a lower cost ($\text{cost}_{\pi^*} > \text{cost}_{\pi'}$).

Humans are able to reason over search directions and find solutions faster than systematic search methods such as breadth-first (frontier is a queue data structure) and depth-first search (frontier is a stack data structure). This led to the development of *heuristic search* algorithms that select search directions based on how ‘close’ states seem to be from satisfying the goal conditions, using priority queue data structures for the frontier.

Definition 11. A heuristic function $h : S \rightarrow \mathbb{R}^{\geq 0}$ approximates the remaining cost of a plan, with respect to the goal conditions G , from some state in the state space. Ideally, h is quick to compute.

Definition 12. A heuristic function h is admissible if its approximations from every state never overestimate the remaining optimal plan cost from that state. That is, for all $s \in S$, $h(s) \leq \text{cost}_{\pi^*}$ where π^* is an optimal plan for a STRIPS problem with the same domain and goal conditions, but $I = s$.

The most popular heuristic search algorithms are best-first search, which selects the state with the smallest heuristic value

$$\operatorname{argmin}_{s \in \text{frontier}} h(s),$$

and A* search [113], which selects the state with the smallest estimated total path cost

$$\operatorname{argmin}_{s \in \text{frontier}} g(s) + h(s).$$

$g : S \rightarrow \mathbb{R}^{\geq 0}$ is a function that returns the least-cost path from I to state $s \in S$ found so far. When the heuristic is admissible, g happens to be optimal so that *the first plan found using A* search is also optimal.*

This ideal property of A* search has led to lots of research on good admissible heuristic functions because each one produces a different footpath through the state space (exploring more or less states than others). One important finding was that greater heuristic values, which are more accurate when constrained by admissibility, require exploring fewer unnecessary states. Thus it is good practice to employ several different admissible heuristic functions and take the greatest one at each state. Although having knowledge about the domain is ideal to make these more accurate estimations, domain-independent heuristics have also been discovered based on the simplified properties of logic-based representations [123, 116, 64, 30]. One of the earlier ones is derived from the method described in Section 2.2.

2.1.1 Knowledge Engineering: Expressive Factored Representations

STRIPS representations unfortunately seem restrictive to describe realistic states and actions, and many extensions have been introduced to handle these situations. The research area of *knowledge engineering* specifically focuses on identifying how to best represent scenarios for accurate, understandable, and efficient automated planning. The majority of these have been wrapped into the PDDL representation mentioned above, and its documentation [191, 75, 93] is the best resource to learn about all the features that *should be available.*

Two of the most noticeable differences are the language's use of first-order predicates and actions as well as the distinct separation of the domain and problem into

two separate files. These features enable the *domain* to be a representation of the world in general and the *problem* to provide the specifics for this instance. That is, predicates and actions with (often typed) variables describe the form of states and transitions in the domain’s state space. This form is *grounded* into an actual state space via the problem’s list of (often typed) objects, and the problem also provides the particular search problem to perform via a particular initial state and goal condition written as conjunctions of fluents (which are grounded predicates, substituting specific objects for each variable).

The domain-independent heuristics applied to various automated planning solvers work under restrictions that only certain features are used. Without such conditions, the approximation methods might become inaccurate (including losing admissibility) or incomputable¹. Despite such limitations, it is actually possible to revise many of the features into a STRIPS representation. So the ones that are not supported can be rewritten by the knowledge engineer to still use the software of choice. We introduce only a few of the features below, particularly those that will be discussed throughout this dissertation.

Negative preconditions address a seemingly overlooked case that would seem common in representing domains. An action $a \in A$ can set fluents to be false via delete effects, but preconditions can only list what *must be true about the world*. As the name implies, this simply allows us to also have two precondition sets $pre_a, \neg pre_a \subseteq F$ such that a is applicable if and only if both $pre_a \subseteq s \in S$ and $\neg pre_a \cap s = \emptyset$. That is, everything in pre_a must be true in state s while everything in $\neg pre_a$ must be false in state s .

¹PDDL has a ‘:requirements’ keyword for this reason—the knowledge engineer should acknowledge the features used, and the solver should politely reject the domain-problem pair if it cannot support any of the listed features.

To convert negative preconditions into STRIPS, we must introduce a new fluent $\neg f$ for each fluent $f \in F$ where $\neg f$ means “the opposite of f ”. Then we can use the appropriate fluent, either f or $\neg f$, as an element of pre_a . It is important that the add and delete effects maintain consistency in this case: $add_a = add_a \cup \neg del_a$ and $del_a = del_a \cup \neg add_a$ where $\neg X$ is the set $\{\neg f \mid f \in X\}$. This is not efficient for the state space representation because f and $\neg f$ can never have the same truth value at the same time, but states where both are true and false simultaneously will exist in the space. Luckily, most search algorithms generate states as they are explored, and these contradictory states should never be reachable from any realistic initial state using any action.

ADL (Action Description Language) [215] introduces natural consequences when actions are performed in addition to the intended effects. These are represented as a set of *conditional effects* $cond_{a \in A}$ whose elements c are of the form $P^c \rightarrow (E_{add}^c \wedge E_{del}^c)$. $P^c \subseteq F$ is a set of preconditions for the condition and $E^c \subseteq P$ are the respective sets of effects for the condition. When action a is performed, each element of $cond_a$ with satisfied preconditions P^c will append its add and delete effects to the action’s; that is,

$$add_a = add_a \cup \bigcup_{\{c \in cond_a \mid P^c = \text{TRUE}\}} E_{add}^c \quad \text{and} \quad del_a = del_a \cup \bigcup_{\{c \in cond_a \mid P^c = \text{TRUE}\}} E_{del}^c.$$

To convert conditional effects into STRIPS, we create $2^{|cond_a|}$ versions of action a to account for whether or not each conditional effect would have been applied. The best way to do this conversion is to create two actions $a+$ (satisfied c) and $a-$ (unsatisfied c) from a single conditional effect c , remove c from $cond_{a+}$ and $cond_{a-}$, and recursively perform this process with both new actions until no conditional effects remain:

$pre_{a+} = pre_a \cup P^c$	$add_{a+} = add_a \cup E_{add}^c$	$del_{a+} = del_a \cup E_{del}^c$	$cond_{a+} = cond_{a-}$
$pre_{a-} = pre_a \cup \neg P^c$	$add_{a-} = add_a$	$del_{a-} = del_a$	$= cond_a \setminus c$

Similarly, ADL allows goal conditions to be conditional via set G_{cond} with elements c of the form $P^c \rightarrow (E^c \wedge E_{\neg}^c)$. Then a goal state $s \in S$ not only satisfies $G \subseteq s$, but also has to satisfy $E^c \in s$ and $E_{\neg}^c \cap s = \emptyset$ for each c such that $P^c \in s$. Conditional goal conditions cannot be converted to STRIPS the same way as conditional effects because only one goal can guide the heuristic search (see Section 6.5.1 to see how the lesser-known multiple goal heuristic search algorithm actually can do this). Instead, one must create a new fluent *done*; create an action *goal* where $pre_{goal} = G$, $cond_{goal} = G_{cond}$, $add_{goal} = \{done\}$, and $del_{goal} = \emptyset$; and change the goal condition to $G' = \{done\}$. Then this new action with conditional effects is converted to STRIPS.

The more unique feature that ADL introduced is the use of logic quantifiers for preconditions and goal conditions. The application of first-order and higher-order logic in AI was not novel due to research on automated theorem proving [45], but it was not considered in automated planning due to the separation of domains and problems. As the definition of the world, domains do not consider how many instances of each typed entity exist. So requiring an unknown number of fluents to perform an action was not possible beforehand (regarding universal quantification \forall), and disjunction was not an option for single sets of fluents (regarding existential quantification \exists). In order to make existential quantification possible, ADL also introduced the ‘or’ keyword for disjunction. Due to their functionality not being possible with STRIPS, there is unfortunately no way to convert disjunction or the quantifiers properly. The closest hack for universal quantification is to *list all the objects in the world in the domain (as constants) and enumerate the fluents via substitution for every constant*—then the domain is less generalized and the specific problems only differ in assignments to I and G .

Equality simply allows an additional precondition fluent that determines whether the constants/objects assigned to two parameters are the same. This only makes sense when parameters are used in action definitions because the truth value is otherwise known at the time of creating the domain. It is thus simple to convert equality into STRIPS when there are no parameters similar to the hack for universal quantification. Rather than enumerate the fluents, we enumerate the actions with the parameter substitutions that match the ones in the precondition.

Numeric values generalizes the factored state representation from logic-based fluents to numeric fluents². The simple approach that PDDL uses only generalizes the variables to store real numbers, but this includes the performance of arithmetic operations such as addition and multiplication. Assignment is also allowed in some automated planning software, but equality checking is usually available. Besides representing numbers in states, numeric values *enable action costs for optimality criteria*.

A variation of numeric values that is worth mentioning is the *state variable*. We do not give them their own section because they are not supported in the current version of PDDL. Instead, they are part of the SAS+ [16] representation as a replacement of logic-based fluents. The state variable is constrained to a set of values V instead of just TRUE/FALSE or the real numbers. The *domain transition graph* (DTG) is a special structure that shows how the values for a single state variable can change, and it enables a domain-independent heuristic that estimates the number of actions needed to change all the DTGs of state variables specified in the goal conditions [116, 279]. SAS+'s state variables can be converted into a STRIPS representation by creating a fluent per assignable value: ‘state variable X has value $v \in V$ ’. Assignments to a state variable are synonymous with an add effect for the assigned value and delete effects

²For those who write PDDL representations, the ‘:fluents’ keyword is used to list numeric fluents while the ‘:predicates’ keyword is used to list the logic-based fluents. It may be a bit confusing, but syntax is syntax.

for all the values not in the assignment. Because numeric values can be continuous and infinite, they cannot be converted into STRIPS unless *the closure of the values the numeric fluent may have in its lifetime is discrete and finite*. In such a case, the numeric fluent is like a SAS+ state variable.

2.1.2 Generalizing Factored Representations with Abstractions

A large benefit of factored state representations is that we can describe the important aspects of actions without overly specific state details. However, this introduces a potential for actions to fail when performed in the real world if those details actually mattered. However, automated planning at very fine-grained levels is computationally intractable due to the massive increase in the state space’s size. Motion and path planning have become their own field of study [171] to find sequences of actions that will work in the real world, and most of them focus on well-distributed sampling techniques to avoid exploring most the space.

When motion and path planning only involve navigation tasks, hierarchical search methods [32, 166] can perform an initial search over the coarse-grained state space (usually a grid with large regions) and constrain the more fine-grained state space (a grid with smaller regions) to only search states that are within the coarse-grained state space’s plan. This incrementally refines the plan to navigate efficiently around obstacles without exploring too much of the state space. Cambon and Alami introduced a similar method for performing automated planning at the high-level (factored states) and at the low-level (with the specific state details), but it can apply to non-navigation tasks [41]. The analogy to the granularity of grids is the *abstraction of state information*.

Definition 13. *An abstraction function $f : S \rightarrow X$, where X is some generic set, provides information about the input state.*

Definition 14. For some set of abstraction functions $F = \{f_1, f_2, \dots, f_n\}$, the abstracted state space is the set $S_F = \{(f_1(s), f_2(s), \dots, f_n(s)) \mid s \in S\}$.

Definition 15. For some set of abstraction functions F , the concretized state operator $[\cdot]_F : S_F \rightarrow 2^S$ returns the set of all states $s \in S$ that have the same abstract state $s' \in S_F$.

Similar to the hierarchy of grids, we can define a hierarchy of abstraction functions that produce abstracted state spaces of varying cardinality. Planning in the abstracted state space with the smallest cardinality is the most coarse-grained search, and its plan can constrain the states visited in state spaces with greater cardinality. Specifically, we can restrict the space to the union of the concretized states

$$\bigcup_{i=1}^{|\pi|} [a_i(\dots a_1(I_F) \dots)]_F \cup [I_F]_F$$

where I_F is the abstracted initial state. If each X_i is simply the set of truth values $\{\text{TRUE}, \text{FALSE}\}$, then S_F would be a STRIPS representation of the state space. Modifications are sometimes necessary for the actions' preconditions and effects, and the abstractions can still miss enough details that either the coarse-grained plan exists when no fine-grained plan exists [258] or the coarse-grained plan omits a crucial part of the concrete state space via its constraints that prevents finding the solution (a controller that updates the abstraction function set can help with this issue [255]).

When developers are producing code for their applications, though, it is not guaranteed that they will have the resources to create a factored-state representation in one of the languages for which automated planning software is developed. Likewise, their code might be too domain-dependent for a generic representation to sufficiently represent all the features. Many representation languages and automated planning programs are also not friendly to average users who lack expertise in the area, which recently revived interest in creating better user interfaces for them [77]. Without ease

of use, there is not a lot of evidence to the general public that automated planning methods can be useful AI tools for their applications.

We thus introduce the *direct implementation of abstraction functions in code* as a simple way for developers to use basic automated planning techniques. This is a by-product of the implementation of our framework discussed in Chapter 6 due to the library developed to manage the search techniques. Rather than encode it to read classical planning representations and take advantage of those techniques, we instead programmed a vanilla heuristic search where the *developer encodes the domain information using whatever representation they see fit*. In particular, for programs that have already been written, there is no requirement to re-implement the code in a unique representation language just to perform classical planning.

Without any form of abstraction, states are defined in code as classes and structures containing variables. At this finest granularity, the state space S is broadly the set of all assignable values to each variable (all possible bit combinations). The set of actions A are the methods/functions/subroutines that change the values of some or all the variables. When actions directly change one state into another state without a factored-state representation³, it is most convenient to represent the outcomes of actions via a transition function $T : S \times A \rightarrow S$. In practice, the table for T can be quite large; so it is ideal to have a collection of functions that perform fail-fast precondition checks (that specify when an action is applicable) and apply the effects. The latter are likely already implemented in the developer's code that imports our library.

Precondition checking can be directly implemented in the code without using a special representation language. First, the developers implement each abstraction

³One may argue that each variable's finite byte size makes it a state variable for the SAS+ representation. However, dynamic memory for data structures violates this unless we have a maximum size limit. The SAS+ representation can apply if the state's variables do not dynamically allocate memory.

function as a method/function/subroutine that takes a state parameter and returns the desired information. These may be as simple as calling accessor functions for some variable’s value, but they can also be more complex and interpret information based on several variables’ values. Then a precondition method/function/subroutine for some action $a \in A$ receives a state argument and evaluates a boolean statement composed of operations involving the abstraction function methods/functions/subroutines. If the implemented precondition check returns TRUE, then a is applicable and we can continue to execute the respective method/function/subroutine that is already in the code. This will modify the current state just like an action’s effect, but it applies to a transition in the fine-grained state space rather than the coarse-grained one.

Despite the effects taking place at a different level of abstraction, the updated state can now be passed into the implemented abstraction functions, *indirectly applying the effects to the coarse-grained state* without having to specify them. That is, one can effectively run automated planning algorithms without using a representation language if they implement the preconditions at a high-level (or multiple for a hierarchy of abstractions) and the effects at a low-level. Furthermore, the application of effects in the original state space avoids the issues of abstracting away important information—the plan should succeed if the code validly emulates the domain. Although this method grants access to the use of search techniques in application-driven code, domain-independent heuristics will not be available due to their uses of their respective factored representation features. Furthermore, the branching factor of available actions is at the discretion of the developer’s choice of granularity, which yields the risk of being too broad or too constrained during search.

2.2 The Planning Graph

Because classical automated planning problems using STRIPS and PDDL-like representations are spatially complex to solve (finding solutions is PSPACE-complete

[38]), one area of research within automated planning is the development of algorithms that are more efficient at finding the solution than traditional search. One such algorithm is GRAPHPLAN [27], which has since had other impacts on automated planning research besides faster algorithms. It requires a special data structure called the planning graph:

Definition 16. *The planning graph G is a directed partitioned graph representing the set of valid STRIPS plans for problem \mathcal{P} all at once, including their simultaneous conflicts. Partitions alternate between proposition layers $P_0, P_1, \dots, P_k \subseteq F$ and action layers $A_1, \dots, A_k \subseteq A$. Edges represent*

- satisfied preconditions $E_{pre} = \{(u, v) | u \in P_i, v \in A_{i+1}, u \in \text{pre}_v\}$,
- added effects $E_{add} = \{(u, v) | u \in A_i, v \in P_i, u \in \text{add}_v\}$,
- deleted effects $E_{del} = \{(u, v) | u \in A_i, v \in P_i, u \in \text{del}_v\}$,
- or mutual exclusion enforcement $E_{me} \subseteq \{(u, v) | (u, v \in P_i) \vee (u, v \in A_i)\}$.

Proposition layer P_i lists all fluents that may hold true in the i^{th} state, and action layer A_i lists all applicable actions that can be performed in the possible states of the $(i - 1)^{\text{th}}$ proposition layer. Because some actions have contradicting effects or preconditions when performed simultaneously, mutual exclusion edges denote which fluents and actions cannot be in the same plan. Two fluents f_1 and f_2 are mutually exclusive in the i^{th} layer, $me_i(f_1, f_2) = \text{TRUE}$, if one of two cases holds:

- They are negations of each other:

$$(f_1 = \neg f_2) \Rightarrow \forall i \in \{0, 1, \dots, k\}. me_i(f_1, f_2) = \text{TRUE}$$

- Every action whose effect is f_1 is mutually exclusive to every action whose effect is f_2 :

$$(\forall a_1, a_2 \in A_i. (me_i(a_1, a_2) = \text{TRUE}) \wedge f_1 \in (add_{a_1} \cup del_{a_1}) \wedge f_2 \in (add_{a_2} \cup del_{a_2})) \Rightarrow me_i(f_1, f_2) = \text{TRUE}$$

while two actions a_1 and a_2 are mutually exclusive in the i^{th} layer, $me_i(a_1, a_2) = \text{TRUE}$, if one of three cases holds:

- Their effects negate each other (that is, a_1 adds some fluent that a_2 deletes, or each action adds/deletes some fluent and its negation):

$$(\exists f \in F. (f \in add_{a_1} \wedge f \in del_{a_2}) \vee (f \in add_{a_1} \wedge \neg f \in add_{a_2}) \vee (f \in del_{a_1} \wedge \neg f \in del_{a_2})) \Rightarrow \forall i \in \{1, \dots, k\}. me_i(a_1, a_2) = \text{TRUE}$$

- a_1 has an effect that will not satisfy some precondition of a_2 :

$$(del_{a_1} \cap pre_{a_2} \neq \emptyset) \Rightarrow \forall i \in \{1, \dots, k\}. me_i(a_1, a_2) = \text{TRUE}$$

- There exists some precondition of a_1 that is mutually exclusive to some precondition of a_2 :

$$(\exists f_1, f_2 \in P_{i-1}. (me_{i-1}(f_1, f_2) = \text{TRUE}) \wedge f_1 \in pre_{a_1} \wedge f_2 \in pre_{a_2}) \Rightarrow me_i(a_1, a_2) = \text{TRUE}$$

Beginning with the initial state as the first layer $P_0 = I$, GRAPHPLAN procedurally generates the planning graph until a solution is found. The forward step produces layers A_{i+1} and P_{i+1} based on the preconditions and effects of the respective previous

layer, including mutual exclusion edges. The backwards step is not performed until some layer P_k contains all the goal conditions without mutual exclusion. Once such a layer is generated and it is thus possible to create a goal state, the backwards step searches for a set of paths from each goal condition in P_k to any predicate in P_0 . For the actions along the set of paths to represent a valid plan, no two paths can pass through mutually exclusive fluents or actions. Unlike the sequential plans found using heuristic search, a plan’s actions can be performed simultaneously if they are in the same layer of the planning graph because they are not mutually exclusive; thus there is no conflict between their preconditions and effects.

Relaxations to GRAPHPLAN, such as removing E_{me} , produce admissible heuristics like fast-forward [123], which was one of the first *domain-independent* admissible heuristics. The intuition is that without mutual exclusion, only the forward step is necessary until layer P_k contains all the goal conditions. Then k is the fewest number of possible actions that can be taken to solve the task from the state that layer P_0 represents. If P_k ’s goal conditions happen to not be mutually exclusive and a path existed from each one to P_0 , then the heuristic is exact when each layer only contains one action from the plan.

2.3 Hierarchical Task Networks

PDDL, even with all its extensions (including the ones not described above) to STRIPS, cannot represent every solvable problem in a way that we can efficiently find a solution. The plans that solve STRIPS problems are linear sequences, which are a recovered trace of the path in search space. Even if GRAPHPLAN finds parallel components, all the actions in a single layer must be performed before proceeding to the next layer’s actions. This is called *partially-ordered*, but is still linear with respect to the found path. Despite these linear solutions, the search process is often exponential with respect to the number of applicable actions per state because they

each need to be considered. If we have some information about the tasks, though, then we can provide additional information in the task representations that help to find quicker solutions.

Definition 17. *Like a context-free grammar, tasks can be terminal or non-terminal. A non-terminal task T can be decomposed into a sequence of tasks that, if solved, also solve T . A terminal task t is atomic and cannot be decomposed into equivalent task sequences.*

Definition 18. *A task network is a tree where a parent task can decompose into its children tasks. Thus only leaves of a task network can be terminal tasks.*

Definition 19. *A hierarchical task network (HTN) [70] is a tuple $\mathcal{H} = (\mathcal{T}_n, \mathcal{T}_t, \delta)$ where \mathcal{T}_n is a set of non-terminal tasks, \mathcal{T}_t is a set of terminal tasks, and $\delta : \mathcal{T}_n \rightarrow 2^{(\mathcal{T}_n \cup \mathcal{T}_t)^*}$ is a function that maps a non-terminal task to its decompositions, which are sequences of tasks⁴.*

Definition 20. *A HTN Problem is a tuple $\mathcal{P} = (\mathcal{H}, T_\top)$ where \mathcal{H} is a HTN and $T_\top \in (\mathcal{T}_n \cup \mathcal{T}_t)$ is the start symbol that describes the overall task that must be completed. This generates an initial task network whose root (and only leaf) is T_\top .*

HTN representations are similar to context-free grammars, which is one of the reasons why it has been the favored representation choice for many recognition algorithms (see Section 3.1.2). Using the decomposition function δ , which returns all the possible decompositions, automated planning chooses some non-terminal task at the current task network’s leaves and one of its decompositions. This updates the current task network incrementally until the process creates a *fully-decomposed* task network whose leaves are all indivisible terminal tasks. These terminal tasks, which

⁴* is the Kleene closure, which is the set of all sequences of 0 or more elements from the closed set. The power set of a Kleene-closed set thus represents a set of sequences.

may have ordering constraints dictated via the decomposed sequences, form the plan that accomplishes T_{\top} . Depending on the decomposition function’s design, recursion is possible to perform a terminal task indefinitely until a condition is met; for example, taking individual steps (terminal task) until successfully moving from one location to another (non-terminal task).

Another benefit of HTNs are their simpler intuition to people, who are typically familiar with the notion of breaking a task down into subtasks. As robots are becoming more ubiquitous, there have even been studies on developing software that can help guide average users as they define HTNs for their personal uses of their robots [197]).

2.3.1 Using HTNs with State-Space Search

The decomposition function δ provides a breakdown of tasks that can constrain the applicable actions considered from a state during search. To integrate HTNs with factor-based representations (not just STRIPS, but PDDL, SAS+, etc.) as a constraint on the search process, it is often the case that $\mathcal{T}_t = A$ so that the plan is the (sometimes partially-ordered) sequence of leaf nodes of the fully-decomposed task network. It is then also important that there is some non-terminal task that can represent the set of goal conditions G .

The advantage of employing a HTN to a state-space search is that *the HTN prunes the branching factor to avoid exploring unrelated regions of the state space*. Even with admissible heuristics, many states will be explored because their underestimated path costs are similar—optimality is only guaranteed when all states through paths of lesser estimated cost are considered. Thus the HTN *guides the search process using domain knowledge* about the problem’s goal rather than blindly relying on a heuristic value. For example, why considering peeling a banana when the task is to pour a glass of

water? (If the banana is encountered on the way to the sink, then blind state-space search will consider this direction just in case it results in a poured glass of water).

With this integration of representations, the application of actions only happens when the task network decomposes to a terminal task/action. The precondition and effects of each action still apply to the current state, which might also impact how automated planners select task decompositions. The preconditions and effects of actions further introduce ordering constraints between them to avoid conflicts, similar to mutual exclusion issues for GRAPHPLAN. In fact, the Simple Hierarchical Ordered Planner (SHOP [206], SHOP2 [205], and SHOP3 [97]) software’s representation allows precondition definitions for non-terminal tasks to guide the decomposition process at earlier stages. Technically, if one was to iterate through every non-terminal tasks’ decompositions and find a *consistent set of preconditions (of the first action in the sequence) and effects (of the last action in the sequence) between every fully-decomposed task network*, then those could be applied to the said non-terminal task as its own preconditions and effects.

2.4 Allowing Uncertainty in Automated Planning

Several assumptions made for classical automated planning removed sources of uncertainty because of the complication they bring to the decision making process. Each one can be addressed either *qualitatively* or *quantitatively*, depending on whether the uncertainty is measured by probability. When uncertainty is not measured (qualitative), it is usually represented as a set of all the possibilities and the objective is to still *accomplish the task with respect to anything that can happen*. When uncertainty is measured (quantitative), the elements of these sets have probability mass for how often they will occur and the objective is to *accomplish the task with as great a probability as possible*.

Non-deterministic action outcomes Determinism guarantees that a plan will always proceed as expected. Without this assumption, nuances in the agent’s or environment’s physics could intervene such as dropping something when trying to pick it up. Fully-observable non-deterministic (FOND) [170, 85] automated planning removes this limitation qualitatively, listing the possible outcomes per action, while Markov decision processes (MDPs) [21] remove the limitation quantitatively with probabilities assigned to each outcome. FOND representations are similar to PDDL, but disjunctions are allowed in the effects. MDPs can use a factored representation such as probabilistic PDDL (PPDDL) [293] or Relational Dynamic Influence Diagram Language (RDDL) [238], but are often defined without factored states:

Definition 21. A MDP is a tuple $\mathcal{M} = (S, A, T, R)$ where S is the set of all states, A is the set of all actions, $T : S \times A \rightarrow \mathcal{S}^{|S|}$ is the transition function, and $R : S \rightarrow \mathbb{R}$ is the reward function. The transition function produces a distribution over the state space ($\mathcal{S}^k = \left\{ p \in [0, 1]^k \mid \sum_{i=1}^k p_i = 1 \right\}$ is the k -dimensional simplex) such that an agent in state $s \in S$ that performs action $a \in A$ will transition to state $s_i \in S$ with probability equal to the i^{th} element of $T(s, a)$. If a is not applicable in s , then performing a self-transitions to s with probability 1.

The solutions to FOND problems extend plans with if/then branches called contingencies, assigning actions for all the states that might result from each outcome. On the other hand, solutions to MDPs further extend this assignment of an action to every reachable state in the state space from the initial state, regardless of how likely it will be encountered⁵.

Definition 22. A contingent plan π is a directed tree with nodes $a_{i,j} \in A \cup \{\text{NIL}\}$ and labeled edges $s_i \in S$ with the following conditions:

⁵Reinforcement learning studies ways to approximate these solutions via ‘trial and error’. This is often quicker than exact computation and can be reasonably accurate, depending on the domain’s structure and distribution over action outcomes.

- The root node's action $a_{0,0}$ is applicable in the initial state I .
- Each non-root node's action $a_{i,j}$ is applicable to the state s_i that labels its incoming edge.
- All outgoing edges from a non-leaf node's action $a_{i,j}$ are distinctly labeled, one per possible state that can result from performing $a_{i,j}$ in the state s_i that labels its incoming edge (or the initial state I if $a_{0,0}$).
- NIL is assigned to a leaf node $a_{i,j}$ if and only if there does not exist a plan that solves planning problem (\mathcal{D}, s_i, G) where s_i is the state that labels $a_{i,j}$'s incoming edge.

This implies that for any path from the root node to a leaf node $a_{0,0}, a_{i_1,j_1}, \dots, a_{i_N,j_N}$, each node's action is applicable when performed in the path's order

$$pre_{a_{i_n,j_n}} \subseteq a_{i_{n-1},j_{n-1}}(a_{i_{n-2},j_{n-2}}(\dots a_{0,0}(I)\dots))$$

for all n from 1 to N and the resulting state after completing all the actions in order satisfies the goal conditions

$$G \subseteq a_{i_N,j_N}(a_{i_{N-1},j_{N-1}}(\dots a_{i_1,j_1}(a_{0,0}(I))\dots)).$$

Definition 23. A policy is a function $\pi : S \rightarrow A$ that instructs what action to perform at each state in the state space.

Definition 24. The expected reward of a policy with respect to MDP \mathcal{M} at state $s \in S$ is

$$\mathbb{E}_{\mathcal{M}}^{\pi}[s] = R(s) + \sum_{s_i \in S} T(s, \pi(s_i))_i \cdot \mathbb{E}_{\mathcal{M}}^{\pi}[s_i].$$

This value can be computed iteratively using the Bellman update [21].

Definition 25. *The optimal policy π^* for MDP \mathcal{M} is one whose mapped actions maximize the expected reward at each state. That is, for all possible policies π' that apply to \mathcal{M} , $\mathbb{E}_{\mathcal{M}}^{\pi^*}[s] \geq \mathbb{E}_{\mathcal{M}}^{\pi'}[s]$ for all $s \in S$.*

Most the time, these actions try to return the agent to a state along the optimal plan’s path if an action’s outcome drives the agent ‘off course’. The reward function can still heavily influence a policy to prefer unlikely outcomes from actions (such as playing the lottery), which requires the domain engineer to be careful when defining it [9]. However, relying on people to provide good reward functions has also introduced issues such as giving inconsistent rewards based on the novelty of the situation (stop rewarding after the agent does what is expected) [130] or agent’s current behavior (punishing for not doing a ‘good thing’ after being ‘good’, but rewarding for not doing a ‘bad thing’ after being ‘bad’, even if these ‘things’ are the same) [183].

Partial observability Fully observable environments avoided hidden information. This guarantees that the agent always knows in which state it is so that a plan can be computed. If any information about the state is missing, then more actions might be applicable than perceived. This can lead to not finding a solution from the currently perceived state even though one exists. If the closed-world assumption is applied and negative preconditions are allowed, then the opposite problem can also happen where an applicable action cannot really be performed because an unobserved fluent is true (and the precondition requires it to be false).

The majority of the research studies partial observability together with non-deterministic action outcomes. Partially-observable non-deterministic (POND) [24] automated planning removes these limitations qualitatively, listing both the possible initial states and the possible outcomes per action; conformant planning uses planning graphs to solve these problems [250]. Partially-observable MDPs (POMDPs) [14, 253] remove these limitations quantitatively with probabilities assigned over the initial

state (called an *initial belief state*) and to each outcome. POND representations are again similar to PDDL, but disjunctions are allowed in the effects and initial state. POMDPs can use the same factored representations as MDPs, but are often defined without factored states just like MDPs. To help orient the agent, each transition also produces an *observation symbol*. These observation symbols allow the agent to reason over the current and/or previous state.

Definition 26. A POMDP is a tuple $\mathcal{P} = (S, A, T, R, \Omega, O)$ where S is the set of all states, A is the set of all actions, $T : S \times A \rightarrow \mathcal{S}^{|S|}$ is the transition function, $R : S \rightarrow \mathbb{R}$ is the reward function, Ω is the set of possible observation symbols, and $O : S \times A \rightarrow \mathcal{S}^{|\Omega|}$ is the conditional probability distribution over observation symbols. Given the performed action $a \in A$ and resulting state $a(s \in S) \sim T(a, s)$ (sampled from the transition function), the probability of observing some $o \in \Omega$ is $O(a(s), a) = P(o | a(s), a)$.

2.5 Applications of Automated Planning in Videogames

Although many videogames rely on expert systems for non-playable characters (NPCs) to meet the real-time demands of nearly sixty frames-per-second processing and rendering without lag, planning has been used in quite a few game-related applications. The most typical case is character navigation via motion planning. A unique advantage for motion planning in the majority of videogames is that the levels and maps are designed before the games' release. Thus it is possible to precompute more complicated heuristic values [225, 125].

Besides motion planning tasks, the game FEAR [214] developed its own representation for AI-controlled agents based on high-level planning representations such as STRIPS and PDDL. Kelly, Botea, and Koenig [153] used HTNs to create unique schedules for each NPC to follow. The schedules were generated off-line and used like expert systems, but the autonomous generation is still noteworthy given that

most NPC behavior is hand-coded by the game developers. A planning formulation was also used to create actions as game mechanics given an initial configuration and completed configuration of a level [301]. This enables content creators to receive suggestions based on what the player-controlled characters need to do in simulation to complete the level.

2.6 Concluding Remarks: Evolution of Automated Planning

The ability to make decisions is far more complex than the founding scientists of AI expected. While the ability to solve problems is as simple as searching, the challenge lies in efficiently searching for the solution. People are able to identify what features of the environment matter for solving their task and performing actions. They can also simulate a few steps in their head and quickly abandon options that do not appear successful. Heuristics and state space pruning techniques still have a lot of research to go before they can be useful for larger problems that are more realistic. The current methods require special representation choices for their algorithms to work, and real-world applications cannot always comply with the assumptions. We introduced one way that application developers can adapt their code to use some of the search techniques, but the developers still need to use their domain expertise in order to implement their own heuristic and pruning shortcuts for efficiency.

Efficiency techniques are still a major challenge for automated planning under uncertainty as well. Besides modified search techniques and approximations via reinforcement learning, one of the popular techniques for solving MDPs is *determinization* [292], which solves a collection of classical planning problems without the probabilities as an approximation of the optimal policy. Likewise, POMDPs can map into belief MDPs that encode the set of belief states as the observable belief state [141]. Thus the best ways we can currently address complicated decision making problems

is to turn them into simpler ones that we can solve, even if we cannot solve those simpler problems well.

CHAPTER 3

PLAN, ACTIVITY, AND INTENT RECOGNITION

How many of you have walked into a room [all the students in the room raise their hand in excitement], saw someone, and wondered, “what is that person doing?” [several hands go down].

— Richard (Rick) G. Freedman,
explaining his research to a classroom of third-grade students

Several decades since the establishment of automated planning as an area of study, the AI community has branched into various specialized communities that pursue specific challenges that have been identified within the scope of AI. Quite a few of them began to focus on perception tasks, and the majority of the early research on perception was strictly interpreting sensor data. With much inspiration from the application of intelligent robotics, this led to a lot of research within areas such as computer vision [188] and tactile sensing [112]. However, perception involving higher-level thought was eventually realized in story analysis [150] and understanding user activities within program applications [184].

The most distinguishing feature between these two forms of perception is the *observation focus*. The former, lower-level perception aims to obtain an overall environment description; this creates a world state in which the perceiving agent can act. The latter, higher-level perception is associated with understanding agents and their underlying decision making processes. The focus on observing other agents independently of the environment itself adds a “black box” around their actions that must be interpreted in addition to rules of the environment (such as physics), which all affect how the perceiving agent can act in the world. The formal establishment of

the plan, activity, and intent recognition (PAIR) community studying this happened within the past decade [96] following a short series of workshops originally titled “Modeling Others from Observations” (MOO). The workshop series was eventually renamed PAIR due to the common methods and themes in presented research, but there are still discrepancies regarding what PAIR specifically studies in each type of recognition.

One goal of this dissertation is to begin addressing these discrepancies and *investigate how all the types of recognition are interrelated*. Despite their common origin and being discussed at the same venues, they are not often discussed with respect to the other types of recognition (though a recent compendium does mention in the preface that plan and intent recognition are strongly related [263]). Each form of recognition is considered to be different to some degree, but these degrees have not been formally described outside qualitative descriptions.

Throughout this chapter, we will introduce each type of recognition in PAIR and investigate some common challenges and frameworks with respect to their specific perspectives on the following generalized formal definitions:

Definition 27. *A recognition problem is a tuple $\mathcal{R} = (\mathcal{KB}, \mathcal{O}, \mathcal{H})$ where \mathcal{KB} is a knowledge base representing what the observing agent knows (and thus what information is at its disposal), \mathcal{O} provides a sequence of observations, and \mathcal{H} lists the possible hypotheses that can be recognized.*

Many recognition algorithms, despite the type and challenge, broadly resemble *matching* \mathcal{O} to some element of \mathcal{H} based on \mathcal{KB} . \mathcal{KB} and \mathcal{H} are often intertwined such that the question “is $h \in \mathcal{H}$ the correct match?” can be answered using a portion of \mathcal{KB} that relates to just h . \mathcal{KB} is specifically defined with respect to the observing agent that performs recognition, R_{ing} . Any information regarding the observed agent that is recognized, R_{ed} , is only in \mathcal{KB} if R_{ing} knows or assumes it.

3.1 Plan Recognition

Plan recognition is viewed as an inverse problem to automated planning because the observation sequence is composed of executable actions. That is, each $o \in \mathcal{O}$ often satisfies $o \in A$. However, \mathcal{O} is not often complete; R_{ing} might ‘blink’, a certain action might not be observable with the available sensors, or a transcription might only point out ‘key actions’ rather than provide a ‘play-by-play’ summary. Thus the primary challenge in plan recognition is to answer the question, “what is the agent doing *overall*?”. \mathcal{H} is usually a set of plans or high-level tasks. However, the answer to the question is paired with the secondary challenge that has received more emphasis lately [91, 195]: “why is this the correct answer?”

Definition 28. *An explanation is a plan or policy π that best resembles \mathcal{O} ’s sequence and justifies why some $h \in \mathcal{H}$ is the answer to plan recognition problem \mathcal{R} .*

Depending on \mathcal{KB} , the explanation may have contingencies, be a task network, or more. Overall, the structure of, and information in, the knowledge base has the most impact on the different plan recognition algorithms. Two common knowledge bases are based on how much R_{ing} knows about R_{ed} and the world in which they are acting.

Definition 29. *A plan library is a type of knowledge base that contains precomputed plans for solving some set of automated planning problems. This can include a grammar that constructs plans without solving the automated planning problems.*

Definition 30. *A planning domain is a type of knowledge base that models the world in which the agents act. This is identical to Definition 2, but is now a tool for “thinking in another agent’s shoes” rather than personal decision making.*

The knowledge base was partitioned into subsets of matching plans for each possible hypothesis in earlier research, which became plan libraries [149]. While libraries are still used and have their advantages, knowledge bases are more expressive and

generalized when the planning domain itself is provided [226]. In particular, \mathcal{H} can change independently of updating \mathcal{KB} as long as novel hypotheses are solvable in the original domain. However, we demonstrate in Section 6.5.2.2 that there is more overlap between plan libraries and planning domains than people might expect, assuming that \mathcal{H} is a constant set of goal conditions (which often holds in present recognition approaches, even if hypotheses are incrementally pruned [281, 246]).

Unlike automated planning, it is more difficult to define the case where there is no solution in \mathcal{H} . Some algorithms return a distribution over \mathcal{H} rather than returning a subset of elements in \mathcal{H} [227], but this is still a relative comparison that implies more ambiguity as the distribution over some subset of \mathcal{H} becomes uniform. In contrast to the adage “a picture may be worth one thousand words,” they are not always the correct ones. The spread metric $S\%$ [227] and quality metric $Q\%$ [68] have been proposed to quantify the number of most-likely hypotheses in

$$\mathcal{H}\% = \{h \in \mathcal{H} \mid P(h | \mathcal{O}) \text{ is at the \% percentile or greater}\}$$

and how often $\mathcal{H}\%$ contains the correct answer respectively. Ideally, for greater values of $\%$, $Q\%$ will be large and $S\%$ will be small.

3.1.1 Abduction

The earliest work in plan recognition focused on identifying the task from a sequence of descriptive observations rather than just observed symbols/actions \mathcal{O} [150]. These particular works focused on an application that has since become its own research area called *semantic understanding*, the process of identifying the meaning of natural language artifacts such as dialogue and text documents (present day research in semantic understanding has a heavy focus on search engines). The descriptive observations can be converted into a logical representation with respect to the current state of the world, and each action $a \in A$ in a logic-based factored representation can

be converted to a set of logical derivation rules. In relation to a planning problem with tuple (\mathcal{D}, I, G) , we find that:

- The observations $o_i \in \mathcal{O}$ are simply a set of fluents $o_i \subseteq F$ similar to a state representation. Like the set of goal conditions, we are not guaranteed full observability of the state so that the observations are just some features of the current state.
- The hypotheses are the set of all *initial states* that can reach the set of observations

$$\begin{aligned} \mathcal{H} = & \{s \in S \mid \\ & \exists [a_1, a_2, \dots, a_n \in A]. \forall [o \in \mathcal{O}]. \exists [k \leq n]. a_k(\dots(a_1(s))\dots) \supseteq o\}, \end{aligned}$$

which captures the notion that the initial state contains information about a task that must be performed. This appears to be counter-intuitive with present-day definitions due to their emphasis on recognition corresponding to the goal state and conditions, but *information about the task is usually removed from the state definition in present-day definitions*. That is, a feature of such an initial state was often of the form “ R_{ed} decides to do task t .” Because many states in a state space eventually become reachable, a predefined set of possible initial states serve as the plan library $\mathcal{L} \subseteq S$ where each state is assigned a task label. Then we can restrict $\mathcal{H} \subseteq \mathcal{L}$ compared to the broad definition above.

- The knowledge base is composed of logical derivation rules from all the actions

$$\mathcal{KB} = \left\{ \bigwedge_{i \in pre_a} i \rightarrow \left(\bigwedge_{i \in add_a} i \wedge \bigwedge_{i \in del_a} \neg i \right) \middle| a \in A \right\}.$$

In particular, if the preconditions of an action are satisfied, then its add and delete effects may be applied. As it is easy to convert any conditional logic

statement into a *set* of Horn clauses, we will assume that \mathcal{KB} is a set of Horn clauses for the remainder of this section.

Definition 31. A Horn clause is a conditional logic statement of the form $p_1 \wedge p_2 \wedge \dots \wedge p_y \rightarrow q$. Written as a disjunction, the clause only contains one positive literal, q , and all p_i literals are negative: $\neg p_1 \vee \neg p_2 \vee \dots \vee \neg p_y \vee q$. For Horn clause r , we define function $L(r) = \{\neg p_1, \neg p_2, \dots, \neg p_y, q\}$ to decompose r into the set of its literals when written as a disjunction.

Due to the initial natural language applications, the STRIPS notation was limited to define actions' preconditions as entity resolution bindings (often written as 'is' or '=' with a variable name and a dictionary definition of the form WORD#) and atomic actions whose parameters are arguments. These actions usually had no effects ($\forall a \in A \text{ } add_a = del_a = \emptyset$), which creates a structure Kautz formally defines as an *event hierarchy* [150] similar to the grammatical structure used for parsing approaches in Section 3.1.2. One benefit of this hierarchical design is that we can observe both actions and some environmental features simultaneously. F may then be viewed as a union of two sets of first-order logic fluents: *dict*, a set of entity resolution bindings to identify objects in the environment, and *acts*, a set of atomic actions that cannot be broken down further ($acts \cap A = \emptyset$). Then we use knowledge engineering to create a different initial knowledge base \mathcal{KB}' that formally defines relations between elements of *dict* and *acts*—this approach was commonly done by actual humans since computationally-efficient automated annotation systems did not yet exist.

Given this hand-crafted knowledge base and logic rules \mathcal{KB}_H allocating the non-atomic actions in A to labeled tasks that R_{ed} can select in \mathcal{H}^1 , we attempt to explain some \mathcal{O} using *abduction*. Much like theorem-proving, abduction is the iterative appli-

¹These logic rules have the form $h \rightarrow (\bigvee_{a \in A_h} a)$ for each $h \in \mathcal{H}$ where A_h is the set of actions involved in the plan library's plans that perform the task R_{ed} selects in h . We can also convert each of these rules into a *set* of Horn clauses when constructing \mathcal{KB}_H .

cation of backwards-derivations to find elements of \mathcal{H} that ‘prove’ \mathcal{O} . We combine all the knowledge bases together to define the set of rules $\mathcal{KB}^{\cup} = \mathcal{KB} \cup \mathcal{KB}' \cup \mathcal{KB}_{\mathcal{H}}$. For a set ξ , $C(\xi) = \bigwedge_{i \in \xi} i$ and $D(\xi) = \bigvee_{i \in \xi} i$ are treated as a respective conjunction and disjunction of ξ ’s elements in logical evaluations below. Starting with the set of observed fluents as $O^0 = \bigcup_{O \in \mathcal{O}} \bigcup_{o \in O} o$, we find rules $R^i \subseteq \mathcal{KB}^{\cup}$ that overlap with at least one observation $o \in O^i$ (overlap with a Horn clause means o is one of the p literals in r , which negates them when written as a disjunction) and do not yield a contradiction \perp :

$$R^i = \{r \in \mathcal{KB}^{\cup} \mid (\exists [o \in O^i] . \neg o \in L(r)) \wedge (C(O^i) \wedge D(L(r)) \neq \perp)\}.$$

We then conjunct one such rule $r \in R^i$ to get a new unexplained observation set

$$O^{i+1} = C^{-1}(C(O^i) \wedge D(L(r))).$$

The new elements $O^{i+1} - O^i$ increase the set of observations to include some ‘assumptions’ that must later be proven true for the conjuncted rule to hold while the removed elements $O^i - O^{i+1}$ are the observations and assumptions that r explains. \mathcal{H} is regarded as the set of axioms that are true by default such that they do not need to be satisfied. This axiomatic assumption comes from the concept that we want to recognize the underlying tasks that are the *cause* of the observations. This process ends in success if we eventually derive some $O^x \subseteq \mathcal{H}$, which means only the axiomized hypotheses remain unexplained. This process ends in failure if we can no longer find any rules R^x to apply to some $O^x \not\subseteq \mathcal{H}$ even after backtracking to try different selections from each $R^{0 \leq i \leq x}$. In the case of success, the set of recognized tasks is $R = C(O^x \cap \mathcal{H})$. For entity resolution to hold in first-order logic representations, other processes such as Skolemization will also need to be performed throughout the abduction.

One difference between abduction for plan recognition and theorem proving is the uniqueness property. Theorem proving is complete once success occurs because the observations (formally called a query) are confirmed, which is sufficient to prove a theorem. The argument/explanation itself does not matter as long as it is valid. On the other hand, each possible derivation in plan recognition could yield different O^x from various choices for each $R^{0 \leq i \leq x}$, presenting ambiguity in what was recognized. Thus we need to select the *best explanation* for the observations and choose its respective O^x as the set of recognized tasks. Multiple explanations are obtainable through either (1) backtracking and storing each success or (2) simultaneously storing every sequence of rules that do not yield a contradiction:

$$O_{r_0, r_1, \dots, r_i}^{i+1} = C^{-1} \left(C \left(O_{r_0, r_1, \dots, r_{i-1}}^i \right) \wedge D(L(r_i)) \right) \text{ for each } r_j \in R_{r_0, r_1, \dots, r_{j-1}}^j.$$

The latter approach introduces nesting so that each set can conjunct with a different sequence of rules in future iterations (hence the new subscripts). It becomes easier to manage this information as an iteratively expanding graph $G_i = (V_i, E_i)$ with vertex set

$$V_i = \bigcup_{j=0}^i \bigcup_{r_0 \in R^0} \cdots \bigcup_{r_{j-1} \in R_{r_0, r_1, \dots, r_{j-2}}^{j-1}} O_{r_0, \dots, r_{j-1}}^j$$

for the individual fluents/hypotheses involved in some derived conjunction after applying i rules to explain \mathcal{O} and directed edge set

$$\begin{aligned} E_i = & \{ v_1 \rightarrow v_2 \mid (v_1, v_2 \in V_i) \wedge \exists [0 \leq j < i] . \exists [r_0 \in R^0] . \exists [r_1 \in R_{r_0}^1] . \\ & \cdots \exists [r_{j-1} \in R_{r_0, \dots, r_{j-2}}^{j-1}] . [r_j \in R_{r_0, \dots, r_{j-1}}^j] . (\neg v_1, v_2 \in L(r_j)) \wedge \\ & (v_1 \in O_{r_0, \dots, r_{j-1}}^j \wedge v_2 \notin O_{r_0, \dots, r_{j-1}}^j \wedge v_1 \notin O_{r_0, \dots, r_{j-1}, r_j}^{j+1} \wedge v_2 \in O_{r_0, \dots, r_{j-1}, r_j}^{j+1}) \} \end{aligned}$$

for the explanation relations between derived fluents and hypotheses in V_i . Combining graph-matching with *Occam's Razor*, Kautz [150] proposed creating a set of graphs reminiscent of the plan library via forward derivations from combinations of elements

in \mathcal{H} , finding matches between these generated plans and G_x (or $G_{i < x}$ if incrementally applying process of elimination), and then selecting the one with the *fewest hypotheses used* (or more relatable ones in the case of a tie).

Charniak and Goldman [47] extended this to a more probabilistic approach that infers the probability of each hypothesis node in G_i by converting G_i into a Bayesian network. This conversion uses a similar graph (though its generation applies the more advanced pruning techniques that Kautz mentions for removing unnecessary backwards derivations) with the addition of entity resolution nodes and conditional probability tables for each node. Each hypothesis in the graph $V_i \cap \mathcal{H}$ must be a root of the network so that it is assigned a prior based on the proportion of plans it has in the plan library. Then the other nodes in $A \cup F$ have conditional probabilities defined by their parent nodes' priors based on the rules in \mathcal{KB}' . For the disjuncted branches where multiple explanations are possible, a noisy-or node is added with a conditional probability table that is set to $(1 - \alpha)$ when at least one parent is true and α when no parents are true. Nodes are assigned evidence representing observed fluents in O^0 , and entity resolution nodes that must be true are also assigned evidence. The parent and child nodes in $acts \cup \mathcal{H}$ must use the same object in *dict* for a specific parameter. This Bayesian network for plan recognition inference appears to have much in common with Charniak and Shimony's Bayesian network designed for cost-based abduction [48] except for the truth assignments and replacement of AND nodes since \mathcal{KB} 's rules now create specific edges. They also allowed 'definite true/false' and 'undetermined' as assignable evidence and then used probabilities for true/false assumptions.

An alternative approach for handling the ambiguity of multiple explanations was to use weights specified by the knowledge base's rules [119]. These weights are relatively determined for each rule in the form of a conditional logic statement (and Horn clause) $(p_1 \wedge p_2 \wedge \dots \wedge p_y) \rightarrow q$, which is interpreted as a conjunction of facts

$p_1, \dots, p_y \in F \cup A$ that can explain claim $q \in F \cup A$. Then the factors determining the weights c of each rule r , $c(r) \in \mathbb{R}^{\geq 0}$, are as follows:

- Assume that q has weight $c(q) = 1$.
- If the explanation of facts is considered easier to believe than the claim itself, then it must be the case that $c(p_1 \wedge p_2 \wedge \dots \wedge p_y) < c(q)$. Otherwise, $c(p_1 \wedge p_2 \wedge \dots \wedge p_y) > c(q)$. The latter is more common due to Occam's Razor preferring the simpler explanation.
- It must hold that $\sum_{j=1}^y c(p_j) = c(p_1 \wedge p_2 \wedge \dots \wedge p_y)$. Thus for each fact in the explanation, its individual weight must be proportional to its contribution towards the conjuncted explanation. This effectively lets $\sum_{j=1}^y c(p_j)$ serve as a normalization constant if we have to scale $c(p_1 \wedge p_2 \wedge \dots \wedge p_y)$.

With respect to the abduction process, these notions try to find the derivation with the cheapest hypothesis $\min_{h \in \mathcal{H}} \sum_{p \in h} c(p)$. Each hypothesis is simply any possible explanation for \mathcal{O} composed of i rules in \mathcal{KB} : $O_{r_0, \dots, r_{i-1}}^i = h_{r_0, \dots, r_{i-1}}^i \in \mathcal{H}$, which claims that everything is seen/explained without further justification than the i rules.

Each clause in the initial observation's conjunction O^0 is assigned an independent weight for its believability without any explanation. When performing the backwards derivations at each step, the replaced claim distributes the cost over the claims in the explanation:

$$\sum_{p \in h_{r_0, \dots, r_{i-1}}^i} c(p) = \left(\sum_{p \in h_{r_0, \dots, r_{i-2}}^{i-1} \setminus q_{r_{i-1}}} c(p) \right) + \left(c(q_{r_{i-1}}) \cdot \sum_{p \in p_{r_{i-1}}} c(p) \right).$$

While this will usually increase the cost of the hypothesis, entity resolution can unify multiple instances of the same fluent within the hypothesis. Thus reducing the complexity of the explanation is emphasized not just through simpler conditions and fewer rules, but also by using the same information to explain multiple claims.

After this burst of interest in plan recognition using abduction, it seemed to dissipate for almost sixteen years. Claims have been made that the factor that contributed most to this loss is the exponential computational cost resulting from the sizes of each $O_{r_0, \dots, r_{i-1}}^i$; most iterations, especially the earlier ones, add more unexplained fluents per rule than remove explained fluents. Inoue and Inui approached this issue by converting Hobbs et al.'s [119] weighted method above into an *integer linear program* (ILP) [129]. In this ILP, each hypothesis $h_{r_0, \dots, r_{i-1}}^i \in \mathcal{H}$ has its own equation with the following binary variables and associated constraints. We also define set FH as the set of all fluents found in a hypothesis, including duplicates that have not been unified:

- ILP variable $h_{f \in FH} = 1$ if and only if there is some $0 \leq j \leq i$ such that $f \in h_{r_0, \dots, r_{m-1}}^m$. This indicates that there exists an explanation for f given the current hypothesis's derivations. Because the initial hypothesis is the observation, we have the constraints

$$h_o = 1 \text{ for all } o \in O^0.$$

- ILP variables $u_{f,g \in FH} = 1$ if and only if f and g are unified in this particular equation's hypothesis. Because unification can only happen when both fluents f and g are explained by the current equation's hypothesis, we have the constraint

$$u_{f,g} \leq 0.5 (h_f + h_g).$$

- ILP variables $r_{f \in FH} = 1$ if and only if the weight of f is ignored when computing the total cost. This occurs if (1) it is replaced when applying some rule in the backwards chain or (2) it is unified with another instance of smaller weight. This

is represented by two sets of constraints. The first constraint literally counts the number of opportunities for which f may be replaced:

$$r_f \leq \sum_{e \in \text{explains}(f)} h_e + \sum_{g \in \text{smaller}(f)} u_{f,g}$$

where $\text{explains}(f) = \{e \mid e \in FH \wedge \{e\} \cup \mathcal{KB} \models f\}$ is the set of fluents in some hypothesis that can derive f (thus it contributes to the count if it is in this particular equation's hypothesis) and $\text{smaller}(f) = \{g \mid g \in FH \wedge c(g) < c(f)\}$ is the set of fluents in some hypothesis whose weight is less than f 's (thus it contributes to the count if has unified with f in this particular equation's hypothesis). The second constraint ensures that the sum over explanations holds when there is a conjunction since all such literals must be true:

$$\sum_{a \in \text{conjunct}(f)} h_a = h_f \cdot |\text{conjunct}(f)|$$

where $\text{conjunct}(f) \subseteq FH$ is the set of fluents in some hypothesis that are in the same conjunction as f . Thus $|\text{conjunct}(f)|$ is one less than the number of fluents in the conjunction itself ($f \notin \text{conjunct}(f)$), which requires all $h_a \in \text{conjunct}(f) = 1$ if $h_f = 1$ or 0 otherwise.

- ILP variables $s_{c \in vars, d \in vars \cup consts} = 1$ if and only if, when unifying two fluents, one fluent's variable c is substituted for another variable or a constant d . Because these variable substitutions must hold consistently for unification to work, we have constraints

$$u_{f,g} \leq \frac{\sum_{(c,d) \in \text{unifySubs}(f,g)} s_{c,d}}{|\text{unifySubs}(f,g)|} \text{ for each } f, g \in FH$$

where $\text{unifySubs}(f, g)$ is the set of all variable substitutions that must hold for f and g to be unified. Furthermore, because a variable can only be equivalent

to one constant at a time in logic (distinct constants cannot resolve to the same entity), we have the substitution constraint

$$\sum_{z \in \text{constSubs}(c)} s_{c,z} \leq 1 \text{ for each } c \in \text{vars}$$

where $\text{constSubs}(c) \subseteq \text{consts}$ is the set of all constants to which variable c may be substituted. Lastly, because a variable may be substituted for another variable, transitivity of substitutions must be maintained. The transitivity constraint is captured using a clustering of variables for each possible substitution constant and depends on the size of the cluster.

3.1.2 Parsing

In addition to plan recognition’s origins in the study of natural language, Geib and Steedman formally showed that there are reductions from various natural language tasks to plan recognition [92]. In their initial work, they do this to explain plan recognition as a *parsing* problem to identify part-of-speech tags rather than abduction for semantic understanding. Parsing is a hierarchical breakdown of a natural language artifact where each component is iteratively composed of smaller components until those components are atomic sequences of words that cannot be separated. The formulation and structures resemble those used for automated planning with HTNs (see Section 2.3), as both are derived from the computational structure called a Context-Free Grammar (CFG). The connections between these concepts are:

- The set of terminal symbols Σ for a CFG are the symbols that may appear in strings from its language. These are the same as the terminal tasks \mathcal{T}_t for a HTN. We thus observe an individual performing a sequence of atomic actions $o_1, o_2, \dots, o_n \in \mathcal{T}_t$.

- The set of nonterminal symbols Γ for a CFG are the symbols that capture substructures for string generation. These are non-terminal tasks \mathcal{T}_n for a HTN because specific combinations of atomic actions can solve a task that might be necessary to complete a more complex task. Because an individual performs atomic actions to complete these higher-level tasks, they are the tasks that can be recognized. That is, $\mathcal{H} \subseteq \mathcal{T}_n$.
- A transition function $\delta : \Gamma \rightarrow (\Sigma \cup \Gamma)^*$ represents the string generation process for a CFG where $*$ is the Kleene closure. This substitutes substructures with more specific substructures and actual characters in the string, and the generation process recursively applies δ until all substructures are removed. A HTN also generates plans to solve each subtask as a partially ordered list of other subtasks and atomic actions that must be completed. Thus the derived executable plan must recursively satisfy subtasks with more specific actions and subtasks until only a sequence of atomic actions is present that satisfies every subtask, including the primary task from which all these decompositions were made. This means that there exists a fiber (an inverse for a function that is not one-to-one) of δ to uncover the derivations used to generate a string. Likewise, there exists a fiber for the process of breaking down subtasks to identify the HTN’s planning process. The fiber for CFG’s is called a *parser*.

At the following IJCAI conference, Geib [91] extended this formalization using Steedman’s combinatory categorial grammar (CCG) [259], an equivalent representation of the CFG used for natural language processing. The CCG contains *atomic categories* A_1, A_2, \dots, A_x and *complex categories* C_1, C_2, \dots, C_y ; we will call the set of all atomic and complex categories A . and C . respectively. Atomic categories are similar to terminal symbols and atomic actions above, but complex categories embed the hierarchy’s transitions instead of using nonterminal symbols that decompose. To notate a complex category, a set $\{A_i \mid 1 \leq i \leq n\}$ represents a set of partially ordered

atomic categories $A_{i_1}, A_{i_2}, \dots, A_{i_n}$ and operators / and \ enforce ordering between the sets. Like a parse tree, these operators represent an edge for a parent-child relation (the child is below the parent in a tree's spatial representation—the child is adjacent to the lower portion of the slant-looking operator in a complex category). A single complex category can have any number of sets and operators, but *children must be parsed before the parent*.

For plan recognition using CCGs, there are also a set of observable action types Σ (similar to *acts* in abduction while complex categories are like A , see Section 3.1.1) and a function $f : \Sigma \rightarrow (A. \cup C.)$ that maps each observable action type to a subset of all the categories whose embedded tree structures represent the possible plans that perform the action. Many action types will map to only a respective atomic action (for example, $f(\sigma \in \Sigma) = A_k \in A.$ where σ and A_k have the same label) if they are not intended to be explained/broken-down. Hence f defines the plan library (though Geib calls it a *plan lexicon* instead). f is not unique because there are multiple ways to encode each tree using these operators, but the choice of embedding can play a large role in the computational complexity of parsing.

The parsing process for this application of CCGs is named ELEXIR and takes advantage of the root-result for each category as the overall task/plan for the category.

Definition 32. *For a complex category $C_i \in C.$, the leftmost atomic category in its embedded tree structure is its root-result. An atomic category $A_i \in A.$ is its own root-result. We notate this with function $rr : (A. \cup C.) \rightarrow A..$*

Definition 33. *The head of a plan for a given atomic category $A_i \in A.$ is the set of observable action types $\sigma \in \Sigma$ whose root-results are A_i :*

$$h(A_i \in A.) = \{\sigma \in \Sigma | rr(f(\sigma)) = A_i\}.$$

For the complexity of ELEXIR to be linear with respect to the number of observed action types, all complex categories must:

1. Have a single atomic category as the root-result $\alpha = rr(C_i)$ (for all $C_i \in C$) and
2. Be of the form $(\alpha(/ \beta_i)^*) (\setminus \gamma_j)^*$ where $*$ is the Kleene closure, α is the above root-result, and each $\beta_1, \dots, \beta_{i*}, \gamma_1, \dots, \gamma_{j*} \in A$ is a set of partially-ordered atomic categories. This form guarantees that the first property holds.

This complex category structure creates *step-by-step plans* where all γ_{j+y} must be completed before γ_j , γ_1 must be completed before α , α must be completed before β_1 , and all β_{i-x} must be completed before β_i such that β_{i*} must be completed last. This is specifically a plan of the form $\gamma_{j*}, \gamma_{j*-1}, \dots, \gamma_1, \alpha, \beta_1, \beta_2, \dots, \beta_{i*}$. ELEXIR receives a sequence of observed action types $[\sigma_1, \sigma_2, \dots, \sigma_x] \in \Sigma^x$ and iteratively reads each σ_i from left-to-right to write a category in $f(\sigma_i)$. Geib's small examples and explanations prefer to keep the outputs of f small in cardinality so that enumerating all possible plans is not computationally complex. As categories are written for each read σ_i , the required structure of complex categories allows the right-hand sides of the \setminus operators to cancel with already-written atomic categories. Then the remaining embedded tree structure, which looks like a to-do list for an unfinished plan, serves as a possible explanation for the sequence of observations.

Like in abduction, multiple explanations are derivable. Each such explanation consists of a combination of mapped categories via f as well as assuming any complex category will eventually transform into its root-result; let us call this set of possible explanations E . Thus probabilistic measures are used to compare the likelihood of each possible explanation in E . Ultimately, the powerset of atomic categories $\mathcal{P}(A)$ forms the set of recognized goals over which we apply weighted model (explanation) counting:

$$P(p \in \mathcal{P}(A.) | [\sigma_1, \dots, \sigma_x]) = \sum_{e \in E} \mathbf{1}(p \in e) \cdot P(e | [\sigma_1, \dots, \sigma_x])$$

where $\mathbf{1}(\cdot)$ is the indicator function for statement \cdot . Given that an explanation contains at most $|C|$ unique complex categories, we then compute the probability of each individual explanation as

$$P(e \in E | [\sigma_1, \dots, \sigma_x]) = Z^{-1} \cdot \prod_{z=1}^x P(e_z | \sigma_z) \cdot \prod_{C_i \in e} P(rr(C_i))$$

for a normalizing constant Z . The left product counts the number of times observed action type σ_i derives the category $e_i = f(\sigma_i)$ that it wrote to the explanation e (compared to all of σ_i 's derivations). The right product is the prior that an atomic category in the explanation appears as a root-result. While each derivation for the left product is simply counted using a corpora, the priors must be manually assigned based on *what one expects the observed individual(s) to do*.

Geib updated ELEXIR since proposing this initial algorithm because the model counting was not able to recognize all valid plans until the observation sequence was completed [89]. This post-processing vs. real-time issue is becoming a more present-day concern as researchers in PAIR are moving towards recognition tasks using inputs other than natural language artifacts such as sensor data [79]. Although one of the largest contributions to this weakness in ELEXIR is the restriction on the structure of complex categories, changing the structure would ruin the runtime complexity—this is not ideal. Geib thus introduces heuristics for predicting future categories written to the explanations so that the missing root-results (forming the recognized plan) are considered. The heuristics either maximize or average the change in conditional probability by writing all categories with a specific root-result, canceling any atomic categories written along the way. Then the change in the probability shows whether the new category can simplify the explanation or improve it despite the potential cost to assume the category.

3.1.3 Recognition as Planning

Recognition as Planning (RaP) was introduced not only as the first recognition algorithm for a STRIPS-style representation, but also as a deviation from matching within a plan library to matching within a plan domain [226, 227]. The advantage of the domain is that a *matching plan is generated at recognition time* rather than hoping that a precomputed plan matches.

The RaP class of algorithms' hypotheses are the set of goal conditions $\mathcal{H} = \{G_1, G_2, \dots, G_x\}$. The method used to find the best matching hypothesis is to *simulate the observed agent* solving each hypothesized goal with respect to the observation sequence. This focus on the simulation allows off-the-shelf classical planners to perform the majority of the computational efforts. Matching plans are generated during the recognition algorithm's execution and then compared, rather than the traditional approach of precomputing a library of plans per hypothesis for evaluation. We specifically explain the important details for the probabilistic extension [227] because many variations, including the one discussed in this dissertation, use it as their underlying framework.

Formally, a probabilistic RaP problem is a tuple $\mathcal{R} = (\mathcal{D}, I, \mathcal{O}, \mathcal{G})$ where $\mathcal{D} = (F, A)$ is the planning domain (the set of logic fluents F and actions A generate a search space), I is the known initial state from which the observation sequence $\mathcal{O} = o_1, o_2, \dots, o_m$ begins, each observation is an action $o_i \in A$, and $\mathcal{G} = \mathcal{H}$ is the set of hypothesized goal condition sets. By this definition, $\mathcal{KB} = (\mathcal{D}, I)$. If the correct set of goal conditions $G \in \mathcal{G}$ is recognized, then \mathcal{O} is a subsequence of some plan $\pi_G = a_1, a_2, \dots, a_{n \geq m}$ that solves the planning problem $\mathcal{P} = (F, A, I, G)$. That is, there exists some monotonically increasing function $f : \{1, 2, \dots, m\} \rightarrow \{1, 2, \dots, n\}$ such that $o_i = a_{f(i)}$. Because it can be difficult to identify the exact goal conditions when the observation sequence has missing observations or is ambiguous over a subset of the hypotheses [156], RaP outputs a distribution over how likely each hypothesis

is given the observation sequence: $P(G \in \mathcal{G} | \mathcal{O})$. Using Bayes's Rule, we rewrite this probability as $Z^{-1} \cdot P(\mathcal{O} | G) \cdot P(G)$ where Z is a normalizing constant and $P(G)$ is simply a prior belief over the goals.

The likelihood $P(\mathcal{O} | G)$ is further rewritten as a summation over Π_G , the set of plans that solve G :

$$\sum_{\pi \in \Pi_G} P(\mathcal{O}, G | \pi) = \sum_{\pi \in \Pi_G} P(\mathcal{O} | \pi, G) \cdot P(\pi | G).$$

$P(\mathcal{O} | \pi, G) = P(\mathcal{O} | \pi)$ is binary because \mathcal{O} either is or is not a subsequence of π . This introduces an opportunity to partition Π_G into the subset of plans that solve G and contain \mathcal{O} as a subsequence, Π_{G+O} , and the subset of plans that solve G and do not contain \mathcal{O} as a subsequence, $\Pi_{G+\bar{O}}$. In fact, these sets of plans correspond to two new sets of goal conditions $G + O$ and $G + \bar{O}$ that complement each other. Conceptually, an agent that is intentionally solving G will also be solving $G + O$. On the other hand, an agent that is intending to solve some other goal in \mathcal{G} should be solving $G + \bar{O}$ if they were to solve G coincidentally. This is an *assumption of the observed agent's rationality* [19].

Using this complement, we rewrite the likelihood with a denominator that sums to 1:

$$P(\mathcal{O} | G) = \frac{P(\mathcal{O} | G)}{P(\mathcal{O} | G) + P(\neg \mathcal{O} | G)}. \quad (3.1)$$

As both probabilities have the same form as the likelihood, we get

$$P(\mathcal{O} | G) = \sum_{\pi \in \Pi_{G+O}} P(\pi | G) \text{ and}$$

$$P(\neg \mathcal{O} | G) = \sum_{\pi \in \Pi_{G+\bar{O}}} P(\pi | G).$$

Furthering the assumption of rationality, we expect the observed agent to act as optimally as possible to achieve their intended goal. Thus the probability of performing

some plan is inversely proportional to its cost; Ramírez and Geffner use the Boltzmann distribution for this purpose $P(\pi | G) = Z'^{-1} \cdot e^{-\beta \cdot \text{cost}_\pi}$ where β is some constant and Z' is a normalizing constant over Π_G —from a physical perspective, the system’s set of states is the set of plans Π_G that satisfy the goal where each state’s energy is based on its cost $\text{cost}_{\pi \in \Pi_G}$. Because Π_G can be very large, if not infinite, the final assumption is that the *exponential decay will allow the term with the greatest probability mass to dominate the distribution* and thus eliminate the need for a summation.

That is,

$$P(\mathcal{O} | G) \approx \max_{\pi \in \Pi_{G+O}} e^{-\beta \cdot \text{cost}_\pi} = e^{-\beta \cdot \text{cost}_{\pi_{G+O}^*}}$$

where π_{G+O}^* is the optimal plan that solves $G+O$ with the least possible cost. Likewise,

$$P(\neg \mathcal{O} | G) \approx \max_{\pi \in \Pi_{G+\bar{O}}} e^{-\beta \cdot \text{cost}_\pi} = e^{-\beta \cdot \text{cost}_{\pi_{G+\bar{O}}^*}}.$$

These optimal plans can be found using any off-the-shelf classical planner—there are $2|\mathcal{G}|$ problems to solve. These problems are of the form

$$\mathcal{P}_O^G = (F_O, A_O, I \cup \{p_0\}, G \cup \{p_m\}) \text{ and}$$

$$\mathcal{P}_{\neg O}^G = (F_O, A_O, I \cup \{p_0\}, G \cup \{\neg p_m\})$$

for each $G \in \mathcal{G}$. The set of fluents and actions are augmented to track the simulation’s progress through the observation sequence:

- $F_O = F \cup \{p_0, p_1, \dots, p_m\}$ and
- $\text{add}_{a \in A_O} = \text{add}_{a \in A} \cup \{p_{i-1} \rightarrow p_i \mid o_i = a\}$

where $q \rightarrow r$ is a conditional effect that adds/deletes r from the state s when a is performed if and only if $q \in s$. The solution to \mathcal{P}_O^G is a plan in Π_{G+O} and the solution

to $\mathcal{P}_{\neg O}^G$ is a plan in $\Pi_{G+\bar{O}}$, which together are sufficient for computing the probability estimates and overall distribution $P(G \in \mathcal{G} | \mathcal{O})$.

This version of RaP specifically addresses intent recognition (see Section 3.3) because each hypothesis is a set of goal conditions. That is, we recognize the goals as the driving motivation behind the observed agent's actions. However, an analogous derivation adjusts RaP to perform plan recognition where the hypotheses are a (possibly infinite) set of plans $\mathcal{H} = \tilde{\Pi}$ that the agent could be performing $P(\pi \in \tilde{\Pi} | \mathcal{O})$ [252, 83]. Many of the derivations are analogous to the procedure above when we now compute $P(\pi | \mathcal{O}) = Z^{-1} \cdot P(\mathcal{O} | \pi) \cdot P(\pi)$ by Bayes's Rule. We rewrite the likelihood:

$$P(\mathcal{O} | \pi) = \sum_{G \in \mathcal{G}} P(\mathcal{O}, G | \pi) = \sum_{G \in \mathcal{G}} P(\mathcal{O} | \pi, G) \cdot P(G | \pi)$$

where \mathcal{G} is the set of goals that R_{ing} believes R_{ed} could be trying to accomplish. The term $P(\mathcal{O} | \pi, G)$ from above is not too useful now because it is 0 when \mathcal{O} is not a subsequence of π (hence no partitioning since π is the query). However,

$$P(G | \pi) = Z''^{-1} \cdot P(\pi | G) \cdot P(G)$$

using Bayes's Rule. We found above that

$$P(\pi | G) = \begin{cases} Z'^{-1} e^{-\beta \cdot cost_\pi} & \text{if } \pi \text{ solves } G, \pi \in \Pi_G \\ 0 & \text{otherwise} \end{cases}$$

for some constant β , and the prior is the same from the original equation. The prior is thus contained in \mathcal{KB} for the original algorithm, which means this alternative probability can be computed for plan recognition as well.

There are quite a few extensions of the Ramírez and Geffner method due to its ability to recognize over a generalized domain rather than a constrained plan

library. The same authors, Ramírez and Geffner, created the simplest extension the following year to perform recognition in stochastic shortest path (SSP) domains. This relies on planning with MDPs (see Section 2.4). Sohrabi et al. account for noisy sensors that can ignore observations in the sequence using additional actions that omit observations in exchange for a greater cost [252]. The remaining extensions instead focus on speeding up the recognition algorithm because it is computationally expensive to run classical planners multiple times (specifically, $2|\mathcal{H}|$ times). We refer the reader to Section 3.3.2 for these because they are specifically developed for goal recognition without a confirmed plan recognition analogue.

3.2 Activity Recognition

Activity Recognition is the process of identifying a higher-level representation of a single action or subtask given a sequence of observed information at a lower-level representation. Hierarchical representations such as HTNs often observe the terminal tasks so that each $o \in \mathcal{O}$ also satisfies $o \in \mathcal{T}_t$, but the low-level information might range from sensor data to configuration settings at various time stamps. The latter is unlike plan recognition, which already has the higher-level information as its observed input. We will discuss both cases, but it is important to distinguish the differences in representation because their sources will vary both the knowledge base and types of algorithms used for activity recognition. In either case, the challenge in activity recognition is to answer the question, “what is the agent doing *at the moment?*” Hypotheses are usually a subset of non-terminal tasks $\mathcal{H} \subseteq \mathcal{T}_n$ for HTN-represented \mathcal{KB} or a subset of actions $\mathcal{H} \subseteq A$ when sensor data is observed.

As a disclaimer, the computer vision research community has also studied activity recognition where the sensor is a camera. They consider additional challenges such as identifying where the activity occurs in the image and which entities (agents and objects) are involved in the activity. A survey of the techniques specifically developed

within the computer vision research community was released within the past decade [2] and will not be covered in this dissertation. Though we discuss activity recognition with camera sensors, our focus will be on work within other research communities such as PAIR.

3.2.1 Hidden Markov Models and Their Generalizations

Sensors give insights into the environment, but their insights can still be limited. The hardware can only receive signals for which it is designed (cameras for light, microphones for audio, chemical detectors, etc.), and sensors are not always reliable due to potential hardware faults and physical variability in the world. To address this uncertainty of the environment from sensor data limitations, models analyzing sensor data are often probabilistic. Rather than traditional probability of single events, the stream of information from sensors is ongoing and is expected to change as the environment changes over time.

Definition 34. A Markov chain is a sequence of random variables $X_0, X_1, \dots, X_n \in V$ where V is the set of assignable values and X_i 's value only depends on the value of X_{i-1} . The sequence shows a progression of how the observed value changes over time.

Definition 35. The transition matrix of a Markov chain is a $|V| \times |V|$ matrix where entry

$$T_{i,j} = P(X_k = v_j \mid X_{k-1} = v_i \in V) \text{ for all } k \in \mathbb{Z}^{\geq 1}.$$

This is the probability that the random variable's assignment changes from value v_i to v_j between two consecutive discrete time steps in the sequence.

When the environment is assumed to be a Markov chain, the set of assignable values is the set of states $V = S$. Then the sensor measures its respective signal based on the environment's state. Because the sensor is rarely complete (with respect to the environment) and sometimes unreliable, the measurement is also a probability distribution.

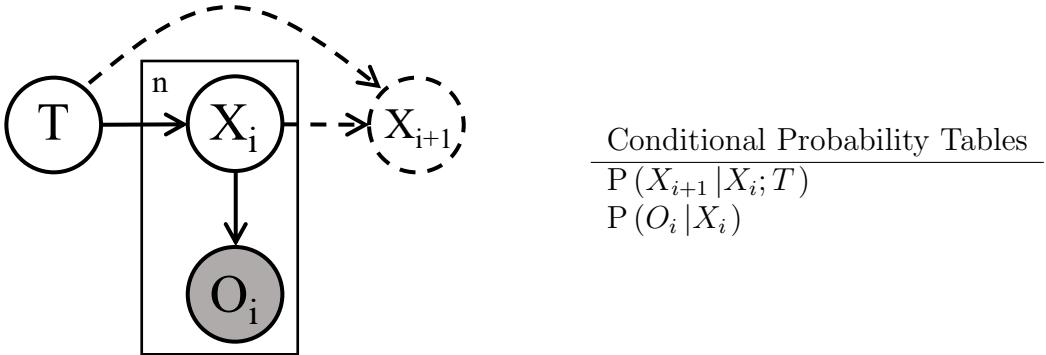


Figure 3.1. A graphical model of a HMM with global transition matrix T . Shaded nodes represent observed random variables and edges $u \rightarrow v$ contain a conditional probability table $P(v|u)$.

Definition 36. *A hidden Markov model (HMM) is a probability model with random variables $X_0, X_1, \dots, X_n \in V$ and $O_1, O_2, \dots, O_n \in \Omega$ where X form a Markov chain and each O_i 's value only depends on the value of X_i . The O random variables are all observed, but the X random variables are not observed. See Figure 3.1 for a graphical model representation.*

The role of a HMM is to *infer the actual state of the environment* given just the observations from the sensor, which allows prediction of the upcoming states via the transition matrix. This transition matrix is similar to the transition function (see Definition 21) assigned to each action, which is the key to using HMMs in activity recognition as well as early speech recognition [87]. Specifically, performing an action or addressing a task has direct changes on the environment based on its current state; thus *each action/task corresponds to its own Markov chain*. If S is not known in advance, especially when the environment is the real world itself, then the number of underlying states is often assumed and each action's/task's transition matrix is learned from recorded observation sequences. This reduces activity recognition to a *supervised machine learning classification* problem where the inferred transition matrix is the input feature and the action/task that was recorded is the output label. The set of observable values Ω will depend on the available sensor information and

should be included in \mathcal{KB} . Likewise, information needed to generate and run the classifier or probabilistic model should be included in \mathcal{KB} .

When using HTN representations rather than sensor data, it is often the case that $\Omega = \mathcal{T}_t$ and $\mathcal{H} = \mathcal{T}_n$. However, the HMM’s single transition of states lacks the hierarchical structure implicit to the representation of task networks. If a non-terminal task T_{\top} decomposes into more non-terminal tasks T_1, T_2, \dots, T_n that can each be recognized, then the transition matrix for T_{\top} might not have a single, constant transition matrix—it would *transition between transition matrices* for T_1 through T_n based on the ordering constraints of the chosen decomposition. This led to the notion of nesting different HMMs within each other so that a state either generates an observation or transitions to another state, either at its level (the same Markov chain), a lower level (a start state for a different Markov chain), or back one level (a termination state for the current Markov chain) [74]. Called a *hidden HMM* (HHMM), it was later generalized to the *abstract HMM* [36] and then redefined using this abstract definition [35]. Blaylock and Allan compromised this generality with a bound on the number of levels, which creates a simpler model for computational purposes called the *cascading HMM* (CHMM) [25]. The CHMM simply rolls out the underlying Markov chains to create a grid of states with a unidirectional, bifurcating flow of information: from greatest level of abstraction to lowest as well as over time.

Once the deep learning craze began, though, this route for activity recognition was quickly abandoned. HMMs were replaced with temporal-based deep learning methods such as *long-short term memory (LSTM) networks* [120]. The output layers of these deep neural networks use the high-level action/activity labels as training output with the corresponding temporal layers as training inputs [213].

3.2.2 Latent Semantic Analysis

Latent semantic analysis (LSA) is a *generative modeling* approach that assumes that the observations in a dataset (especially for a single entry) are similar in semantic meaning and are thus relevant to each other. However, even though each observation has its own set of meanings, it is unknown which ones explain its relation to the other observations. That is, there is a *set of hidden definitions* that describes what overall kinds of information are observable, and identifying these definitions should explain the phenomena captured in the dataset.

From a graphical modeling perspective, we represent these latent semantics as unobserved nodes containing distributions over the set of observations. As a generative model, the ‘story’ explaining their presence in the dataset’s creation involves sampling some distribution over distributions (such as the Dirichlet²), and then those distributions are sampled to identify the actual observations. Learning the *hyperparameters* that shape these distributions creates clusters of observations, and the modes of each cluster represent its semantic interpretation.

The simplest example of LSA is actually simpler than the HMM in design because it *assumes independence over time*. Thus the latent variable is the underlying state, which is sampled from some unknown distribution that does not change with respect to information about any former state. However, this latent variable’s value does have some correlation to the observed value.

Definition 37. A bag of words (*BOW*) model is a probabilistic model with random variables $X_0, X_1, \dots, X_n \in V$ and $O_1, O_2, \dots, O_n \in \Omega$ where all X_i are sampled from some global distribution D independently of each other and each O_i ’s value only de-

²The Dirichlet is a random probability mass function that captures the variance of distributions identified from observed events that actually occur based on some true distribution [84]. For example, flipping a fair coin should produce heads half the time when observed over many trials, but getting heads forty-nine percent of the time would also not be too surprising. However, getting heads ten percent of the time is unlikely.

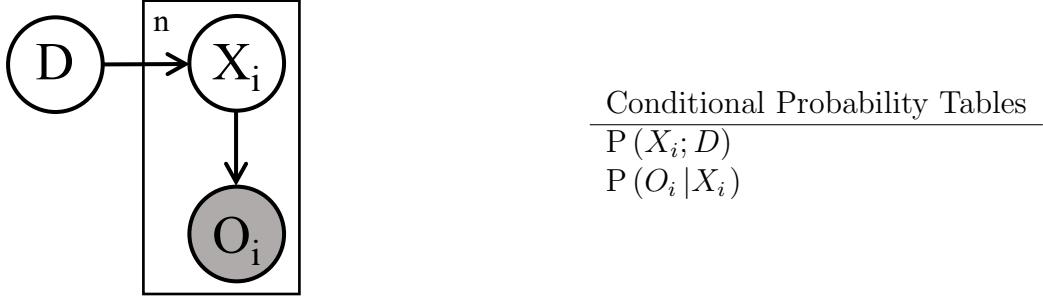


Figure 3.2. A graphical model of a BOW with global distribution D . Shaded nodes represent observed random variables and edges $u \rightarrow v$ contain a conditional probability table $P(v|u)$.

pends on the value of X_i . The O . random variables are all observed, but the X . random variables are not observed. D is typically unobserved as well. See Figure 3.2 for a graphical model representation.

One of the most well-known applications of LSA is topic modeling; the observations are words in a document and the latent semantic distributions are topics. Because the modes of a distribution over related words are easily interpreted as their shared definition, topics are easily defined by the learned clusters of words. The most frequently used and tweaked topic model is Latent Dirichlet Allocation (LDA) that accounts for the fact that a document can be described topically without needing to know the order of the words [26]—this is an extension of the BOW model due to adding additional steps before sampling each observation from D . Its graphical model, generative process, and mathematical representation are presented in Figure 3.3.

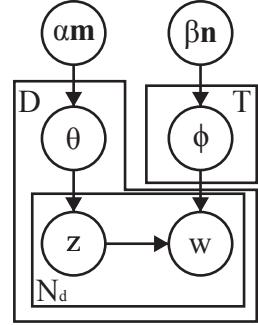
Of the well-known LSA methods, topic models such as LDA gained popularity in activity recognition prior to the deep-learning fad, especially for applications that use out-of-the-box statistical models [139, 51, 233, 50]. Activity recognition methods that use vanilla LDA have to assume that in a similar manner to topic modeling, describing what generally happens during a sequence of sensor readings does not rely on the ordering. Then the clusters of sensor readings can be interpreted by the modes, if interpretation is desired at all. Huynh et al. [128] initially used LDA with

Algorithm 1: GenerateLDA($T, D, N, \alpha\vec{m}, \beta\vec{n}$)

```

forall the topic  $t \in \{1, \dots, T\}$  do
     $\lfloor$  draw  $\phi_t \sim \text{Dirichlet}(\beta\vec{n})$ 
forall the document  $d \in \{1, \dots, D\}$  do
     $\lfloor$  Draw  $\theta_d \sim \text{Dirichlet}(\alpha\vec{m})$ 
    forall the index  $i \in \{1, \dots, N_d\}$  do
         $\lfloor$  Draw topic  $z_{d,i} \sim \theta_d$ 
         $\lfloor$  Draw word token  $w_{d,i} \sim \phi_{z_{d,i}}$ 

```



$$P(\vec{z}, \vec{\theta}, \vec{\phi} | \vec{w}, \alpha\vec{m}, \beta\vec{n}) \propto P(\vec{w} | \vec{z}, \vec{\phi}) \cdot P(\vec{z} | \vec{\theta}) \cdot P(\vec{\theta} | \alpha\vec{m}) \cdot P(\vec{\phi} | \beta\vec{n})$$

where $P(w_{e,j} \in \vec{w} = v | \vec{z}, \vec{\phi}) \approx (\sum_{d=1}^D \sum_{i=1}^{N_d} \mathbf{1}(w_{d,i} = v \wedge z_{d,i} = z_{e,j} \wedge (e, j) \neq (d, i)) + \alpha m_v) / (\sum_{d=1}^D \sum_{i=1}^{N_d} \mathbf{1}(z_{d,i} = z_{e,j}) - 1 + \alpha)$ and $P(z_{d,j} \in \vec{z}_d = t | \vec{\theta}) \approx (\sum_{i=1}^{N_d} \mathbf{1}(z_{d,i} = t \wedge j \neq i) + \beta n_t) / (N_d - 1 + \beta)$

Figure 3.3. LDA’s generative process, graphical model, and mathematical representation.

on-body sensors to recognize daily routine activities. They used the learned model to develop a system that can automatically annotate future recorded data from the sensors. Their original work provided interpretive evidence for recognizing clusters of wearable sensor data as daily activities by aligning the learned topic clusters with an annotated timeline of activities, but few other applications have had access to such annotations.

Wang and Mori [283] trained a LDA model using annotated video sequences to recognize predefined actions—they named the approach *Semilatent Dirichlet Allocation* because it was not completely unsupervised like traditional LSA methods. Zhang and Parker [296] used LDA without any modifications to cluster readings from a RGB-D sensor attached to a robot. While Wang and Mori simply used the pixelated image data with flow fields, Zhang and Parker compressed the three-dimensional point-cloud data using local spatio-temporal features into vectors of four-dimensional cuboids. However, both approaches map their representations to *codebooks* with a finite set of symbols; unsupervised clustering methods derive the mapping from the sensor data to a symbol in the codebook. This limits future observations because they

must also map to a predefined symbol in this codebook. Novel, unique inputs will consequently be misassigned to an input symbol. Freedman (the author of this dissertation), Jung, and Zilberstein previously suggested a third representation to avoid this limitation by simply *discretizing the joint-angles describing the observed agent’s posture* [79, 80]. The joint-angles are independent of scale, identical for the same postures between different body shapes, and exist within a reasonably-sized space based on the granularity parameter (so a codebook is not necessary). We investigate how discretization granularity affects the size of the library of available input symbols and the impact on the performance of LDA in Section 4.2.1.

Once the deep learning craze began, though, this route for activity recognition was quickly abandoned just like HMMs. LDA was replaced with unsupervised deep learning methods such as *deep autoencoders* [160]. Some compressed middle layers within these deep neural networks are used as the feature inputs for classification rather than the latent variable’s values [17].

3.2.3 Activity Recognition in the Wild

Unlike the aforementioned methods that assume a still camera and/or scripted data collection method, Gori et al. [103] placed a RGB-D sensor on a mobile robot and tried to train and recognize activities *in the wild*. This phrase is becoming more popular to refer to research that has a real-time component in the real world rather than a controlled laboratory setting. In this case, the robot wandered through hallways inside an academic institutional building and recorded people without any prior arrangement. Thus the humans in the data act naturally (with respect to being watched by a passing robot) without a script to dictate what to do during the recording session.

Despite the variety of possibilities that one would expect, the authors manually annotated the data and found nine distinct activities worth classifying. The dataset,

which is now available to the public, ignores a few anomalous activities and contains an additional tenth class for false sensor readings. For their own activity recognition research, Gori et al. used the RGB-D data and robot’s position data (both via open odometry and closed map positioning). Since multiple representations of RGB-D have been proposed, they compared the recognition results for five representations. New challenges they encountered from their ‘in the wild’ recording were identifying observed humans’ velocities and distances, which were changing on both the observed humans’ behalf and the moving camera’s. The change was identified by applying the robot’s kinematic rotation matrix (how its heading and position changed in the world) between two time frames to the human’s position in the first of these frames; this gives the human’s change in position and adjusting it for the elapsed time yields their velocity.

To reduce the dimensionality of all the data, they applied the standard codebook process using k -means clustering on the set of features to assign the vast range of inputs to one of k values. Unlike other work described in Section 3.2.2, each feature uses a separate codebook such that the actual feature vector is a concatenation of each feature’s cluster number. Each video’s feature vector input and labeled activity output trained a non-linear support vector machine with a χ^2 kernel. Of the representations, the depth image without any joint position information provided the best results. The authors hypothesize that this is a result of the moving sensor since *current methods for identifying joints within depth images assume a still camera*. They also found that the inclusion of the additional non-image features (robot’s position, human velocity, and human distance from robot) further improves the recognition accuracy in the test set.

3.2.4 Activity Recognition for Assistive Technologies

The domain of assistive technologies has greatly benefited from PAIR research since observing users with special needs allows the system to autonomously act with respect to the recognized prediction and provide help. Previously, such technologies were nothing more than tools that were engineered to merely reduce the overload for users with special needs while still requiring them to do all the work; this can still impose additional stress and difficulty for daily life activities.

Chu et al. [53] created a system that monitors the daily living activities for users with cognitive disabilities who might forget to continue a task that is in-progress or even forget to start the task. Given a schedule of tasks with executable time ranges (start intervals, end intervals, and duration spectrum), a temporal planner is used to construct a schedule with some slack. Then the system must act to ensure that the monitored user maintains this schedule.

Because the user is only measured by a collection of sensors including infrared (IR), radio-frequency identifiers (RFID) placed on objects, and appliance activation sensors, the actual state of the user and their progress is estimated over an entire set of world states. The estimation is a belief state, which is a distribution over states, in a POMDP (see Section 2.4). Thus the set of recognizable activities includes those on the schedule given the assumed world state from the collection of sensor data. Because a POMDP is computationally complex, Chu et al. infer the activity using the traditional HMM over each sensor input's stream of data. It is usually able to take advantage of the schedule's enforced structure on the POMDP to prune the space of hypotheses, easily identifying the activity. This acts as a higher layer in a hierarchy to narrow the focus of the POMDP for localized policies that decide which actions to take in response to the user's progress. When the level of uncertainty in the belief state is great enough that the system's next action (reminding the user which task to perform, telling the user to stop a particular task, waiting, etc.) would change, it

may query the user for their current activity to make a more precise approximation. However, frequent querying introduces unnecessary impingement on the user’s life and the cognitive load might provide additional stress. Thus activity recognition is preferred over frequent questioning.

To measure when the level of uncertainty was too great in belief state b , the first step simply normalizes the entropy of the distribution $H(b)$ and compares it to a small threshold δ . If the threshold is surpassed, then the entropy is considered great enough that assuming the state in b with the greatest probability mass is the true state is too risky. Hence the second step determines whether this ambiguity will result in a change of actions that is *too important to miss*. If the missed opportunity to act is not so vital, then waiting is preferred to avoid the consequences of querying the user as well as take advantage of updated sensor data at a later time step. The expected value for taking action a that is correct for only one of two ambiguous states at current time t is

$$V(b, a, t) \approx b(s_1) \cdot Q(s_1, a, t) + b(s_2) \cdot (\text{ErrorPenalty} + Q(s_2, \text{WAIT}, t))$$

where $Q : S \times A \rightarrow \mathbb{R}$ is a function that approximates the expected reward $\mathbb{E}_{\mathcal{M}}^{\pi}[s]$ when policy π enforces performing action a in state s (in this case, t is considered part of s). This is computed for the inquiry action, the wait action, and each possible action the POMDP can take at b . The action a with the greatest expected value $V(b, a, t)$ is the action the system will take.

3.3 Intent and Goal Recognition

Intent Recognition has the most vague definition among the types of recognition because it studies the ‘driving motivation’ behind what is observed. In most cases, the underlying motivation for doing something is characterized as the goal or some

subset of possible goal conditions. Logic-based representations define $\mathcal{H} \subseteq 2^F$ where F is a set of features that can describe states in S and task hierarchies define $\mathcal{H} \subseteq \mathcal{T}_n$ (or $\mathcal{H} = \Sigma$ when using a CCG representation) as a set of high-level tasks (much like plan recognition). This is the reason that the term *goal recognition* is also used by some PAIR researchers. However, goals themselves can be selected for various reasons. We are not aware of any research on recognition at the metareasoning level, but Callaway et al. trace users' decision making processes to study their planning and information-gathering strategies [40]. Without approaching metareasoning for decision making, it is viable that some actions are performed for the sake of facilitating another action that is crucial to R_{ed} 's plan. This resembles taking actions to repair missing causal links that are required to perform upcoming actions in a plan [174]. If these reparation actions are not useful for the goal's completion, though, then there might be a misinterpreted motive. In this case, intent recognition can be viewed as a prediction problem instead where $\mathcal{H} \subseteq A$ or $\mathcal{H} \subseteq S$.

Though there is potential overlap with plan recognition's formulation, the fundamental question is different: “*why* is the agent doing this?” (both ‘overall’ and ‘at the moment’). The variation of RaP cited at the end of Section 3.1.3 modified \mathcal{H} from sets of goal conditions to sets of plans, but others made variations for intent recognition with a focus on only identifying the goal conditions rather than the plan or policy explanations [228, 69, 217]. In all the intent recognition approaches mentioned so far, each observation $o \in \mathcal{O}$ is again an executable action such that $o \in A$. However, prediction based on agents' trajectories in continuous space [277, 281] is usually observed as either spatial coordinates such as $o \in \mathbb{R}^2$ or kinematic configuration spaces consisting of rotations and translations of joints [186]. Likewise, Sohrabi et al. motivate observations from sensing fluents in the environment $o \subseteq F$ [252].

As with plan recognition and activity recognition, \mathcal{KB} will contain the underlying information to construct any models that are used in the intent recognition algorithm.

Though the specific implementations are different depending on representation choice, they are typically probabilistic models, plan libraries, and/or plan domains.

3.3.1 Prediction of Upcoming Actions and Underlying Motives

One of the greater challenges in prediction tasks is the level of specificity because, unlike recognition tasks that usually have a limited set of plans, activities/actions, or goal conditions to select, the *number of intentions is nearly uncountable*. Reasons can span from simple recognition explanations to complex superstitious rules and even lack of motivation ('just because'). One reason for this variety is that intentions are person-specific [173] rather than fact-based. Lesh approached this variation with the introduction of the *Adapt* algorithm, which modifies the results of generic goal recognition algorithms based on each user's recent behavior and an allowable set of adaptations for the given algorithm (thus the adaptations are specific and vary per algorithm).

In order to evaluate the goal recognition algorithm's behavior and determine which adaptations to make, Lesh proposes a formalized definition for goal recognition:

Definition 38. A goal recognizer is a function $R : acts^* \rightarrow G \cup \{nil\}$ where $acts$ is the set of observable actions (see Section 3.1.1), $*$ is the Kleene closure, and G is the set of goals (having a variety of forms rather than being limited to just states). An output of *nil* indicates that the goal recognizer failed to determine a goal from the current observation sequence input. Lesh emphasizes that R cannot return a distribution over G .

Lesh also introduces two measurements that involve a set of persons Q and two fluents of the form $exec(q \in Q, O \in acts^*)$ = "person q is observed performing action sequence O " and $goal(q \in Q, g \in G)$ = "person q has goal g ". To supply context with respect to the notations that we use in this dissertation, the observing agent R_{ing} uses recognizer R to observe agent $R_{ed} \in Q$:

Definition 39. *The coverage metric determines how many observable action sequences R can use to successfully identify the intents of person $q \in Q$:*

$$COV(R, q) = \sum_{s \in \{O \in acts^* | R(O) \neq nil\}} P(exec(q, s))$$

Definition 40. *The accuracy metric identifies how many of the action sequences in the coverage metric return instances that are correct for the given individual $q \in Q$:*

$$ACC(R, q) = \sum_{s \in acts^*} \frac{P(goal(q, R(s)) | exec(q, s)) \cdot P(exec(q, s))}{COV(R, q)}$$

if the coverage is non-0 (otherwise the accuracy is 0). The fraction within the summation is derived from Bayes's Rule, which further explains coverage as a normalizing constant for the probability that a person executes a sequence of actions given their intended goal $P(exec(q, s) | goal(q, R(s)))$. This addresses the fact that coverage only cares about the recognizer returning a goal regardless of its correctness.

With these measurements, a theoretical hierarchy of adapted recognizers exists for each individual. However, the hierarchy introduces a *trade-off of more accuracy for less coverage* so that a recursive traversal of the hierarchy is recommended to get the most predictions at their greatest accuracy. For a domain-specific score function $S(COV(R, q), ACC(R, q)) \in \mathbb{R}$ and q 's user history, we can identify the best trade-off between accuracy and coverage to adapt R for a specific person q in a given domain. The *Adapt* algorithm does this optimization using hill climbing where configurations of R add/remove up to one adaptation per evaluation. q 's history would have to be complete (all observable action sequences $acts^*$) in order to be exact, but this is infinite. Despite its drawback of having high computational complexity for all the user history simulations, *Adapt* is both anytime and parallelizable so that a solution does not take too long to obtain.

3.3.2 Recognition as Planning

Detailed previously in Section 3.1.3, we revisit the overall class of RaP algorithms because the set of hypotheses can change the focus between plan recognition and goal recognition. The original derivation, which assigns $\mathcal{H} = \mathcal{G} \subseteq 2^F$, is the primary focus for many PAIR researchers at the moment. Besides the probabilistic version we derived above that returns a distribution over \mathcal{H} , others have focused on the original version [226] that only returns a subset of \mathcal{H} as the recognized goals. The primary motivation for many of these variations is to speed-up the RaP algorithm’s runtime, which requires $2|\mathcal{G}|$ calls to an off-the-shelf planner.

Shortly after RaP algorithms were introduced, Masters and Sardina investigated its application to the popular GridWorld domain [189]. Because an agent acting in GridWorld can move freely to any adjacent space without an obstacle, they realized that there are almost always multiple plans that optimally solve the goal. With enough observations, they further argue that at least one of these optimal plans *should not match the observation sequence* and the specific plan does not matter for intent recognition (the calculations above only use the plans’ costs). Therefore, they solved the original planning problem for each goal $G \in \mathcal{G}$ as a precomputation, assuming it to be the respective solution to $\mathcal{P}_{\neg O}^G$. This cut the number of off-the-shelf planner calls in half. A second speed-up technique that Masters and Sardina performed applies a stronger assumption that the observation sequence was complete. Then they could *avoid the overhead of generating the augmented fluent and action sets* and instead solve the original planning problem from the current state as the new initial state, adding its cost to the cost of the observation sequence, to solve \mathcal{P}_O^G .

As both a response and follow-up to this research of RaP on the GridWorld domain, Kaminka, Vered, and Agmon focused on motion planning over continuous Cartesian surfaces [144]. To improve the runtime of RaP, Vered and Kaminka previously found two heuristics that prune the set of hypothesized goals \mathcal{G} over time [281].

The first one compares the distance between the most recently observed state/action and the goals. If this distance surpasses a threshold, then they assume that it is unlikely that the agent is actively pursuing that goal; *removing each unlikely goal from the set of hypotheses* reduced the number of off-the-shelf planner calls. The second heuristic evaluates whether the newest observation will impact the ranking of the hypotheses, *skipping computational effort that will return the same general results as before* and reducing the number of planner calls that iteration to 0.

E-Martín, R-Moreno, and Smith later investigated a more domain-agnostic approach that took advantage of the planning graph representation of classical planning problems [68]. This representation allowed them to *quickly compute interaction estimates*, relative weights for various tuples of fluents being true simultaneously, via propagation over the graph with constraints defined via the observation sequence. Due to the alternative representation and lack of solving any problems with off-the-shelf planners, their approach is one of the first re-imaginings of RaP. Their results show that, though faster overall, there were some cases where their approach was slower or incorrect. Due to the differences in implementation, further research is needed to better understand what causes these variations in runtime and performance.

The most recent re-imagining of RaP returns closer to its roots[217]. The key observation Pereira, Oren, and Meneguzzi provide has some relation to Vered and Kaminka’s second heuristic. Specifically, they take advantage of the fact that not all observations drastically impact the most-likely hypothesis unless they indicate a deviation from the set of more optimal solutions to some goal. These particular deviation moments are strongly correlated with landmarks, (features of) states that must hold at some point in order to solve the problem no matter which plan the agent uses. Using this assumption, they created a *special data structure that identifies landmarks and their partial ordering* for each hypothesized goal. There are off-the-shelf planners that can identify these landmarks, which allow their version of RaP

to precompute the data structures and then monitor them—hypothesis ranking is approximated based on a heuristic for how many landmarks the observation sequence satisfies in each data structure.

3.3.3 Goal Recognition Design

Rather than develop an algorithm that improves intent recognition, Keren et al. decided to design environments in ways that reduce the challenge for current intent recognition algorithms [156]. For a typical recognition problem, \mathcal{KB} is constant and embeds the state space within which the agent acts. *Goal recognition design* manipulates this state space to find a \mathcal{KB} that removes ambiguity between solving different tasks. This is measured with respect to \mathcal{H} :

Definition 41. *A sequence s_1, s_2, \dots, s_n is a prefix of sequence $t_1, t_2, \dots, t_{m \geq n}$ if and only if $s_i = t_i$ for all $i \in \{1, 2, \dots, n\}$.*

Definition 42. *Let \mathcal{D} be an automated planning domain, I be an initial state, and \mathcal{G} be a set of goal conditions; then we can define automated planning problems $\mathcal{P}_i = (\mathcal{D}, I, G_i \in \mathcal{G})$ with their respective set of optimal plan solutions Π_i^* . A sequence of actions π is non-distinctive if there exists unique problems \mathcal{P}_1 and \mathcal{P}_2 (that is, $G_1 \neq G_2$) such that π is a prefix of both some optimal plan $\pi_1^* \in \Pi_1^*$ and some other optimal plan $\pi_2^* \in \Pi_2^*$.*

Definition 43. *The worst case distinctiveness (WCD) is the greatest amount of ambiguity possible between optimal plans for two different goals given an automated planning domain \mathcal{D} , initial state I , and set of goal conditions \mathcal{G} . That is,*

$$wcd(\mathcal{D}, I, \mathcal{G}) = \max_{\pi \in \Pi_{\mathcal{D}}^{I \rightarrow \mathcal{G}}} |\pi|$$

where $\Pi_{\mathcal{D}}^{I \rightarrow \mathcal{G}}$ is the set of all non-distinctive paths for \mathcal{D} , I , and \mathcal{G} .

Greater WCD measurements mean that it is possible for an agent to appear ambiguous by executing a plan that is illegible for a long time between at least two different goal conditions. This increases the difficulty of a recognition task because the observing agent cannot confidently identify to which set of plans the observations match. On the other hand, lesser WCD measurements imply that there is little overlap between each goal condition’s optimal plans and *the environment enforces legibility rather than legibility being an agent’s choice*. WCD does assume that the agent acts optimally, which is clearly an unrealistic assumption. This was accounted for in an extension the following year [157]. Goal recognition design with respect to stochastic action outcomes [284] uses similar definitions to those above with policies instead of plans.

It is typically the case that the hypothesized goal conditions cannot be altered, but the state space S and initial state I can. Though this is the end result that will be created, goal recognition design’s reduction algorithm actually removes individual grounded (substituting parameters to become very specific) actions from the set of all actions A . After searching for the reduced set of actions A' that yields an optimal WCD, a *designer can modify the environment* to ensure that the preconditions for all actions in $A - A'$ are always violated. It suffices to prevent entry into states in which any $a \in A - A'$ is applicable, but modifying S can make the enforcement more subtle. For example, in a navigation task where there are doors at the end of the left and right sides of a hallway, actions walking down the center of the hallway should be removed to reduce ambiguity. A designer then prevents agents from walking down the center of this hallway by forcing a forked path with a crowd-control barrier (connectable poles found at the movies, airport security, and other places where people line up and form long queues).

3.4 Applications of Recognition in Videogames

Plan recognition with probabilistic bounds has been applied to StarCraft to identify HTN-represented player strategies [266]. Poo Hernandez, Bulitko, and Spetch [224] recognized themes from player choices in a choose-your-own adventure story to guide the events so that their presentation better connected emotionally with the player. A MDP whose actions represent the player’s choices tracked these emotions. Likewise, predicting a player’s intent of where to hide in a first-person shooter game allows computer-controlled agents to better decide where to go [267]. Using stacked denoising autoencoders rather than traditional clustering, Min et al. [192] identified general tasks that players were performing in an open-ended educational game that did not explicitly reveal the goals to the players. Underlying user goals in educational simulators have also been recognized using another HTN-like representation called an exploratory grammar [195]. However, more than single-player strategies can be identified. Hajibagheri et al. [110] studied logs from massively multiplayer on-line games in order to study the formation and evolution of ad-hoc groups.

3.5 Concluding Remarks: A Unification of PAIR

In order to focus on what each type of recognition shares and how they complement each other, we need a single context in which they all apply. As discussed throughout this chapter, PAIR revolves around understanding other agents from their actions (whether observed directly or through changes to the world). Thus we will consider the *generative planning context* in which R_{ed} selects actions that are then performed while R_{ing} observes [83]:

1. We assume that R_{ed} generally uses some automated planning algorithm \mathcal{A} to solve its problems. We will also assume that R_{ed} has some degree of awareness so that R_{ed} knows about the surrounding environment and can reasonably identify the current state of the world.

2. Suppose that R_{ed} now has a goal G that must be satisfied, either directly assigned by some agent or derived from some personal desire [231] or interactive task [82].
3. Using the assumptions, R_{ed} is able to construct an automated planning problem \mathcal{P} using the current world state (or belief if there is partial observability) for I , knowledge about the environment for \mathcal{D} , and the given G .
4. R_{ed} now uses \mathcal{A} to solve \mathcal{P} . Depending on \mathcal{A} , the plan or policy π can be a sequence of actions, have contingencies, only map a subset of the state space, be a task network, etc.; what matters most is that R_{ed} now has instructions for how to act.

This process provides insight into two components of recognition problems: \mathcal{KB} and \mathcal{H} . Specifically, the *knowledge base is primarily built upon the assumptions about R_{ed}* and the *hypotheses are derived from the possible ways that a goal can be assigned*. Because we do not know exactly what R_{ed} knows or is thinking (if we did, then recognition would be a more trivial task), these are designed for R_{ing} to use as models for each observed agent. Interactive systems that combine recognition and planning sometimes impose R_{ing} 's own knowledge on those it observes [174, 176, 175, 82] such as shared planning algorithms and/or plans, and others use unique models for R_{ing} and R_{ed} [90]. However, the thought process alone is not often sufficient for the recognition problem because R_{ing} needs to observe the actions itself. This progresses the process above to R_{ed} 's execution of actions and how they affect the world state:

5. Without any form of intervention, only the ‘natural physics’ of the environment apply as a *closed system*. We can represent this closed system as a Markov chain (see Section 3.2.1) because it is usually not the case that every physics calculation is accounted for. In a deterministic world where they are perfectly

predictable, each row of the transition matrix is simply a vector of all 0’s except for a single entry with value 1 for the guaranteed state transition.

6. When R_{ed} performs a particular action $a \in A$ that manipulates the environment, a ’s effects will change the world state with respect to its own transition matrix. As a generalized case, consider a corresponding Markov decision process’s state transition function T that yields the distribution over states given a current state and the action performed. Then we have a new Markov chain for the environment while this action is being performed; this should also account for the intervention of ‘natural physics’ alongside R_{ed} ’s action. If the action is deterministic alongside the world, then the transition matrix rows will again be vectors of all 0’s and a single 1.
7. Observations are restricted by R_{ing} ’s available sensors. They might provide information about some part of the current state such a subset of state features F [252] (hardware sensor readings or environment descriptions), the action that was performed [150] (a story or broadcast), or R_{ed} ’s underlying thought process [194, 90, 76] (querying the agent directly).

The observations about the other agents and/or the environment are thus dependent on R_{ing} ’s sensing capabilities and the state of the world. This implies that the sequence of observations \mathcal{O} are constrained snapshots of the world state. Because the state transitions with respect to one of many Markov chains, we can view a short-term version of this context as a HMM. R_{ed} ’s current action $a_r \in \pi$ determines which transition matrix to apply to the latent states, and R_{ing} extracts some information from these states to get some $o_i, \dots, o_{i+j} \in \mathcal{O}$. Then the action changes to a_{r+1} based on what π says to do next, which also changes the transition matrix between the latent states while R_{ing} perceives $o_{i+j+1}, \dots, o_{i+j+k} \in \mathcal{O}$. This continues until R_{ed} satisfies G .

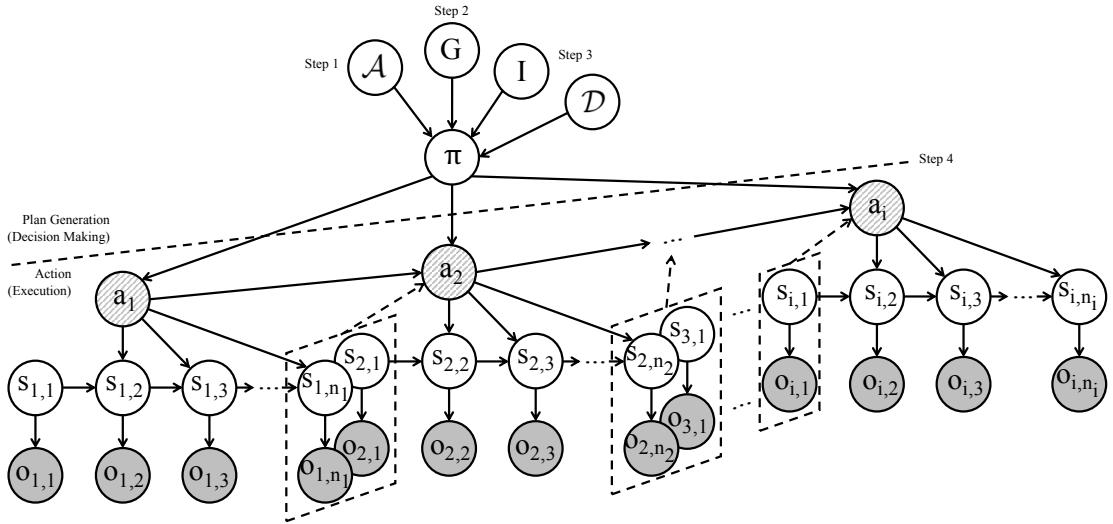


Figure 3.4. A visualization of R_{ed} 's decision making and action execution processes. Directed edges indicate a dependency between two components, but dependencies are not necessarily probabilistic. The dashed boxes indicate the same variable under two different names to bridge the HMMs: $s_{k,n_k} = s_{k+1,1}$. Shaded nodes indicate observations. The partially shaded action nodes a_1, a_2, \dots, a_i are the executed actions according to π , which *might* be observed depending on the observation inputs. Plan recognition typically identifies π and intent recognition typically identifies G or some a_{i+j} where $j \geq 1$. \mathcal{A}, \mathcal{D} , and I are not shaded nodes because they are assumed, rather than observed.

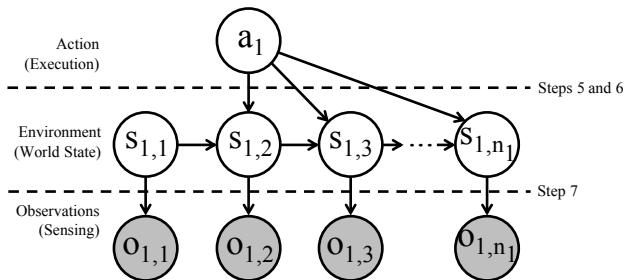


Figure 3.5. A focus on a single action execution HMM from Figure 3.4, broken down into its steps. Activity recognition typically identifies a_1 given $o_{1,1}, \dots, o_{1,n_1} \in \mathcal{O}$. These observations can range from raw signal data to discrete actions.

Given a subsequence of \mathcal{O} over the duration of a single action, activity recognition with simple-action hypotheses $\mathcal{H} = A$ can be performed in a similar fashion to early works in speech recognition using HMMs [87]. Specifically, \mathcal{O} and \mathcal{KB} provide enough information to infer the sequence of states that generated the observations. This sequence of states provides information about the transition matrix that defines the Markov chain, and the transition matrix is more likely associated with certain actions in A . Likewise, given a sequence of these performed actions either directly observed or obtained via lower-level activity recognition, we have a new observation sequence where each $o \in \mathcal{O}'$ also satisfies $o \in A$. Then higher-level activity recognition with complex-action or macroaction hypotheses can be performed based on the transition function between inferred states in which the observed actions are applicable.

When there are no further abstractions over the actions, or if there is a task hierarchy in \mathcal{KB} , the action sequence serves as observations \mathcal{O}'' for plan recognition. This is because \mathcal{O}'' is effectively a subsequence of π , which R_{ed} is using to solve G . \mathcal{A} is assumed to be included in \mathcal{KB} , and R_{ing} had some degree of knowledge about the world state when it began observing so that $I \in \mathcal{KB}$ as well. Then we only need to define \mathcal{H} based on the environment's context and possible plans.

If we instead define \mathcal{H} to be sets of possible goal conditions (ideally including G), then we are performing intent recognition to identify the motives behind why R_{ed} computed π in the first place. If we do not have enough observations for such an all-encompassing recognition problem, then we can also try to predict upcoming actions in π that come after the last action in \mathcal{O}'' using intent recognition.

From this perspective of *reverse-engineering the plan generation and execution steps*, all the types of recognition in PAIR rely on each other in order to completely understand the observed situation. An illustration of these interdependencies with respect to the HMM, higher-order latent variables, and step-by-step procedure are shown in Figures 3.4 and 3.5.

CHAPTER 4

INTEGRATION OF ACTIVITY RECOGNITION AND APPROXIMATE PLAN RECOGNITION

Think of it like a flipbook; so each page’s picture is a word. If people can look at a bunch of words and say, “that’s their common topic,” then people can probably look at a bunch of pictures from an animation reel and say the same thing.

— Richard (Rick) G. Freedman,
justifying the rationale behind his Masters research

Section 3.2.2 introduced the use of LSA for activity recognition, but this is limited to shorter sequences of sensor readings that represent a single action. However, just as extending HMMs into a hierarchy also extends their recognition of single actions to full tasks (explained in Section 3.2.1), we hypothesize that LSA methods whose models contain a hierarchy can recognize more than single actions. Many LSA approaches specifically use topic models, which apply a bag-of-words (BOW) assumption that justify applying BOW models (see Definition 37):

Definition 44. *The bag-of-words assumption states that each input (word, sensor reading, etc.) is sampled independently of the others. That is, for all $i \in \mathbb{N}$ and $j \in \mathbb{N} \setminus i$, $P(X_i = x | X_j) = P(X_i = x)$.*

We consider the following reasons why a BOW assumption applies to observation sequences based on observed agent R_{ed} ’s plan execution:

- A plan π is described by its actions, but partial ordering allows some actions to be *performed interchangeably*. Thus two executions of the same plan might have distinct action sequences, but the frequency counts of each action’s execution

are the same. Viewing the frequencies as a distribution over the set of actions A , these distinct executions of the same plan appear identical.

- An agent is rarely required to execute its plan perfectly, and the environment can introduce distractions that might take the agent’s focus away from performing π . These diversions are *small anomalies* that do not affect the plan itself, and their independence from the actual plan’s actions enforces this notion. The frequency counts of such actions should be small enough that the corresponding distribution over A is unaffected.

Based on the above justification, this chapter *explores how* to employ topic models for activity recognition. The approach introduces a convenient side effect that implies that we can perform approximate plan recognition simultaneously with the activity recognition algorithm, but its potential still remains to be studied. However, many challenges exist for the activity recognition aspect alone, including sensor representations and a challenge to the BOW assumption—we will introduce them with deeper investigations and theoretical discussion. Even with the recent transition of the activity recognition field towards deep-learning approaches, these challenges still persist because they are often *analogues to non-deep machine learning models* that replace some components with deep neural networks.

4.1 Topic Model Approach and Analogy to Recognition

Of the probabilistic topic models available in the literature, we use latent Dirichlet allocation (LDA) [26], which is illustrated in Figure 3.3. A popular unsupervised machine learning model at the time this research was conducted, LDA’s corresponding generative process yields a set of D documents from a set of T topics:

- Each topic is a distribution $\vec{\phi} = \{\phi_1, \dots, \phi_T\}$ over the set of all word tokens in a V -token vocabulary. Thus for some topic $t \in \{1, 2, \dots, T\}$, $\phi_t(v) = P(\text{word} = v | \text{topic} = t)$ and $\sum_{v=1}^V \phi_t(v) = 1$.
- Each document d contains N_d tokens, written as a text sequence $\vec{w}_d = [w_{d,1}, w_{d,2}, \dots, w_{d,N_d}]$. As each word has some semantic meaning within d , a hidden transcription of the text is the sequence of topics pertaining to each word $\vec{z}_d = [z_{d,1}, z_{d,2}, \dots, z_{d,N_d}]$. All the documents are described topically (“the document is about...”) as a distribution $\vec{\theta} = \{\theta_1, \dots, \theta_D\}$ over T . Thus $\theta_d(t) = P(\text{topic of word} = t | \text{document} = d)$ and $\sum_{t=1}^T \theta_d(t) = 1$.
- Entries of $\vec{\phi}$ and $\vec{\theta}$ are drawn from a Dirichlet distribution with a hyperparameter in $H = \{\alpha\vec{m}, \beta\vec{n}\}$. Each hyperparameter is a product of a scalar pseudocount $\alpha, \beta \in \mathbb{R}^{\geq 0}$ and vector prior mean $\vec{m} \in \mathbb{S}^D, \vec{n} \in \mathbb{S}^V$ where \mathbb{S}^k is the k -dimensional simplex [260].

As an analogy, we view a plan execution as a single document, an action in the plan as a topic, and a single sensor reading as a word token. Then a plan execution appears to be an observed sequence of sensor readings, but is actually composed from a distribution over actions. Each action is consequently a distribution over sensor readings, which *introduces a hierarchy of distributions that describe the plan execution* in a manner similar to Section 3.5’s unification of PAIR approaches.

Formally, agent R_{ing} ’s sensor observes R_{ed} executing various plans $\pi_1, \pi_2, \dots, \pi_D$, receiving data such that only the low-level signal tokens

$$\vec{w} = \{[w_{1,1}, \dots, w_{1,N_1}], [w_{2,1}, \dots, w_{2,N_2}], \dots, [w_{D,1}, w_{D,2}, \dots, w_{D,N_D}]\}$$

are recorded. Processing this sequence of signal data is like reading words in text documents. We then need to infer the latent activity variables per recorded token \vec{z}

that will serve as our activity recognition system. Inferring \vec{z} allows us to approximate $\vec{\theta}$ as ratios of frequencies, which serves as an approximation of the executed plans—we do not have the order of actions, but we *can still explain the plan execution as a distribution over actions*.

For training, we use collapsed Gibbs sampling [106] to assign values to \vec{z} . By the conjugate prior property of the Dirichlet distribution, we can integrate-out the parameters $\vec{\phi}$ and $\vec{\theta}$ to approximate the sampling likelihood

$$P(z_i | \vec{z}_{\setminus i}, \vec{w}, H) \propto \frac{N_{z_i}^{\setminus i}(w_i) + \beta n_{w_i}}{\sum_{v=1}^V N_{z_i}^{\setminus i}(v) + \beta} \cdot \frac{N_d^{\setminus i}(z_i) + \alpha m_{z_i}}{\sum_{t=1}^T N_d^{\setminus i}(t) + \alpha} = f_\phi \cdot f_\theta \quad (4.1)$$

where $N_t^{\setminus i} : \{1, \dots, V\} \rightarrow \mathbb{Z}^{\geq 0}$ is the number of times signal token v is assigned action t excluding the token at the sampled index i

$$N_t^{\setminus i}(v) = \sum_{d=1}^D \sum_{j=1}^{N_d} \mathbf{1}(index(d, j) \neq i \wedge w_{d,j} = v \wedge z_{d,j} = t)$$

and $N_d^{\setminus i} : \{1, \dots, T\} \rightarrow \mathbb{Z}^{\geq 0}$ is the number of times a signal token in plan execution d is assigned action t excluding the token at the sampled index i

$$N_d^{\setminus i}(t) = \sum_{j=1}^{N_d} \mathbf{1}(index(d, j) \neq i \wedge z_{d,j} = t).$$

In these counting functions, $\mathbf{1}$ is the indicator function. As $(\sum_{t=1}^T N_d^{\setminus i}(t) + \alpha)$ is a constant value $N_d - 1 + \alpha$ with respect to the sampled variable z_i , it only serves as a normalizing factor of the sampling likelihood and can thus be excluded. Then, to use LDA as an integrated activity recognition and approximate plan recognition procedure, we similarly derive the predictive probability of new observation sequences $\vec{w}' = \{[w_{D+1,1}, \dots, w_{D+1,N_{D+1}}], \dots, [w_{D+D',1}, \dots, w_{D+D',N_{D+D'}}]\}$ given the training data and new observations up to the current one:

$$\begin{aligned} \text{P}(z'_i | \vec{z}, \vec{z}'_{<i}, \vec{w}, \vec{w}'_{<i}, H) &\propto f_\phi^+ \cdot f_\theta^+ = \\ &\frac{N'_{z'_i}^{<i}(w'_i) + N_{z'_i}(w'_i) + \beta n_{w'_i}}{\sum_{v=1}^V (N'_{z'_i}^{<i}(v) + N_{z'_i}(v)) + \beta} \cdot \frac{N'_{D+d}^{<i}(z'_i) + \alpha m_{z'_i}}{\sum_{t=1}^T N'_{D+d}^{<i}(t) + \alpha} \end{aligned}$$

where we first perform Gibbs sampling on the previous new observations’ action assignments $\vec{z}'_{<i}$ as done during training. Although the previous activity assignments were already classified and are likely being used by the decision making components of the proposed PRETCIL framework (introduced in Section 1.4), we still resample them to refine our likelihoods. This will improve the recognition for future actions and get the most likely distribution to approximate each θ_{D+d} for approximate plan recognition.

4.2 Domain-Independent Representations for Signal Data

There are many different ways to represent signal data from sensors, the input for many activity recognition algorithms. The *choice of representation critically affects the trade-off between the value of information and complexity of recognition algorithms*. For example, a text prediction system can learn that its user types ‘you’ after the words ‘where are’ when the GPS is at latitude y and longitude x , clock reads h hours and m minutes, and accelerometer is tilted (θ, ϕ, ρ) degrees. This rule is clearly too specific; not all the information is necessary to learn the higher-level pattern. However, consider instead a factory domain where a robot needs to collect heavy boxes from a coworker and transport them across the facility. Then it is important to know that the coworker stands in a specific posture, d units away from a delivery truck, with boxes located directly in front of them. This level of clarity further applies to what information the system can portray to users as explanations

or interpretations because people may not be aware of the reason for the system’s decisions during their interaction.

For example, RGB-D sensors provide not just a camera image, but also an overlaid infra-red scan to obtain three-dimensional point clouds of their surroundings. Due to their present availability, affordability, and growing popularity [134, 276], we focus on these particular sensors. Although standard camera images can be used, problems with pixelated raster images such as lack of scalability and indiscriminate boundaries about objects have limited the utility of computer vision methods such as *flow field analysis* [283] and *color blotches for object segmentation* [254]. Combining such computer vision techniques with an infra-red scan of distances from the sensor makes it possible to actually detect distinct objects. A panoptic dome consisting of a variety of RGB-D sensors and traditional cameras even enables the distinction to be made from almost any perspective [137].

In particular, two-dimensional camera images can be projected into three dimensions when given pixel depths and/or multi-ocular setups, which aid segmentation and associating individual components’ directions of motion [117, 295]. Zhang and Parker detect regions with the largest change between consecutive recordings within the point cloud for their compressed vector format [296]. Kelley et al. use point cloud representations to actually identify objects with which humans interact in their fixed environment [152]. Others used point clouds to identify sets of objects with pre-registered properties as related to human activities [135, 164]. These works also use the fact that *RGB-D sensors view the human as a collection of body-part objects in the point cloud*. Therefore, an individual’s body can now be denoted through the sensor as a collection of positions and orientations of joints relative to a central world frame [245]. This *stick figure rendering* is also referred to as a *joint-angle representation*. Faria et al. recently used the joint-angle representation to extract 51 features that they found useful for an ensemble of classifiers with dynamically updat-

ing weights [71]. It is possible to also identify human posture data using a variety of other sensors, including traditional motion capture [121] and IMU sensors attached to the body [140], but they are neither as mainstream nor as portable. Depending on the trends of wearable fashions in the future, though, these downsides might become less applicable [154].

We believe that this bare-bones joint-angle representation easily generalizes across multiple domains. That is, posture distributions for a given set of activities will remain consistent across typical domains, but other features such as specific objects might vary by location and time. On the other hand, relying on the joint-angles that construct a raw posture may not help to realize all possible activities. A few representations worth considering, some applying to more than just RGB-D sensor data, thus include:

- **Joint-Angles** Joint-Angles are the rawest format for the posture representation. Often provided as a triplet of Euler angles per joint (θ_j, ϕ_j, ψ_j), but sometimes in quaternions to avoid gimbal lock (an issue involving ambiguity when visualizing the angle), these values derive the matrix rotation transformations that render the stick figure. Translations come from the link lengths, which vary per individual, but the rotations are generally independent of the individual. This makes them ideal for *generalizing the learned activity models to other people* who might interact with the robot, and they allow training data to come from multiple subjects. It is also the reason that the relative distance or position of points in Cartesian space is less ideal for representing the stick figure instead. Although these generalizations are advantageous, more steps are required to extrapolate joint-angles from the sensor’s raw point data—this can be an issue if the recognition system performs under real-time constraints.
- **Derivatives** Just as the rawest format of a posture is the joint-angles, the rawest format between consecutive postures is the change in joint-angles. Mo-

tion is a continuous process, which has been used to find keyframes in motion capture videos and present them in comic book form with ‘whoosh’ lines to summarize the motions [52]. This allows embedding the temporal aspect within the representation to a very small extent, but still keep the dimensionality much smaller than through the use of a sliding window. By computing approximations of the derivative as the difference between joint-angles in consecutive frames, each derivative is still easy to obtain and then use right away (assuming the joint-angles are already computed). However, without a frame of reference such as a starting joint-angle configuration, these derivatives can be very vague and lead to ambiguity during clustering. For example, raising the arm above the head from in front will be identical in this representation to raising the arm to the front from the side.

- **Features** We discuss the extraction of features mathematically from a single human posture in more detail in Section 4.2.2. In short, the posture can be broken down into labeled features such as whether each arm is bent, the back is straight, a leg is raised, etc. If such information is already available, then it may be worth clustering feature vectors directly so that one can inspect the shared features between the modes of a cluster. Furthermore, because many of the features have finite possibilities (e.g. the arm is above the head, at the side, or in front of the body), it is possible to take advantage of the Bayesian Case Model that broke its inputs/observations down in a similar way [158]. A prototype would be a posture with specific parts of the body in fixed relative positions (joint-angle) or simple motions (derivative) while the rest the body remains flexible. This is different from unsupervised feature learning [177], which employs unsupervised learning techniques on a dataset to get clusters as lower-dimensional features. These clusters are used to aid in supervised learning tasks; thus *unlabeled features are obtained for classification with labeled classes*.

Instead, we propose extracting lower-dimensional *labeled features from data to use for classification without labeled classes*.

- **Parametric** In contrast to the above representations that view each frame as a single observable input, it is possible to break the information down into smaller pieces. For example, each joint’s joint-angle or derivative can be viewed as an individual triplet (versus concatenating all the joints’ together) and each feature can also be inspected independently. Although such perspectives cannot be handled by traditional topic models that use single word inputs, we can adjust the generative models with new ‘stories’ that better explain the generation of human postures for various activities. One such model that could be of use for this is Parameterized LDA [80], which runs LDA with a single latent topic across multiple vocabularies simultaneously. In this case, each vocabulary may seem identical such as the interval triplet $[-\pi, \pi]^3$ for a single joint-angle, but it would allow the separate distributions to be *more informative* than a single joint distribution about how each joint is involved in an activity’s description. A uniform distribution over this space would imply that the joint has no significance because its high entropy is not discerning, but a multi-modal peak would imply that some angles or derivatives for this joint are more commonly associated with the clustered activity. Section 4.3 provides additional details about Parameterized LDA.
- **Multimodal** The increase in amount and types of sensors has also led to a boom in multimodal learning. This allows synergy because one sensor may identify things that can complement the information missing from another sensor. Besides using additional sensors to provide more information than the human’s posture, we consider *combining multiple representations of the same data*. Because they all have different interpretations and a strong chance of

recognizing different activities, we should consider taking advantage of multiple perspectives at once to get the ‘complete picture’ at each frame. However, it is also important to consider the dimensionality of the data because the space of inputs will grow drastically as more representations are used simultaneously.

4.2.1 Representing RGB-D Data with Joint-Angles

Related works that learn each activity’s distribution over frames of image-based sensor data using methods such as topic models require a countable vocabulary with some duplicates to avoid a nearly uniform distribution over observed poses. Wang and Mori [283] and Zhang and Parker [296] created codebooks to accomplish this by clustering the inputs in their training set and selecting each cluster’s center as a word in their discretized vocabulary. However, codebooks are difficult to generalize because inputs appearing exclusively in the test data are assumed to resemble an input from the training data rather than be considered new. Hence, *it is hard to accommodate new user sensor inputs after training these recognition systems*. We instead propose a *discretization of the joint-angle values* so that novel postures are not subject to being “clustered away” when they are not similar to anything that was observed during training [79].

We discretize the space $[-\pi, \pi]$ for each angle with respect to a *granularity* parameter $\gamma \in \mathbb{N}$. We map each angle φ to integer $0 \leq i < \gamma$ such that $(i / \gamma) \cdot 2\pi \leq \varphi + \pi < ((i + 1) / \gamma) \cdot 2\pi$. This reduces the vocabulary to $\{0, 1, \dots, \gamma - 1\}^{3|J|}$ where J is the set of all joints, which is still large in size for small γ . However, *many of these poses do not represent feasible body postures*; the limitations of each joint’s range of motion will prevent observing joint-angles that include hyperextended limbs, for example. This is analogous to the fact that many combinations of orthographic letters do not form actual words used in a natural language.

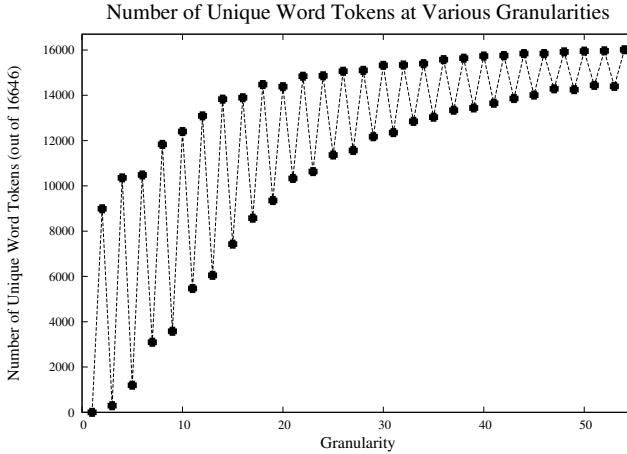


Figure 4.1. A plot of unique word tokens in a collection of forty recorded plan executions at various granularities.

Figure 4.1 plots the number of unique word tokens generated from a small collection of plan execution recordings at various granularities. As expected, increasing the granularity reduces the number of duplicate poses because each interval is smaller. One interesting feature of the plot is the drastic difference between the number of unique tokens based on the parity of the granularity. This phenomenon is explainable through kinematics. When an even granularity is chosen, small joint movements near the vertical axis $\varphi \in [-\epsilon, \epsilon]$ will be assigned to one of two different groups: $(\gamma / 2)$ if $\varphi \geq 0$ or $(\gamma / 2) - 1$ if $\varphi < 0$. On the other hand, an odd granularity will always assign these movements to group $((\gamma - 1) / 2)$. Natural, small body movements and oscillations about the vertical axis, such as an arm swaying slightly at the user's side, will then map between two groups rather than one when γ is even. This yields a significantly larger number of distinct integer combinations compared to odd γ .

4.2.2 Interpretability of Representations

While the joint-angle representations discussed above for RGB-D sensor data are easy to interpret for a single observation, it is typically more difficult for a person to explain how a collection of them (such as the joint-angle-derived stick figures in Figure 4.2) are related. For unsupervised learning methods in natural language pro-

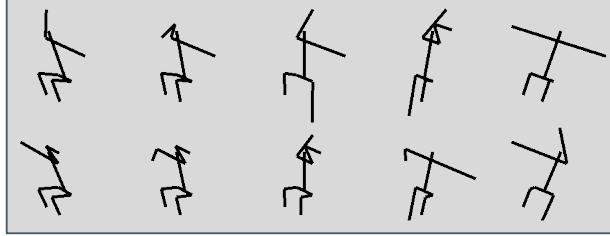


Figure 4.2. These most likely postures for some activity appear to share relations for ‘sitting’ such as legs bent and perpendicular to the torso, but how do we know this for certain?

cessing (NLP) such as the topic models, the learned topic distributions are typically viewed as a collections of common words that are deemed relatable without any further explanation. The *domain expert* who collected the data and/or ran the algorithm *interprets the learned distributions’ modes*, and others can confirm the interpretations via the *normative semantic definitions* found in natural languages. However, such semantic definitions do not yet exist for many sensor data inputs.

The challenge with distributions that unsupervised PAIR algorithms generate is that users cannot easily identify or agree on what was recognized. Hence a *qualitative, feature-based representation for human interpretability is just as important as a quantitative one for algorithmic use and evaluation*. With a little effort, like knowledge engineers developing a logic-based factored representation of a domain for autonomous planning, we can derive qualitative features using the quantitative information.

For example, an elbow joint might be considered bent if the angle between the upper and lower arm links is in the interval $[0, 3\pi/4]$ and straight if it is in the interval $(3\pi/4, \pi]$. Given lengths for each link in the observed agent’s body, the joint-angle measurements from the shoulder to the elbow, and the joint-angle measurements from the elbow to the wrist, we can assume that the shoulder is at coordinate location $\overrightarrow{\text{shoulder}} = (0, 0, 0)$ and then compute the positions of the elbow $\overrightarrow{\text{elbow}}$ and wrist $\overrightarrow{\text{wrist}}$ using homogeneous translation and rotation transformations. With these

coordinates, the law of cosines allows us to find the angle between the upper and lower arm:

$$\cos(\angle elbow) = \frac{-|\overrightarrow{shoulder} - \overrightarrow{wrist}|^2 + |\overrightarrow{elbow} - \overrightarrow{shoulder}|^2 + |\overrightarrow{wrist} - \overrightarrow{elbow}|^2}{2 \cdot |\overrightarrow{elbow} - \overrightarrow{shoulder}| \cdot |\overrightarrow{wrist} - \overrightarrow{elbow}|}.$$

This quantitative angle is now translatable into a qualitative feature for whether the arm is bent [81]. Figure 4.3 illustrates such labels throughout a single posture. We can use this approach to actually formulate sensor data directly into a factor-based representation. A SAS+ state variable's assignment would be based on the interval(s) containing the computed quantities. Besides these exact assignments, there are two ways that we can instead generate *distributions over the values*:

1. Instead of a single cut-off partitioning the intervals for each value assignment, we can *probabilistically interpolate* within an interval about the cut-off. In the example for a bent elbow, we could select some $\epsilon \in \mathbb{R}^{>0}$ and then assign probability mass $P(\text{elbow bent}) = (3\pi/4 + \epsilon - \angle elbow) / 2\epsilon$ when $\angle elbow \in [3\pi/4 - \epsilon, 3\pi/4 + \epsilon]$.
2. If we apply a granularity parameter γ to discretize a continuous interval of values, then we can sample over the interval of continuous values to determine a distribution over the features for each integer. The sampling would thus approximate ratio

$$\frac{|\{v \in V_\gamma(i) | \angle elbow_v \in [0, 3\pi/4]\}|}{|V_\gamma(i)|}$$

where $V_\gamma(i)$, the set of all signal values that map to integer i when discretized with granularity γ , is sampled uniformly¹.

¹A non-uniform distribution will be more accurate if certain signal values are more common than others, much like how many joint-angle postures are physically impossible.

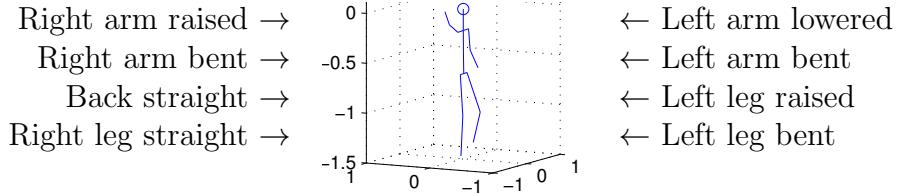


Figure 4.3. Some human-interpretable features for a posture.

A complete factor-based representation listing all the features of a single observation is similar to the feature vector Kim, Rudin, and Shah create for their Bayesian Case Model [158]. To aid users interpreting collections of clustered items, their features highlight one element from each cluster as the representative *prototype* and emphasize the specific features that the prototype shares with the rest the cluster. They also used this prototype as a *heuristic to improve the quality of their learned distributions* when reclustering. Hayes and Shah [115] recently introduced another explainable activity recognition algorithm that uses features and temporal information to produce simple sentences explaining what the algorithm is observing for classification.

The Bayesian Case Model [158] mentioned above describes each cluster k using a prototype $p_k \in V_k = \{v \in \{1, 2, \dots, V\} \mid \exists d, i. w_{d,i} = v \wedge z_{d,i} = k\}$ and subspace feature vector $\omega_k \in \{0, 1\}^{|F|}$ where F is the set of features. Specifically, p_k is a representative example of all the inputs in the cluster with respect to the subspace feature vector, which indicates the important features $\{f \in F \mid \omega_k(f) = 1\}$. This model learns the clusters, prototypes, and subspace feature vectors simultaneously using a generative approach that assumes that, for a given cluster k , all sensor inputs in V_k and the current prototype should share the same assigned value for all the important features' variables. Values assigned to the variables of unimportant features (where $\omega_k(f) = 0$) may be arbitrary. Then the Bayesian Case Model's optimization criteria is to *minimize the Hamming distance of important features between the prototype and*

all inputs in the cluster

$$d(p_k, v \in V_k) = \sum_{f \in F} \mathbf{1}(p_k(f) \neq v(f)) \cdot \omega_k(f).$$

For our feature-based representation, we can also derive *interpretable features to describe the learned distributions* over signal tokens ϕ_k . From the activity recognition perspective, we want to identify the qualitative features that best describe the majority of the sensor readings that each cluster, effectively an instance of samples from the learned distribution, represents. We use the same set of features that describe each individual signal token. Our description measures the relevance of each feature over all tokens with respect to their proportion of the cluster, preferring the *relevance of more common signal tokens*. Then our description does not need to exist in the training data like a prototype, instead representing an ‘average’ signal token that would hypothetically belong in the cluster. Our optimization criteria is to *minimize the distance of relevance between the feature descriptor and all feature vectors of tokens in the cluster, allowing slack for less common inputs*

$$d(\vec{x}_k, \vec{x}_{v \in V}) = \sum_{f \in F} |x_k(f) - x_v(f)| \cdot \phi_k(v)$$

where \vec{x}_k is the vector of values assigned to each feature for cluster k and \vec{x}_v is the vector of values assigned to each feature for signal token v . Using the set of \vec{x}_v for all $v \in \{1, 2, \dots, V\}$, we propose the following three approaches for computing \vec{x}_k ; each has its own advantages and disadvantages.

- **Expected Value** If we consider each \vec{x}_v as a feature vector, then each token v is located at some point within the $|F|$ -dimensional simplex $\mathbb{S}^{|F|}$. Because v also has probability mass $\phi_k(v)$ within cluster k , we can describe the most relevant

features of the most common inputs in k as the expected value of each feature $f \in F$:

$$\overrightarrow{x}_k = \sum_{v=1}^V \phi_k(v) \cdot \overrightarrow{x}_v.$$

This method is most similar to explicit semantic analysis (ESA) [86] where a sentence is described as the sum of the feature vectors for each of its words (the features for a word are measurements of relation to a set of keywords). This is similar to a distribution over the set of words that is proportional to the sentence's word frequencies. Although simple to compute, this approach is naïve because it simply finds the *weighted union of features*. Thus a single v with a large $\phi_k(v)$ would contribute all its features to the cluster's feature descriptor even if no other objects with considerable mass share some of them.

- **Agglomerative Clustering** As an alternative to the union of features found in the expected value approach, we also introduce a method that includes the intersection of features. Agglomerative clustering hierarchically builds a partition of the set of signal tokens $\{1, 2, \dots, V\}$ such that each subset's signal tokens share like features, beginning with singleton subsets that contain each token separately and then iteratively combining similar subsets until the larger partitions are too distinct to combine. The likeness between two subsets $C_1, C_2 \subseteq \{1, 2, \dots, V\}$ with respect to cluster k is measured using

$$d(C_1, C_2) = \left| \sum_{v \in C_1} \phi_k(v) - \sum_{v \in C_2} \phi_k(v) \right| \cdot \|\overrightarrow{x}_{C_1} - \overrightarrow{x}_{C_2}\|_1$$

where \overrightarrow{x}_{C_i} is the feature descriptor for subset C_i . d is not a metric because a distance of 0 does not guarantee that the two subsets are equal. However, it does emphasize which *pairs of subsets would have a smaller degree of change when their individual feature descriptors are intersected*. The comparison of probability mass within ϕ_k is also used to avoid placing inputs with lesser representation

of cluster k into the same partitions as inputs with greater representation of cluster k , following our optimization criteria above.

In contrast to the union of feature vectors, which resembles an expected value, we define the intersection of feature vectors for elements of combined subclusters C_1 and C_2 as

$$\overrightarrow{x_{C_{1,2}}} = \bigodot_{v \in C_1 \cup C_2} \overrightarrow{x_v}$$

where \odot is element-wise multiplication. This consequently describes combined subcluster $C_{1,2}$. Intersection might be too strong because it has the opposite problem of the union: a single token with large probability density could lack one feature ($x_v(f) \approx 0$) that is greatly relevant to the remaining sensor inputs with significant probability. This feature would consequently be excluded from the cluster's set of describing features. To address this, we further introduce the unweighted average of subclusters as a *soft intersection* that accounts for the number of objects sharing the presence/lack of a feature. We compute

$$\overrightarrow{x_{C_{1,2}}} = |C_1 \cup C_2|^{-1} \cdot \sum_{v \in C_1 \cup C_2} \overrightarrow{x_v}$$

as the soft intersection of the feature vectors of the elements of combined subsets C_1 and C_2 . With respect to interpretability, 0 means that a feature is not relevant to any inputs representing the cluster, 1 means that a feature is relevant to all inputs representing the cluster, and a value of 0.5 means that a feature is not useful for a description since it is equally present and absent from the inputs representing the cluster.

When the distances between subsets become too great, we have partitions expressing unique features that each describe the cluster. We hypothesize that the expected value over the partitions' feature vectors will be more informative for describing clusters than the expected value over each sensor input's feature vec-

tors. However, there are also advantages to using a disjunction of the subsets' weighted feature vectors to describe the cluster: any exclusive-or relationships between features can be obscured by combining them. For example, using the RGB-D case, if one set contains postures with "the left arm raised and the right arm not raised" and the other set contains postures with "the left arm not raised and the right arm raised," then this may imply that the cluster contains postures with exactly one arm raised—adding these together for an expected value would instead yield a compromise that the right and left arms might or might not be raised (an irrelevant feature weight near 0.5).

- **Supervised Learning** The last approach acknowledges the fact that some supervised learning output representations such as decision trees learn interpretable functions, which is also a motivation for wordification [218, 219]. For example, the traversal from a decision tree's root to any leaf node produces a conjunction of conditions that explains the leaf's classification assignment. If we consider every input, including duplicates, as a separate data point, then we have supervised inputs $\overrightarrow{x_{w_{d,i}}}$ with assigned outputs $z_{d,i}$ from our unsupervised learning model. We can use off-the-shelf supervised learning algorithms to learn a function mapping between each signal token's feature vector and its associated cluster rather than independently computing feature descriptors for each cluster. Changuel and Labroche [46] used such off-the-shelf classifiers to learn missing metadata values from present ones to improve categorization of library resources. The only limitation is that each algorithm has a specific type of function that it can learn. For example, decision trees can only learn perpendicular partitions of the feature space. Thus different supervised learning methods will yield different justifications for the unsupervised algorithm's label assignments.

4.3 Using Temporal and Spatial Relations

We now consider ways to improve the performance of unsupervised activity recognition (and approximate plan recognition) techniques via temporal and object relations in addition to just signal tokens, such as postural data from RGB-D sensors. Temporal relationships can help recognize activities with cyclic structure and are often implicit because plans have degrees of ordering actions. Relations with objects can help disambiguate observed activities that otherwise share a user’s posture and position.

Even though much more information such as temporal and spatial relations is available, our initial approach used only the postural information from the RGB-D sensor because input representations that are too specific limit generalizability across domains (see Section 4.2 for a deeper discussion). However, *activities can describe more than just one’s pose or motion*. Zhang and Parker [296] as well as Koppula and Saxena [164] developed activity recognition models that take into account both temporal and spatial relations. Instead of combining all these factors into a single token representation like they do, we propose *extending traditional models with conditionally independent modular components that each account for a distinct aspect of the available information*. Figure 4.4 summarizes the model extensions that will be discussed below.

This multi-modal approach still allows generalized recognition models where modules may be added, removed, or interchanged depending on the environment and available sensor information. Additionally, the *aspects that these modules address can be left constant or toggled off to speed up computations when real-time constraints apply*. This potential benefit involves future research on metareasoning to determine when the modules would provide sufficient information efficiently; it will not be explored in this dissertation.

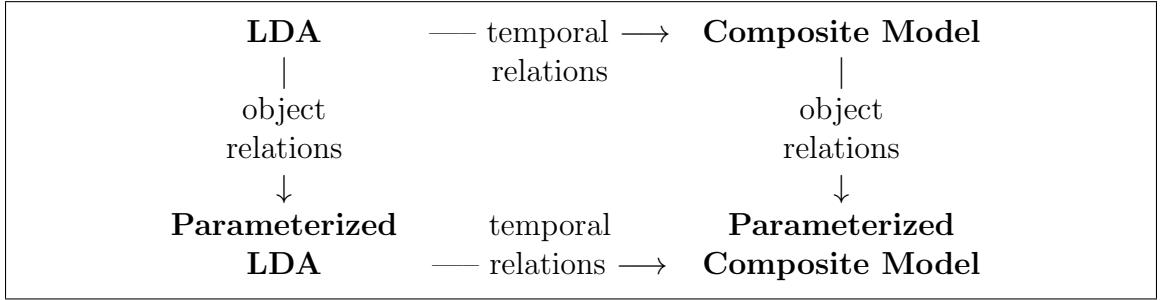


Figure 4.4. Proposed models enhancing LDA, based on their additional information.

```

1 for each topic  $t \in \{1, \dots, T\}$  do
2   | draw  $\phi_t \sim \text{Dirichlet}(\beta \vec{n})$ 
3   | draw  $\Omega_t \sim \text{Dirichlet}(\gamma \vec{o})$ 
4   end
5 for each state  $c \in \{2, \dots, C\}$  do
6   | draw  $\varphi_c \sim \text{Dirichlet}(\delta \vec{u})$ 
7   end
8 for each state  $c \in \{0, \dots, C\}$  do
9   | draw  $\xi_c \sim \text{Dirichlet}(\epsilon \vec{\pi})$ 
10  end
11 for each document  $d \in \{1, \dots, D\}$  do
12   | draw  $\theta_d \sim \text{Dirichlet}(\alpha \vec{m})$ 
13   | assign  $s_{d,0} \leftarrow 0$ 
14   | for each index  $i \in \{1, \dots, N_d\}$  do
15     |   | draw topic  $z_{d,i} \sim \theta_d$ 
16     |   | draw state  $s_{d,i} \sim \xi_{s_{d,i-1}}$ 
17     |   | if  $s_{d,i} = 1$  then
18       |   |   | draw word token  $w_{d,i} \sim \phi_{z_{d,i}}$ 
19     |   | end
20     |   | else
21     |   |   | draw word token  $w_{d,i} \sim \varphi_{s_{d,i}}$ 
22     |   | end
23     |   | for each index  $j \in \{1, \dots, K_{w_{d,i}}\}$  do
24       |   |   | draw parameter  $p_{d,i,j} \sim \Omega_{z_{d,i}}$ 
25     |   | end
26   | end
27 end

```

Algorithm 2: Generative Process for Parameterized Composite Model

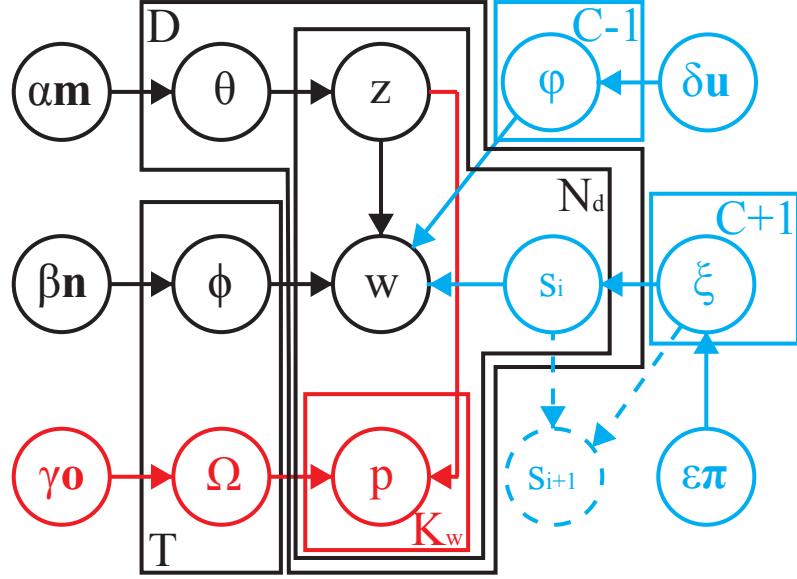


Figure 4.5. Graphical model representation of the parameterized composite model with LDA, the composite model, and parameterized LDA as subgraphs.

4.3.1 Capturing Temporal Relations with the Composite Model

NLP research, an area suggested to have much in common with plan recognition [92, 79], has raised similar concerns about assuming only local temporal dependencies between word tokens in text.

Definition 45. *The k -degree Markov assumption (the degree is often omitted when $k = 1$) states that each input (word, sensor reading, etc.) is sampled independently of all others except the x -most recent inputs others. That is, for all $i \in \mathbb{N}$, $P(X_i = x | X_{i-1}, X_{i-2}, \dots, X_0) = P(X_i = x | X_{i-1}, \dots, X_{i-k})$. We note that $k = 0$ is the BOW assumption from Definition 44, which ignores local temporal dependency.*

Local temporal dependencies enforced by models such as HMMs place a strong emphasis on *syntactic properties of phrases without any consideration of semantics*. The global dependencies that topic models enforce instead emphasize the *semantic features of text without acknowledging its syntax*. Griffiths et al. developed the composite model [107] to bring HMMs and LDA together for a single model that

takes both syntax and semantics into account. We suggest that the composite model might also be useful for analyzing sequences of signal token observations for improved activity recognition and approximate plan recognition.

The composite model integrates LDA with a HMM over C hidden states by setting *one of the states to call LDA* for selecting a semantic word. We remain consistent with Griffiths et al.’s notation and let state 1 call LDA. The remaining states contain their own distributions $\vec{\varphi} = \{\varphi_2, \dots, \varphi_C\}$ over the vocabulary of word tokens $\{1, 2, \dots, V\}$ so that they may select words during the document generation.

They empirically supported that most of $\vec{\varphi}$ ’s probability mass is found about *stopwords*, tokens with very high frequencies that usually have to be removed before running LDA. Without the removal of stopwords, their high frequency causes them to appear with decent probability mass in almost every learned topic by random chance. Stopwords typically serve a syntactic purpose in documents rather than a semantic purpose, which is what LDA captures. The HMM *captures structure through its dependency on the previous state in the latent Markov chain* overseeing the document’s composition. The transition functions $\vec{\xi} = \{\xi_0, \xi_1, \dots, \xi_C\}$ represents this Markov chain, and the distribution ξ_0 determines its initial state. The black and blue subgraphs in Figure 4.5 form the composite model’s graphical model, which supports the generative process composed of lines 1, 2, 4-7, and 8-17 of Algorithm 2. Unlike the notation, we must use different variable names from Griffiths et al. because our other models in this section have naming conflicts.

In activity recognition, it is possible to encounter signal tokens that act like stopwords if some subset of signal data is very common in the observation sequences [79]. Although these are typically removed prior to training and testing, the framework laid out by these poses can be beneficial for recognition tasks. A transition between certain states might serve as a boundary between actions if the observation sequence needs to be segmented into distinct activities. It also provides an ordering for the

poses associated with activities so that some structures such as loops in plans might become easier to identify.

Similar to LDA, we only observe the poses \vec{w} extracted from the postural data. Unlike LDA, we now have to infer two latent variables per observation: the latent activity variables \vec{z} and the latent HMM state variables \vec{s} . Because Gibbs sampling only samples one random variable at a time, we must alternate between sampling z_i and s_i . Like other topic model computations, we use the conjugate prior to approximate the sampling likelihood of the latent activity:

$$\begin{aligned} P(z_i | \vec{z}_{\setminus i}, \vec{w}, \vec{s}, H) &\propto f_{\phi}^{\mathbf{1}(s_i=1)} \cdot f_{\varphi}^{\mathbf{1}(s_i \neq 1)} \cdot f_{\theta} \\ &= f_{\phi}^{\mathbf{1}(s_i=1)} \cdot \left(\frac{N_{s_i}^{\setminus i}(w_i) + \delta u_{w_i}}{\sum_{v=1}^V N_{s_i}^{\setminus i}(v) + \delta} \right)^{\mathbf{1}(s_i \neq 1)} \cdot f_{\theta} \end{aligned}$$

where $\mathbf{1}(x)$ is the indicator function that returns 1 if x is true and 0 otherwise, $N_c^{\setminus i} : \{1, \dots, V\} \rightarrow \mathbb{Z}^{\geq 0}$ is the number of times token v is assigned HMM state c excluding the token at the sampled index i

$$N_c^{\setminus i}(v) = \sum_{d=1}^D \sum_{j=1}^{N_d} \mathbf{1}(index(d, j) \neq i \wedge w_{d,j} = v \wedge s_{d,j} = c),$$

and $N_t^{\setminus i}$ now only considers tokens assigned to HMM state 1 (i.e., the tokens that LDA generates)

$$N_t^{\setminus i}(v) = \sum_{d=1}^D \sum_{j=1}^{N_d} \mathbf{1}(index(d, j) \neq i \wedge w_{d,j} = v \wedge z_{d,j} = t \wedge s_{d,j} = 1).$$

Because the current HMM state is fixed during this computation,

$f_{\varphi}^{\mathbf{1}(s_i \neq 1)} \cdot \left(\sum_{t=1}^T N_d^{\setminus i}(t) + \alpha \right)$ is constant with respect to the sampled random variable and may thus be omitted from the sampling likelihood. We note that $N_t^{\setminus i}$ and

$\left(\sum_{t=1}^T N_d^{>i}(t) + \alpha\right)$ are part of f_θ 's definition from Equation 4.1. After reassigning the latent activity, we sample the HMM state using the sampling likelihood:

$$\begin{aligned} P(s_i | \vec{s}_{\setminus i}, \vec{w}, \vec{z}, H) &\propto f_\phi^{\mathbf{1}(s_i=1)} \cdot f_\varphi^{\mathbf{1}(s_i \neq 1)} \\ &\cdot \frac{F_{s_{i-1}}^{\setminus i, i+1}(s_i) + \varepsilon \pi_{s_i}}{\sum_{c=1}^C F_{s_{i-1}}^{\setminus i, i+1}(c) + \varepsilon} \cdot \left(\frac{F_{s_i}^{\setminus i+1}(s_{i+1}) + \varepsilon \pi_{s_{i+1}}}{\sum_{c=1}^C F_{s_i}^{\setminus i+1}(c) + \varepsilon} \right)^{\mathbf{1}(i < N_d)} \end{aligned}$$

where $F_{c_1}^{\setminus i} : \{1, \dots, C\} \rightarrow \mathbb{Z}^{\geq 0}$ is the number of times the transition from HMM state c_1 to HMM state c has occurred excluding the transitions both to and from the HMM state at the sampled index i

$$F_{c_1}^{\setminus i}(c) = \sum_{d=1}^D \sum_{j=1}^{N_d} \mathbf{1}(index(d, j) \neq i \wedge s_{d,j-1} = c_1 \wedge s_{d,j} = c).$$

For this sampling likelihood, only $\left(\sum_{c=1}^C F_{s_{i-1}}^{\setminus i, i+1}(c) + \varepsilon\right)$ is constant with respect to s_i so that it may be ignored during the computation. After training, the predictive probability of an observation in \vec{w}' given the training data and new observations up to the current one requires computing the joint probability of both the latent activity and the latent HMM state:

$$\begin{aligned} P(z'_i, s'_i | \vec{z}, \vec{z}'_{\setminus i}, \vec{s}, \vec{s}'_{\setminus i}, \vec{w}, \vec{w}'_{\setminus i}, H) &\propto f_\theta^+ \cdot f_\xi^+ \cdot (f_\phi^+)^{\mathbf{1}(s_i=1)} \cdot (f_\varphi^+)^{\mathbf{1}(s_i \neq 1)} = \\ &f_\theta^+ \cdot \frac{F'_{s'_{i-1}}^{<i}(s'_i) + F_{s'_{i-1}}(s'_i) + \varepsilon \pi_{s'_i}}{\sum_{c=1}^C (F'_{s'_{i-1}}^{<i}(c) + F_{s'_{i-1}}(c)) + \varepsilon} \\ &\cdot (f_\phi^+)^{\mathbf{1}(s_i=1)} \cdot \left(\frac{N'_{s'_i}^{<i}(w'_i) + N_{s'_i}(w'_i) + \delta u_{w'_i}}{\sum_{v=1}^V (N'_{s'_i}^{<i}(v) + N_{s'_i}(v)) + \delta} \right)^{\mathbf{1}(s'_i \neq 1)} \end{aligned}$$

When performing Gibbs sampling on the previous new observations, we again alternate between activity assignments $\vec{z}_{\setminus i}$ and HMM state assignments $\vec{s}_{\setminus i}$.

4.3.2 Capturing Object Relations with Parameterized LDA

Besides using temporal information, recognition systems can benefit from information regarding relations between the observed agents and objects in the environment because the observing agent’s interactions with the user will likely involve handling the same objects. For example, when moving furniture [200], both agents need to handle the same object in order to coordinate carrying it. Furthermore, the object can provide information about the observed user’s plan/action that is not available from their posture and position. For example, the most notable difference in a kitchen environment [254, 60] between cleaning a spill and mopping sauce from a plate is holding a napkin versus using a slice of bread. This has also been addressed in the field of robotics under *tool affordances*, a psychological theory stating that people perceive the functionality of objects based on the object’s features [95].

Our proposed parameterized extension of LDA considers documents whose word tokens may contain a list of parameters. That is, a single word in a document is now a K -ary proposition of the form $w_i(p_{i,1}, \dots, p_{i,K_i})$ where K_i is the arity of token w_i . Due to K_i ’s notation relying on the index, rather than the token itself, the number of parameters may vary between identical tokens to account for the observation instance. We assume that these arguments are elements of a second vocabulary of Q items. Each topic has an additional distribution $\vec{\Omega} = \{\Omega_1, \dots, \Omega_T\}$ over this second vocabulary that is drawn from a Dirichlet distribution with hyperparameter $\gamma \vec{\sigma} \in H$. Thus its graphical model is the black and red subgraphs in Figure 4.5 and lines 1-3, 8, 9, 11, 12, 15, 18, and 19 from Algorithm 2 detail its generative process.

This model essentially runs an additional K_i LDA topic models simultaneously that all share the same topic. That is, we sample each parameter as in LDA, but from a distribution over objects only conditioned on the topic of the current word token Ω_{z_i} . We note that parameterized LDA applies the *BOW assumption to the*

parameters so that each argument is independent of the others and the order does not matter.

For our application of activity recognition, each parameter is an object with which the observed agent R_{ed} is interacting. Identifying which objects should be considered as parameters per token is left for future research. We currently use objects that are within a fixed distance from R_{ed} 's joints based on the RGB-D sensor. Song et al.'s [254] extraction of clauses for recognizing activities with a Markov logic network identifies objects by such proximity to the observed agent's hands. By considering other joints, we can also extract localization information such as one's position in a room (near a refrigerator) and consider items near the head or feet that might become involved as the activity progresses (such as picking up an object from the ground).

Poses and objects together can remove ambiguity that the other one would indicate about the activity alone. For example, Freedman, Jung, and Zilberstein acknowledged that some discretized poses for activities such as squatting and jumping appeared identical because they generated similar signal tokens. If the ground is an object within the vicinity of such a pose, then we are more likely to recognize the activity as squatting than jumping. This is because *objects provide semantic context to the activity and plan*. Inversely, Jain and Inamura [133] explain that more than one activity can use a single object depending on its orientation and utilized affordances. To accommodate the different orientation and/or affordance, it is likely the case that the pose of the agent differs as in their example that uses the back end of a screwdriver as a hammer—the arm would alternate between rising and falling rather than rotating in place.

We assume that our system has a sensor that can perform object recognition in addition to its typical sensor's signal data so that we observe token-object pairs $(\overrightarrow{w}, \vec{p}) = [(w_{1,1}, \vec{p}_{1,1}), \dots, (w_{D,N_D}, \vec{p}_{D,N_D})]$ where $\vec{p}_{d,i} = (p_{d,i,1}, \dots, p_{d,i,K_{w_{d,i}}})$. Hence we still need to infer just the latent activity variables \vec{z} . We continue to use Gibbs

sampling to assign values to \vec{z} using the same techniques described for approximating the sampling likelihood in other topic models:

$$\begin{aligned} \text{P}\left(z_i \mid \vec{z}_{\setminus i}, \overrightarrow{(w, p)}, H\right) &\propto f_\phi \cdot f_\theta \cdot f_\Omega = \\ f_\phi \cdot f_\theta \cdot \prod_{j=1}^{K_{w_i}} &\frac{A_{z_i}^{\setminus i}(p_{i,j}) + \gamma o_{p_{i,j}}}{\sum_{q=1}^Q A_{z_i}^{\setminus i}(q) + \gamma} \end{aligned}$$

where $A_t^{\setminus i} : \{1, \dots, Q\} \rightarrow \mathbb{Z}^{\geq 0}$ is the number of times object q is assigned topic t excluding the parameters in the token-object pair at the sampled index

$$A_t^{\setminus i}(q) = \sum_{d=1}^D \sum_{j=1}^{N_d} \mathbf{1}(index(d, j) \neq i \wedge z_{d,j} = t \wedge \exists k. p_{d,j,k} = q).$$

As in LDA, we can omit $(\sum_{t=1}^T N_d^{\setminus i}(t) + \alpha)$ from the computation because it is constant with respect to the sampled variable z_i . For a new sequence of observed token-object pairs $\overrightarrow{(w, p)}' = [(w_{D+1,1}, p_{D+1,1}), \dots, (w_{D+D', N_{D+D'}}, p_{D+D', N_{D+D'}})]$, the predictive probability of a single observation given the training data and the new observations up to the current one is:

$$\begin{aligned} \text{P}\left(z'_i \mid \vec{z}, \vec{z}'_{\setminus i}, \overrightarrow{(w, p)}, \overrightarrow{(w, p)}'_{\setminus i}, H\right) &\propto f_\phi^+ \cdot f_\theta^+ \cdot f_\Omega^+ = \\ f_\phi^+ \cdot f_\theta^+ \cdot \prod_{j=1}^{K_{w'_i}} &\frac{A'_{z'_i}^{\setminus i}(p'_{i,j}) + A_{z'_i}(p'_{i,j}) + \gamma o_{p'_{i,j}}}{\sum_{q=1}^Q (A'_{z'_i}^{\setminus i}(q) + A_{z'_i}(q)) + \gamma} \end{aligned}$$

As with LDA for activity recognition, we must perform Gibbs sampling on the previous new observations' activity assignments $\vec{z}'_{\setminus i}$.

4.3.3 The Parameterized Composite Model

The parameterized composite model combines parameterized LDA with the composite model by simply changing state 1's call from LDA to parameterized LDA.

However, we assume that the *object parameters only have semantic information* and cannot be used for syntax. Thus the parameters are generated from the latent topic even if the HMM state is not 1. From an automated planning perspective, we interpret this as *passing arguments between consecutive actions* in a plan. When local ordering between actions is necessary, the order is usually important because a subset of the next action’s prerequisites are only satisfied by the effects of the current action. This is the key idea behind causal links in the classic UCPOP planner [216]. Figure 4.5 displays this hybrid graphical model and the entirety of Algorithm 2 explains the generative process.

The parameters are conditionally independent of the token and HMM state given the activity so that we simply multiply the sampling likelihood of the topic by the approximate likelihood of the parameters given the topic:

$$P(z_i \mid \vec{z}_{\setminus i}, \overrightarrow{(w, p)}, \vec{s}, H) \propto f_{\phi}^{\mathbf{1}(s_i=1)} \cdot f_{\varphi}^{\mathbf{1}(s_i \neq 1)} \cdot f_{\theta} \cdot f_{\Omega}$$

Due to the independence assumptions depicted by the directed edges in Figure 4.5, the sampling likelihood for the HMM state does not change from the composite model. The new terms for parameters \vec{p}_i are constant with respect to all HMM state random variables as long as the activity z_i is observed. On the other hand, the joint predictive probability does receive an update based on the observed parameters and token:

$$P(z'_i, s'_i \mid \vec{z}, \vec{z}'_{\setminus i}, \vec{s}, \vec{s}'_{\setminus i}, \overrightarrow{(w, p)}, \overrightarrow{(w, p)'_{\setminus i}}, H) \propto f_{\theta}^+ \cdot (f_{\phi}^+)^{\mathbf{1}(s_i=1)} \cdot f_{\Omega}^+ \cdot (f_{\varphi}^+)^{\mathbf{1}(s_i \neq 1)} \cdot f_{\xi}^+$$

4.3.4 Empirical Evaluation of Alternative Topic Models

We implemented LDA and the three topic models described above such that each component is a module with the same underlying framework, as Figure 4.5 and their computations portray. Although not as efficient as some state-of-the-art implementations of LDA, our implementations offer a basis for a fair comparison between the

methods with respect to their relative runtimes and performance. These factors are important for recognition systems in actual applications because there are often real-time constraints on the machine’s response time. A computer or robot must be able to successfully identify the R_{ed} ’s actions and/or plan before providing a valid response. Hence we focus on the relative trade-offs between performance and runtime for these models in order to investigate the practical extents of including temporal and object information. We assume that relative results would be proportional if more efficient implementations are made using state-of-the-art versions of LDA such as Mallet [190].

For these comparison tests, we used the Cornell Activity Dataset 120 (CAD-120) [15]. It contains 124 recordings of short plan executions (the authors call them activities) with a total duration of approximately eleven minutes. Each RGB-D recording is fully annotated with orientation and position information for the acting agent’s posture, objects used, object affordance labels based on how each object is used in each frame, activity labels, and segmented subactivity labels for training supervised learning-based algorithms.

When converting the dataset to a corpus of documents, we ignore the activity and subactivity labels because our models are intended for unsupervised learning approaches. We use each frame’s orientation data to generate the observed agent’s posture as a signal token using the modified joint-angle representation described in Section 4.2.1 with granularity parameter $\gamma = 21$. This gives us a vocabulary containing $V = 42588$ unique signal tokens out of 65133 total in the corpus. The dataset depicts objects in each frame using a two-dimensional bounding box from the RGB image based on SIFT features without depth. Hence we are only able to identify parameters as objects whose bounding boxes are within 150 millimeters of a joint of the observed user in the x - and y -directions, which we assume accounts for the bounding box not always capturing the entire object. Due to the design of CAD-120,

the lack of the z -direction in these proximity calculations does not greatly affect the list of parameters for each word.

To ensure optimal performance of each topic model with CAD-120, we trained the topic models using a sweep of parameter settings for T and C and then selected the values yielding the greatest log-evidence of generating the training dataset. While hyperparameters α , β , γ , δ , ε , and \vec{m} were optimized throughout the Gibbs sampling process, \vec{n} , \vec{o} , \vec{u} , and $\vec{\pi}$ could not be optimized due to biases they introduced during training. The number of activities T and HMM states C were held constant once initialized. Initial hyperparameter concentration values were always set to $\alpha = T$, $\beta = \delta = 0.02V$, $\gamma = 2Q$ where $Q = 10$, and $\varepsilon = C$; prior means \vec{m} , \vec{n} , \vec{o} , \vec{u} , $\vec{\pi}$ were always set to a uniform distribution. A burn-in period of twenty-five iterations was applied to make sure that the state sequences were truly random before optimizing ε in the transition functions. We trained the models on 99 sequences (80% of CAD-120) and then tested them on the remaining 25. To avoid an anomaly, five such partitions were randomly generated a priori for use in each parameter setting, and the same training and testing partitions were used across all models for a fair comparison. The number of signal tokens in each partition $P1$ through $P5$'s test set is 11388, 12438, 14855, 12406, and 11647 respectively.

4.3.4.1 Runtime Performance

Table 4.1 displays the empirical time to test and train each model in seconds. We also provide the number of Gibbs sampling iterations used to converge to the maximized log-evidence during training since this varied per model and likely impacted the runtime. Because the train-test set choice did not impact runtime significantly, we only present the results with partition $P1$.

Although the training set is larger than the testing set, the testing times take longer for the less computationally intensive models due to the step that resamples

Table 4.1. Elapsed Runtimes for Optimally-Trained Models ($P1$)

	Gibbs (iterations)	Train (seconds)	Test (seconds)
LDA	50	362.398	1318.926
Parameterized LDA	100	1122.416	2411.291
Composite	540	4080.849	2915.388
Parameterized Composite	540	4945.438	3436.523

Table 4.2. Log-Evidence of Test Sets with Optimally-Trained Models

	LDA	Parameterized LDA	Composite	Parameterized Composite
$P1$	-112096.2	-111055.4	-111097.6	-109512.0
$P2$	-123786.9	-123084.8	-122722.0	-122156.1
$P3$	-148437.2	-147803.2	-146915.9	-146763.1
$P4$	-121191.7	-119860.2	-118929.7	-119477.9
$P5$	-115952.3	-114871.8	-113931.9	-114142.1

every newly observed pose before classifying the next observation. These times show that the inclusion of temporal relations can greatly increase the training time needed to perform recognition. On the other hand, the inclusion of object relations appears to increase the amount of time to a lesser degree. Regardless of the degree of increase, any of the models extending LDA take about two times longer or greater to perform recognition.

4.3.4.2 Recognition Performance

We measure the recognition performance by the *log-evidence* of the testing set after training. For generative models, this value tells us the logarithm of the probability that following the step-by-step procedures described within Algorithm 2 would have actually generated the observation sequences in the testing set. A greater log-evidence implies that the model is a better fit for the observed data. Table 4.2 provides these log-evidence values. In all cases, the modifications to LDA develop a model that better fits the data. This implies that LDA itself is a simplification of the generation process and omits information used for generating the observations. Although the

results provide evidence for the case that temporal relations are more informative than object relations, we believe this may not be concluded due to CAD-120’s method for annotating objects (SIFT features in two-dimensional space); a dataset with more precise recording of objects in the environment is necessary for confirmation. It is particularly worth noting that the parameterized composite model outperforms both parameterized LDA and the composite model in three of the train-test partitions and outperforms one of them in the other two partitions. Thus the *use of both temporal and object relations appears to often be more informative than either relation alone*. This synergistic evidence supports our hypothesis that these relations contain mutually exclusive information regarding the observed agent’s activity.

4.3.4.3 Topic and State Investigation

Because unsupervised learning algorithms identify their own patterns in data, it is important to study the learned clusters and ensure that the results are coherent. The learned topic/activity and state distributions might not resemble what a human would classify as a distinct category, but some distinctions should be evident within and between the distributions as discussed in Section 4.2.2. This also provides an opportunity to compare what kinds of information the different models capture. Table 4.3 lists the the optimally-trained models’ actual values of T and C for each train-test partition. We observe that the models extending LDA were assigned very similar numbers of topics/activities T regardless of the partition. This number is the edge case of our parameter sweep, which implies that increasing the range would have further improved performance. It could also indicate concern that the choice of γ yielded a vocabulary of signal tokens that *do not share the properties of a natural language* that allow topic models to work as intended—besides the explicit BOW and Markov assumptions, many *models have implicit assumptions based on unrealized properties of the domain*, including various autonomous planning benchmark defini-

Table 4.3. Number of Topics/Activities T and HMM States C in the Optimally-Trained Models per Partition of CAD-120.

	$P1$	$P2$	$P3$	$P4$	$P5$
LDA	$T = 61$	$T = 63$	$T = 45$	$T = 59$	$T = 55$
Parameterized LDA	$T = 89$	$T = 83$	$T = 89$	$T = 89$	$T = 89$
Composite	$T = 62$ $C = 4$	$T = 62$ $C = 6$	$T = 62$ $C = 2$	$T = 62$ $C = 5$	$T = 62$ $C = 6$
Parameterized Composite	$T = 92$ $C = 6$	$T = 92$ $C = 4$	$T = 89$ $C = 3$	$T = 92$ $C = 2$	$T = 86$ $C = 3$

tions [122]. However, the models' log-likelihoods of generating the training data were reaching the asymptotic limit and the log-evidence would most likely only experience marginal increase, if any.

We thus want to identify in which ways the different topic models classify the plan executions in CAD-120. Because they have different log-evidence values, it would seem to be the case that each one is recognizing something unique in comparison to the other models. Figure 4.6 displays a frame-by-frame breakdown of inferred \vec{z} (by activity/topic) and \vec{s} (by HMM state) for a plan execution in $P1$'s training set. The first thing to notice is that LDA infers a single topic for all postures in a single execution. This implies that LDA is only able to classify the overall activity, but we must consider that the length of an average recording in CAD-120 is less than twenty seconds (600 frames). Although this chapter proposes that the entire execution represents the plan execution so that θ is a distribution of actions throughout the entire recording, it is possible that these documents are *too short* for such analysis. In the case that there are too few signal tokens, θ should be unimodal so that activity recognition is nominally plan recognition. That is, the plan consists of a single action/activity to recognize. Despite this potential setback, we are unaware of other datasets that record object relations and continue to use CAD-120.

We also observe that the parameterized variations infer a single topic for the entire document. However, these topics are different from those in LDA due to the

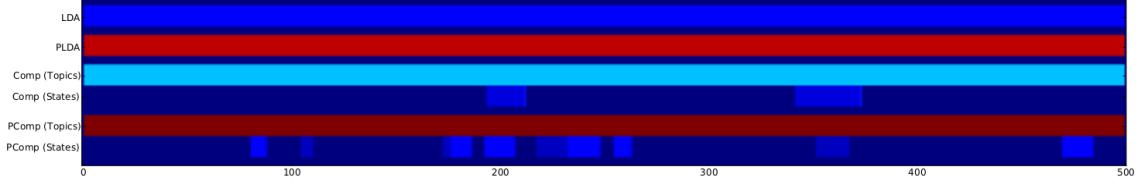


Figure 4.6. Visualization of each model’s inferred topic and HMM state assignments for an execution of ‘having a meal’ which breaks down into ‘moving’ (1-83; 101-134; 156-194; 241-276; 346-370; 385-444), ‘eating’ (84-100; 371-384), ‘reaching’ (135-155; 333-345), ‘drinking’ (195-240), and ‘placing’ (277-332).

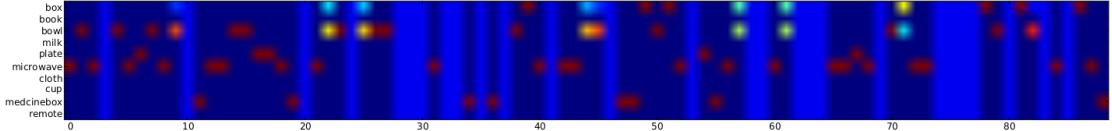


Figure 4.7. A heat map illustrating the distributions over objects Ω_t for each activity t in the optimally-trained parameterized LDA with respect to $P1$. Each column is single topic where red indicates the greatest probability mass and dark blue indicates the least probability mass.

inclusion of object relations. Table 4.3 shows that there are more topics in these models, allowing more precise cases during recognition. Evidence of this diversity exists within the learned object distributions $\vec{\Omega}$ for each topic. Figure 4.7 illustrates $\vec{\Omega}$ for the optimally-trained parameterized LDA on $P1$ as a heat map (the parameterized composite model’s heat map is very similar). We identify three distinct distribution types among $\vec{\Omega}$:

- The *unimodal distribution* contains one red dot per column. This likely indicates an activity whose signal tokens commonly interact with a single object.
- The *uniform distribution* is a solid blue column. This presents a lack of preference for objects, which seems to indicate that either no objects are involved or the signal tokens are distinct enough that the objects involved in such activities do not matter for recognition.

- The *bimodal distribution* has two dots each ranging in hue from light blue to orange. We only find this distribution type with objects ‘box’ and ‘bowl’ as the modes. These two objects are commonly found together in the ‘making cereal’ activities in CAD-120, which implies that the objects are expected as a pair for ‘making cereal’.

Although we hypothesized that objects with similar affordances would be clustered together in each activity’s Ω distribution, it is likely the case that ten objects are too few to identify these higher-level functional purposes. CAD-120 only provides annotated affordance labels based on the activity so that a single object will appear to change during the plan execution if we were to use their affordance labels in place of the objects themselves. *A more robust dataset with longer plan execution recordings and a greater variety of objects will be necessary for a full analysis of the impact of object relations in our models for unsupervised recognition.*

On the other hand, the (parameterized) composite model’s inferred states have a more distinguishable trend. The majority of the inferences for \vec{s} are dark blue, which represent state 1 where (parameterized) LDA is used for sampling the signal token. However, there are a few streaks where the color changes to indicate that a different state was inferred. Figure 4.6’s caption reveals that most the streaks appear as a transition between two annotated subactivities. This alludes to the syntactic properties that Griffiths et al. observed when they introduced the composite model for analyzing text documents. This appears to imply that despite the shorter length, *the underlying structure is recognizable even if the actual activities cannot be distinguished.* The learned transition functions $\vec{\xi}$ were almost always unimodal favoring state 1. If the transition from some state did not favor state 1, then it favored transitioning back to itself (which would explain the streak of a single color rather than a variety of colors). As both $\vec{\xi}$ and $\vec{\Omega}$ were similar between the parameterized composite model and the respective composite and parameterized LDA topic models,

there appears to be little overlap between the information gained from temporal and object relations.

4.4 Concluding Remarks: Recognizing Activities and Plans

A single moment in time can provide a surprising amount of insights into what an agent is doing, but *proper sensing and representations* are both necessary to make this effective for computational systems. Without the correct sensor, R_{ing} cannot be aware of the information’s presence unless there exists some other means to infer it. If the information is sensed or inferred, then R_{ing} still needs to encode that information in a way that is useful for recognition—such representation choices apply to designing both the recognition models and their inputs. This chapter has begun to merely scratch the surface when R_{ing} uses a single RGB-D sensor, yet humans have a wider variety of sensing abilities.

Although the field of activity recognition has generally moved to deep learning, the importance of these challenges still persist. We described throughout Section 3.2 how LSA and HMM models both have replacements, and it is important to acknowledge that a replacement of a function is simply its implementation. So creating useful inputs is still necessary to get practical outputs, and the implementation is still a constructed model. In this case, a story deriving the data with an interpretable graphical model representation is now a ginormous graphical model with connections between various nodes representing substructures that have empirical evidence of specific purposes. Whether constructing a story or connecting functional substructures, *knowledge of the how the sensed information relates to R_{ed} ’s decision-making process should play a role in the process.*

If the activity recognition algorithm is designed using some form of topic modeling, then we have provided evidence that the inferred topic $z_{d,i}$ per signal token $w_{d,i}$ serves as a form of activity recognition. However, the actual activity label is unknown and

might not be easy for a human to interpret because such models are unsupervised. We proposed several approaches to automatically generate interpretations of each activity/topic, but much still remains to be done in terms of their verification.

In contrast to the unlabeled actions within a sensed plan execution d , each execution is a discrete entity whose ID number serves as a label to a possibly unique plan π_d in the worst case. This means that we infer the activities \vec{z}_d as a by-product of training the topic model, which yields an approximation of distribution θ_d . After observing a new plan execution d' for some unknown plan $\pi_{d'}$, activity recognition infers $\vec{z}_{d'}$ and thus also approximates $\theta_{d'}$. This means we can compare $\theta_{d'}$ with each θ_1 through θ_D in the training dataset and determine which π_d share a similar distribution of actions with $\pi_{d'}$. This yields an approximate form of plan recognition because we are ultimately matching the observation sequence of signal tokens to an execution sequence. This approach must be done as post-processing in order to have a complete θ_d approximation, which is not ideal for the real-time constraints of interaction in the PRETCIL framework. Although we could create partially-complete distributions $\theta_{d,1..i}$ when performing activity recognition so far over $w_{d',1}$ through $w_{d',i}$, this does not guarantee accounting for partial ordering of actions or noise.

CHAPTER 5

INTEGRATION OF RECOGNITION AND AUTOMATED PLANNING

I am not holding the door for you because I am a gentleman. I am holding the door to help you because your hands are full with all those pizza boxes, and I can tell you plan to take them somewhere outside this room. So I can at least get the door so you don't have to struggle with it.

— Richard (Rick) G. Freedman,
in a conversation with a colleague after hosting an event

In order for an agent, whether they are a robot, computer, or human, to interact with others around them, they cannot be aware of just the presence of others. The interactive agent must also be able to *understand what they are doing*. In particular, interaction involving multiple agents requires some form of coordination during plan execution. Communication or executing a precise, predetermined plan can be achieve coordination, but plenty of interactions begin spontaneously (such as aiding someone who seems to be struggling with a task) with nothing more than observations available to direct the coordination efforts. Maeda et al. [185] collected data from human teams performing tasks that enabled robots to mimic the interaction after learning probabilistic motion primitives from the data. This form of interaction is *reactive*, serving as a low-level reflex to the partners' motions. We instead focus on *responsive* interaction, considering the partners' activities at a higher level and deciding how to act alongside them.

Despite the past separation between automated planning and PAIR research, the *integration of these areas is crucial for responding well to others*. Recognizing the plans and goals of those with whom an agent interacts provides a context for the task

and an expectation of how the other agents are approaching it. Planning within a similar context and accounting for the other agents' potential actions as constraints allow the observing agent to identify a sequence of actions that work well with everyone else. This is not a one-directional relationship because the *recognized/inferred actions may not be exactly correct*, or the other *agents might vary their behaviors and invalidate the observed agent's responding plan*. Thus interactive agents must perform recognition continuously in order to confirm the validity of their current action choices and update the information for potential replanning. To account for these circumstances and the fact that people often vary their approach, and .

This chapter investigates the impacts that recognition and automated planning can have on each other, introducing the study of *responsive planning*. There are multiple components to responsive planning once recognition is complete, and each one presents new challenges from other areas of research. *Goal reasoning* [280] normally assigns goals to agents based on the circumstance, but the in-the-moment nature of interaction can cause the circumstance to evolve more rapidly. Using current information from recognition algorithms, we explore the process of developing intermediate goals that can guide the agent for the near-future of the interaction. However, because the family of recognition algorithms we use primarily assume that the observed agent completed its task, we further consider how to make the algorithms' outputs more prudent. It is also ideal to measure the quality of the generated response, a plan solving this intermediate goal. We introduce the notion of *helpfulness* for this purpose, and we provide a preliminary investigation exploring its potential uses and definition challenges.

Our approach to responsive interaction relies on the probabilistic predictions from Ramírez and Geffner's recognition as planning (RaP, see Section 3.1.3) algorithm. However, the RaP family of recognition algorithms were not designed with interaction

tasks in mind, and this leads to a few necessary modifications. After explaining the primary changes, we discuss how RaP provides sufficient information for responsive interaction.

5.1 Foresight Using Survival Analysis

Despite the recognized distributions over \mathcal{H} in the probabilistic RaP experiments [227] appearing to be very effective, the performance was best when a reasonable percentage of the actions in the agent’s (possibly optimal) plan execution are observed. This is because a greater number of observations increases the probability of including one of the later actions taken in the sequence. Although the earlier and intermediate actions of a plan π play a role in the recognition algorithm, the later ones impose constraints near the goal-satisfying states that require an agent to go out of their way to follow/avoid the observed action(s). Such distinguishing actions motivate goal recognition design’s WCD [156] mentioned in Section 3.3.3, revealing the observed agent R_{ed} ’s intentions as early as possible.

The influence of the most-recent observation is visually evident in Ramírez and Geffner’s random walk example, shown in Figures 1 and 2 of their work—the values of $P(G \in \mathcal{H})$ for each step of a noisy random walk produced a plot whose goal location(s) with the greatest probability was(were) closest to the current location of the agent at that time step. The approach is thus greedy with respect to assuming that R_{ed} acts optimally because it motivates finishing the task as soon as possible. This creates a *bias towards recognizing locally short-term goals* and ignoring tasks that require a plan of greater cost. The bias from this RaP-defining assumption is ideal for post-processing when it is known that R_{ed} already completed their plan execution and satisfied the goal.

However, for recognition while the *plan’s execution is in-progress*, applying foresight to recognize long-term goals earlier is necessary in order to properly predict the

tasks that require more effort. Otherwise, it might be too late to determine a proper response and/or interact. We *counter-balance RaP’s greedy likelihood computation from Section 3.1.3 with a prior that is biased towards the more costly tasks* that are typically ignored. This is most important at the beginning of the interaction when one cannot observe enough actions to solve any of the more costly tasks. However, it is not beneficial to continue favoring plans corresponding to long-term goals after there are enough observations to confidently identify R_{ed} ’s task. That would reverse the direction of the bias towards future tasks when the current ones are the most likely.

We introduce a *dynamic prior* that favors long-term goals with greater-cost optimal plans when the plan execution time/resource consumed r is lesser, and it converges to the true prior $P(G)$ as r increases so that the likelihood’s optimality assumption later takes precedence. To do this, we revise the probability formulation to include r for a joint distribution over the observed agent’s task $G \in \mathcal{H}$ and resource consumption $r \in \mathbb{N} \cup \{0\}$:

$$P(G|\mathcal{O}, r) = Z^{-1} P(\mathcal{O}|G, r) P(G|r)$$

where $P(\mathcal{O}|G, r) = P(\mathcal{O}_{\leq r} | G)$ is the likelihood for the observation prefix with cumulative cost $\leq r$ and $P(G|r) = Z''^{-1} P(r|G) P(G)$ is the prior over the joint distribution. We define the newly introduced probability $P(r|G)$ using survival analysis.

Definition 46. *A plan π that successfully solves a planning problem with goal G is in-progress at step i if G ’s conditions are not yet satisfied after performing $\pi_{\leq i}$.*

Definition 47. *A plan π that successfully solves a planning problem with goal G is maintaining if G is already satisfied while performing each $\pi_{\geq j}$ for $j < |\pi|$.*

We only consider plans without any maintaining actions. Then a *plan is surviving as long as it is in-progress*, and it ceases execution once the goal conditions are

satisfied. Clearly any plan for some goal $G \in \mathcal{H}$ must survive at least $cost_{\pi_G^*}$ (such as $|\pi_G^*|$ actions) because the optimal solution is the least cost needed to complete the plan. Depending on R_{ed} 's optimality, we might observe some additional amount of resources k spent during the plan's execution. As this is counting a finite quantity of events (additional amount of discrete resources/executed actions) within a specific time window (one plan execution), we assume that the probability of a plan π 's cost to solve G is

$$P(cost_\pi = r | G) = \begin{cases} 0 & \text{if } r < cost_{\pi_G^*} \\ \frac{(\lambda+1)^{r-cost_{\pi_G^*}+1}}{(r-cost_{\pi_G^*}+1)! \cdot (e^{\lambda+1}-1)} & \text{otherwise} \end{cases}$$

based on the *Poisson distribution*. Situations involving counts of events over a fixed time window use the Poisson distribution

$$\text{Poisson}(k \in \mathbb{N} \cup \{0\}; \lambda \in \mathbb{N}) = \frac{\lambda^k}{k! e^\lambda}$$

where λ is the expected count and standard deviation. So for expected plan execution cost $cost_\pi = (cost_{\pi_G^*} + \lambda)$, an optimal agent should have parameter $\lambda = 0$ while a less optimal agent should have a greater λ value. However, the Poisson distribution cannot allow $\lambda = 0$ because the numerator would always be 0. We instead use the positive Poisson distribution

$$\text{Poisson}^+(k \in \mathbb{N}; \lambda \in \mathbb{N}) = \frac{\lambda^k}{k! (e^\lambda - 1)}$$

[248] in the ‘otherwise’ case above and increment both $k = (r - cost_{\pi_G^*})$ and λ by 1 to address the removed 0. Besides Bayesian updating to find a more accurate prior $P(G)$, we can update λ through multiple trials or interactions as R_{ed} is observed and better understood.

Survival analysis is an area of statistics that determines the probability that something continues to live with respect to a life-expectancy distribution. Thus, given the piecewise equation above as the life-expectancy of a plan solving goal G , the probability of the plan's survival is relative to how likely it is still in-progress for the current cost r :

$$P(r|G) = P(cost_{\pi} \geq r | G) = 1 - \text{cdf}_r(cost_{\pi} | G)$$

where cdf_r is the cumulative distribution function from 0 to r . The hazard function $h(r) = P(cost_{\pi} = r | G) / P(cost_{\pi} \geq r | G)$ also tells us the likelihood that the plan will terminate execution after total cost r , which might be useful for reasoning about whether the observed agent is almost finished with its task.

Instead of using the hazard function, we only use the survival function as *foresight to advance the resource consumption*. When computing the likelihoods for RaP, we can find the plan with the fewest consumed resources satisfying the current observations to identify a current value for r . Because the likelihood prefers goals that will not consume too many more resources than r , we then compute the prior at future time $r + \epsilon$ for some $\epsilon \geq 0$. If there is enough variation between $cost_{\pi_G^*}$ for each $G \in \mathcal{H}$, then this will bypass the goals that the likelihood prefers and redistribute probability mass to the long-term goals that cannot yet be achieved. Specific assignments for ϵ will vary by many factors such as the variation of the goals' optimal plan costs, time already elapsed (if r becomes too large, then all the goals will be surpassed for a more uniform prior), and preferred amount of look-ahead for interactive purposes. We leave the investigation of how these factors impact choices for ϵ to future work.

5.2 Finding Intermediate Interaction Tasks with Necessities

Up to this point in the dissertation, we have only discussed recognition algorithms to interpret R_{ed} 's actions. As a partner in the interactive experience, we introduce the observing agent R_{ing} . R_{ing} plans their own responses with respect to their predictions

of the observed agent R_{ed} . We use the output from probabilistic RaP, which is a distribution over \mathcal{H} , to extract the goal conditions that R_{ed} most likely intends to solve:

Definition 48. *For a set of hypothesized goals \mathcal{H} and a probability distribution over \mathcal{H} , the necessities are an intermediate goal composed of overlapping features or conditions of the most likely elements of \mathcal{H} according to the distribution.*

Definition 49. *Let \mathcal{H} be a set of STRIPS-represented goals, each a conjunction of fluents from some set F . The necessity of a propositional fluent $f \in F$ with respect to \mathcal{H} and a sequence of observed actions \mathcal{O} is the expected probability that f is a goal condition given \mathcal{O} . That is,*

$$N(f \in F | \mathcal{O}) = \sum_{G \in \mathcal{H}} P(G | \mathcal{O}) \cdot \mathbf{1}(f \in G)$$

where $\mathbf{1}$ is the indicator function.

A necessity of 1 implies that all hypotheses with non-0 probability require f as a goal condition, and a necessity of 0 implies that no hypotheses with non-0 probability require f as a condition. Then for some threshold τ , we define

$$\widehat{G}_{R_{ed}} = \{f | N(f | \mathcal{O}) \geq \tau\}$$

as the estimated goal conditions that R_{ed} is trying to satisfy, which represents the necessities. Using this estimation in addition to their own set of actions $A_{R_{ing}}$ and current state I_{now} , we define the responsive interaction problem as a *centralized multi-agent planning problem* of the form

$$\mathcal{P}_{R_{ing}} = \langle F' \cup F_{R_{ed}+R_{ing}}, I', A_{R_{ed}+R_{ing}}, G_{R_{ing}} \rangle.$$

If $\mathcal{P}_{R_{ing}}$ has no solution due to conflicting necessities, such as $f, \neg f \in G_{R_{ing}}$ for some $f \in F$, then R_{ing} does not have enough knowledge about the necessities and continues to observe without interaction¹. We specifically propose three forms of responsive interaction based on the types introduced in Section 1.2.3. We define them using a STRIPS-style representation to continue RaP’s tradition of using off-the-shelf classical planners, but Chapter 6 explains through our implementation that one can use alternative representation languages.

Definition 50. Assistive Interaction means that R_{ing} ’s goal is to only help R_{ed} accomplish their goal. This planning problem $\mathcal{P}_{R_{ing}}^{\text{Assistive}}$ is of the form $F' = F$, $I' = I_{\text{now}}$, $A_{R_{ed}+R_{ing}} = (A_{R_{ed}} \cup \{\text{no-op}\} \times A_{R_{ing}} \cup \{\text{no-op}\}) \cup A_{\text{Joint}}$, and $G_{R_{ing}} = \widehat{G_{R_{ed}}}$.

The remaining forms of interaction indirectly use $\widehat{G_{R_{ed}}}$ through a new fluent that denotes whether R_{ed} accomplished these conditions. We call this fluent *success* and add it through the additional add effect

$$\text{success} \vee \bigwedge_{g \in \widehat{G_{R_{ed}}}} g \rightarrow \text{success}$$

for each action in $A_{R_{ed}}$, implying that solving the goal conditions once is sufficient to complete the task. We respectively call these modified sets F^S and A^S .

Definition 51. Independent Interaction means R_{ing} has a personal goal G' to accomplish, but should avoid preventing R_{ed} from accomplishing its own task at the same time. This planning problem $\mathcal{P}_{R_{ing}}^{\text{Independent}}$ is of the form $F' = F^S$, $I' = I_{\text{now}}$, $A_{R_{ed}+R_{ing}} = ((A_{R_{ed}} \cup \{\text{no-op}\} \times A_{R_{ing}} \cup \{\text{no-op}\}) \cup A_{\text{Joint}})^S$, and $G_{R_{ing}} = G' \cup \{\text{success}\}$.

¹In Section 7.2.2, we present an issue with this concept in practice. There are cases where doing nothing can be detrimental to the interactive experience, and some alternative, domain-specific default behavior is the better response.

Definition 52. Adversarial Interaction means R_{ing} 's goal is to prevent R_{ed} from achieving their goal for some duration d (rather than STRIPS, a representation language such as linear temporal logic [223] must be used when $d = \infty$). This planning problem $\mathcal{P}_{R_{ing}}^{Adversarial}$ is of the form $F' = F^S \cup \{0, 1, \dots, d\}$, $I' = I_{now} \cup \{0\}$, $A_{R_{ed}+R_{ing}} = (((A_{R_{ed}} \cup \{\text{no-op}\}) \times A_{R_{ing}} \cup \{\text{no-op}\}) \cup A_{Joint})^S\right)^{step}$, and $G_{R_{ing}} = \{\neg\text{success}, d\}$ where $\{\cdot\}^{step}$ applies an incremental add effect $i \rightarrow (i + 1)$.

For each form of interaction, R_{ing} derives the joint optimal plan

$$\pi_{G_{R_{ing}}}^* = \bigcirc_{i=1}^y (a_{R_{ed},i}, a_{R_{ing},i})$$

that R_{ing} and R_{ed} should perform alongside each other. They can find $\pi_{G_{R_{ing}}}^*$ using the same off-the-shelf classical planner that they used to perform RaP. However, this plan is optimistic because R_{ed} acts independently and *is not guaranteed to follow the joint plan* unless there is direct communication where R_{ing} tells R_{ed} what actions to perform. If there was communication, then it would have been possible (though not required) for R_{ed} to reveal G to R_{ing} in the first place; so we consider the case where direct communication between the agents is absent. Then R_{ing} can only perform its assigned actions from $\pi_{G_{R_{ing}}}^*$, which we call

$$\pi_{R_{ing}} = \bigcirc_{i=1}^y a_{R_{ing},i}.$$

This introduces the need for replanning with new observations throughout the interaction. There should be more observations available after performing $\pi_{R_{ing}}$ for more accurate recognition, meaning that $\widehat{G_{R_{ed}}}$ should also become more specific. We defer finding the ideal moment(s) to replan to future work, but Section 6.2.3 considers replanning based on monitoring R_{ed} 's plan execution with respect to each $a_{R_{ed},i}$.

5.3 Helpfulness as a Measure of Impact on the Interaction

As R_{ing} performs the actions in their plan $\pi_{R_{ing}}$, new changes to the state will also occur that might affect R_{ed} 's performance of π_G^* . Assistive interaction intends for these changes to facilitate R_{ed} 's ability to complete task G , changes from independent interaction should not greatly affect R_{ed} 's plan unless there is a resource conflict to resolve, and changes from adversarial interaction should inhibit R_{ed} from completing G . For all these categories, we can run another execution of the off-the-shelf classical planner to measure $\pi_{R_{ing}}$'s *helpfulness*. Instead of using a transformation similar to the one in RaP algorithms, we simulate $\pi_{R_{ing}}$ within R_{ed} 's centralized multi-agent planning problem such that R_{ed} plans according to R_{ing} 's response. We define

$$\mathcal{P}_{R_{ed} \leftarrow R_{ing}} = ((F_H, (A_{R_{ed}} \cup \{\text{no-op}\} \times A_{R_{ing}, H} \cup \{\text{no-op}\}_{H+}) \cup A_{\text{Joint}, H}), I_H, G)$$

where

$F_H = F \cup \{p_i \mid 0 \leq i \leq y + 1 = |\pi_{R_{ing}}| + 1\}$, $I_H = I_{now} \cup \{p_0\}$, and $\{\cdot\}_H$ and $\{\cdot\}_{H+}$ are the following modified sets of actions in \cdot :

- $add_{a,H} = add_{a,H+} = add_a \cup \{p_{i-1} \rightarrow p_i \mid a = \pi_{R_{ing},i}\}$
- $del_{a,H} = del_{a,H+} = del_a$
- $pre_{a,H} = pre_a \cup \left\{ p_{y+1} \vee \bigvee_{i \in \{j \mid a = \pi_{R_{ing},j}\}} (p_{i-1} \wedge \neg p_i) \right\}$
- $pre_{a,H+}(a) = pre_a \cup \left\{ p_y \vee \bigvee_{i \in \{j \mid a = \pi_{R_{ing},j}\}} (p_{i-1} \wedge \neg p_i) \right\}$

Thus the fluents p_i now signify the performance of each action in $\pi_{R_{ing}}$, and the impossible precondition p_{y+1} of $\{\cdot\}_H$ forces R_{ing} to execute no-ops once their plan is completely executed—these no-ops could become other actions if replanning is

performed later. Similar to the derivation of $\pi_{R_{ing}}$, the new actions that R_{ed} will perform as adaptation to R_{ing} 's responsive actions are

$$\pi_{R_{ed} \leftarrow R_{ing}} = \bigcirc_{i=1}^z a_{R_{ed}, i}$$

where the optimal solution to $\mathcal{P}_{R_{ed} \leftarrow R_{ing}}$ is

$$\pi_{G+\pi_{R_{ing}}}^* = \bigcirc_{i=1}^z (a_{R_{ed}, i}, a_{R_{ing}, i}).$$

Definition 53. *The helpfulness of a responsive plan $\pi_{R_{ing}}$ is the change in cost from agent R_{ed} acting on their own to both agents working simultaneously*

$$H(\pi_{R_{ing}}) = \text{cost}_{\pi_{G, \geq now}^*} - \text{cost}_{\pi_{R_{ed} \leftarrow R_{ing}}}.$$

We assume $\text{cost}_\pi = \infty$ if π does not exist.

Lemma 1. *If R_{ing} knows G , R_{ing} is being assistive or independent, and there exists a non-invasive sequence of actions such that R_{ing} never affects a precondition of any action in π_G^* , then $H(\pi_{R_{ing}}) \geq 0$.*

Proof. Because R_{ing} knows the correct goal, $G_{R_{ing}} = G$. This implies that the problems $\mathcal{P}_{R_{ing}}$ and $\mathcal{P}_{R_{ed} \leftarrow R_{ing}}$ are identical (the additional fluents and action modifications only ensure following the solution). Thus $\pi_{G_{R_{ing}}}^* = \pi_{G+\pi_{R_{ing}}}^*$, and R_{ing} can at least perform the non-invasive sequence of actions as $\pi_{R_{ing}}$ while R_{ed} performs its initial plan such that $\text{cost}_{\pi_{G+\pi_{R_{ing}}}^*} \leq \text{cost}_{\pi_{G, \geq now}^*}$. Hence $H(\pi_{R_{ing}}) \geq 0$. \square

Lemma 1 shows that, with a good prediction from recognition, the observing agent will rarely hinder the observed agent's progress unless the domain and current state force R_{ing} to get in the way. Clearly this should not hold for adversarial interaction

because such an agent wants to provide as little help as possible. However, it is more difficult to guarantee not being helpful due to the fact that there is usually more than one (optimal) plan. That is, R_{ed} can perform an alternative sequence of actions that $\pi_{R_{ing}}$ does not ‘block’ and still complete the task—this phenomenon is strongly related to Section 1.2.4’s explanation about why many two-player games are PSPACE-complete.

Lemma 2. *If R_{ing} knows G , R_{ing} is being adversarial, and there exists an invasive sequence of actions such that R_{ing} always affects a precondition of some action in every possible π_G^* , then $H(\pi_{R_{ing}}) \leq 0$.*

Theorem 1. $-\infty < H(\pi_{R_{ing}}) \leq c(\pi_G^*) - c(\pi_{R_{ed}+R_{ing}}^*)$ where $\pi_{R_{ed}+R_{ing}}^*$ is the optimal plan that solves the centralized multi-agent planning problem $((F, A_{R_{ed}+R_{ing}}), I, G)$ and $A_{R_{ed}+R_{ing}} = A_{R_{ed}} \cup \{\text{no-op}\} \times A_{R_{ing}} \cup \{\text{no-op}\}$.

Proof. The lower bound follows from Lemma 2 because R_{ing} can perform some action that permanently prevents R_{ed} from satisfying the preconditions of all actions whose effects satisfy one of G ’s conditions. The upper bound extends from the proof of Lemma 1; the best case is that both agents work together from the beginning (even if that means R_{ing} does nothing). As each time step progresses, the cost for both the single-agent plans and the multi-agent plan will decrease uniformly so that $cost_{\pi_G^*} - cost_{\pi_{R_{ed}+R_{ing}}^*} = cost_{\pi_{G,\geq now}^*} - cost_{\pi_{R_{ed}+R_{ing},\geq now}^*}$.

□

5.3.1 Challenges with Computing Helpfulness

This section elaborates on an ongoing discussion between Richard G. Freedman, the author of this dissertation, and Steven J. Levine, the creator of PIKE [174, 176] and RIKER [175]. When performing some comparison experiments between RIKER and the PRETCIL framework (implemented using RaP and responsive planning,

see Chapter 6) for Levine’s dissertation [175]², they chose helpfulness as a means of comparing the results of the interaction frameworks. However, discussions throughout the experiments led to new ideas and challenges related to helpfulness as a measure of interaction quality.

5.3.1.1 Helpfulness without Bias from Problem Instances

One of the earliest concerns arose from fairness in comparisons: *how do we avoid the bias of more opportunities to improve helpfulness for plans with greater cost?* For example, suppose one interactive agent R_{ing}^1 ’s responses always cut the cost of π_G^* in half while the other interactive agent R_{ing}^2 ’s responses always reduce the cost of the plan by some constant amount k . We cannot best express this with respect to the approaches’ helpfulness because $cost_{\pi_G^*}$ ’s variance between trials (and even in the benchmark design) plays a large role in the collection of computed $H(\pi_{R_{ing}})$. In this case, simpler problems where $cost_{\pi_G^*} \leq k$ yield $H(\pi_{R_{ing}^2}) = cost_{\pi_G^*}$ and $H(\pi_{R_{ing}^1}) = 0.5cost_{\pi_G^*}$. On the other hand, more complex problems where $cost_{\pi_G^*} \geq 2k$ present $H(\pi_{R_{ing}^2}) = k$ and $H(\pi_{R_{ing}^1}) = 0.5cost_{\pi_G^*} \geq k$. In the former, R_{ing}^2 ’s approach appears to be more helpful; yet R_{ing}^1 ’s approach is more helpful if k is reasonably small.

Besides a fair comparison of approaches, the fact that different goals have varying optimal solution costs might provide contrasting insights for a single approach. Like the short-sighted side-effect of RaP algorithms (see Section 5.1), there are far fewer opportunities to be helpful when responding to a plan of cost c compared to responding to a plan of cost $10c$. Helpfulness returns the difference in costs, which is reasonable for a single-case, post-processing comparison in which R_{ed} will perform

²We refer the reader to this dissertation for the experimental results. They are not reprinted in this dissertation to (1) avoid plagiarism and (2) not present ‘outdated’ results because the PRETCIL framework implementation was a prototype at the time of the experiments. Unfortunately, time constraints prevented Freedman and Levine from rerunning the experiments.

a task with or without R_{ing} 's assistance. However, *the difference does not reflect the potential to be helpful*, which matters in situations where R_{ed} can decide whether to perform the task and they are aware of R_{ing} 's intentions to interact—in this case, helpfulness can serve like value of information. As a grounded example, consider the two tasks where

- R_{ed} has to move a large piece of furniture across a room [200] and
- R_{ed} needs to move all the furniture in a room onto a moving truck.

The first task is inherently lower-cost so that R_{ed} can consider moving the piece of furniture alone, and any help R_{ing} provides is a nice bonus. The second task is much more costly to perform and R_{ed} will likely want to ensure that R_{ing} will provide sufficient help *relative to the task's effort* before starting to execute a plan. This can extend to waiting for R_{ing} to be ready to interact in a more helpful manner [299] or finding another interactive partner who is available to provide sufficient help [235].

This issue of varying input sizes is not unique to computing helpfulness. The term frequency-inverse document frequency representation is one such solution for documents of different lengths; the normalized vectors allow the cosine similarity metric to provide semantic comparisons between the documents' contents [249]. In a similar way, Levine proposed normalizing helpfulness to account for how much R_{ing} could help in the first place.

Definition 54. *The normalized helpfulness of a responsive plan $\pi_{R_{ing}}$ is the ratio comparing the helpfulness $H(\pi_{R_{ing}})$ to the maximum helpfulness where R_{ed} and R_{ing} worked together simultaneously from the start*

$$H_0^1(\pi_{R_{ing}}) = \frac{cost_{\pi_{G,\geq now}^*} - cost_{\pi_{R_{ed} \leftarrow R_{ing}}}}{cost_{\pi_{G,\geq now}^*} - cost_{\pi_{Joint,\geq now}^*}}.$$

The plan π_{Joint}^ assumes that both R_{ed} and R_{ing} know the shared goal G .*

Normalized helpfulness clearly applies most to assistive interactions, and Levine created both PIKE and RIKER exclusively for such interactions. The more assistive an agent is, the more $H_0^1(\pi_{R_{ing}})$ approaches 1 *regardless of the lowest-cost solution to the problem*. Normalized helpfulness can also apply to independent interactions, where R_{ing} 's lack of aid and interference is expected to yield values of $H_0^1(\pi_{R_{ing}})$ closer to 0. Adversarial interactions will be more difficult to assess because a joint plan implies that both agents work together to accomplish the goal; so a different definition of normalized helpfulness is necessary. However, Theorem 1 indicates that normalized helpfulness can be feckless for studying adversarial interactions if there exists a plan $\pi_{R_{ing}}$ that completely hinders R_{ed} , which would set the denominator of the ratio to ∞ .

5.3.1.2 The Roles of Cost in Computing Helpfulness

The next challenge Levine and Freedman encountered was the actual definition of ‘cost’ for plans involving multiple agents, including $\pi_{R_{ed} \leftarrow R_{ing}}$ and π_{Joint}^* . Traditionally, a plan’s cost is the sum of the costs of all the actions, no matter which agent performs them. This is because *automated planning is primarily concerned with problem solving more than agent contribution*, which means all optimal plans are equally ideal even if each agent’s efforts are disproportional. Freedman proposed exclusively counting the costs of R_{ed} ’s actions in such plans, which already coincides with the extraction of actions from $\pi_{G+\pi_{R_{ing}}}^*$ to create $\pi_{R_{ed} \leftarrow R_{ing}}$.

This enables R_{ing} to be more helpful if they can perform more actions in the centralized multi-agent problems, but it also introduces new issues to keep in mind. If R_{ed} and R_{ing} are both capable of solving the problem on their own, then normalized helpfulness is only maximized when R_{ing} does all the work. If they are ever assigned a *utility function to maximize R_{ing} ’s helpfulness* (see Section 5.3.2), then this effectively

encourages R_{ing} to act like a servant or slave to R_{ed} and also encourages R_{ed} to be as lazy as possible and take advantage of R_{ing} .

As we defined actions on a simultaneous-move basis (indicated by the action tuples in the multi-agent plans), we addressed this issue by *assigning the no-op action a non-0 cost*. Specifically, we assumed that all actions, including no-op, have unit cost like traditional STRIPS planning. Thus R_{ed} 's portion of the plan has some cost even if they do nothing, and the multi-agent planner then finds optimal solutions where R_{ed} and R_{ing} split the workload in tandem to reduce the number of actions and no-ops. However, it is reasonable to question whether no-op should have a cost, and this comes down to what ‘cost’ specifically measures. An agent is *unlikely to expend resources such as energy* when doing nothing, but doing nothing for the duration that the other agent acts is *spending resources such as time*. Freedman and Levine’s experiments counted the number of turns taken to solve the problems, which is an instance of time. However, what happens when time is not as important as energy? Should these cases force R_{ing} to do everything when helpfulness matters?

5.3.1.3 Helpfulness at the Local and Global Perspectives

Though less of an issue, Freedman and Levine began to consider the moments at which (normalized) helpfulness changed in their approaches. Although RIKER and the PRETCIL framework are both interactive, they do not plan their interactions with respect to the same events. RIKER precomputes a library of plans and compiles them into a single probabilistic temporal plan network with choice nodes that branch according to differences between plans in the library. On the other hand, the PRETCIL framework uses the responsive planning techniques described in this chapter, which are iterative and replan given the new observations. This means that the overall policy is constant for RIKER throughout the *global interactive experience*, but it is dynamic for the PRETCIL framework. This distinction is the reason for Levine’s

observation that RIKER runs faster than the PRETCIL framework in exchange for less robustness to R_{ed} executing unexpected actions [175]. Despite this difference, both approaches have dynamic intermediate plans $\pi_{R_{ing}}$ within the *local interactive experience* because R_{ed} 's ongoing actions affect how R_{ing} responds in its future plan.

Ultimately, responsive planning continuously updates a single plan $\pi_{R_{ing}}$ and consequently overwrites actions that R_{ing} did not yet execute. For each iteration, we can compute the (normalized) helpfulness and see how it *evolves over the course of the interaction*. In contrast to this, RIKER does not commit to later actions until it can deduce which action R_{ing} must take at an upcoming choice node. This incremental extension of $\pi_{R_{ing}}$ makes it difficult to examine the evolution of the helpfulness value. If we assume that an agent employing RIKER for interactive decision making performs only no-op actions from the next unresolved choice node, then we can explore how (normalized) helpfulness *refines over the course of the interaction*. Upon completing the interaction, it is possible to *evaluate the response's (normalized) helpfulness in its entirety* by defining $\pi_{R_{ed} \leftarrow R_{ing}}$ as the actions R_{ed} performed throughout the interaction and $\pi_{R_{ing}}$ as the actions R_{ing} actually performed throughout the interaction. The results in Levine's dissertation only include the analysis of normalized helpfulness in its entirety.

5.3.2 Helpfulness as a Heuristic

A future direction of research worth considering is how to apply measuring helpfulness *actively during interaction* rather than passively as a post-hoc analysis. Lemma 1 indicates that increasing helpfulness is ideal for assistive interaction, and Lemma 2 similarly implies that decreasing helpfulness is ideal for adversarial interaction. This motivates our hypothesis that helpfulness can *guide R_{ing} 's decision making process* throughout their interaction.

Currently, the type of interaction exclusively plays a role in R_{ing} 's intermediate goal generation, and the search process then finds any path to this goal from the initial state I_{now} . Although this sequence of actions satisfies the intermediate goal upon completion, there is no guarantee that *the interaction type is acknowledged throughout the responsive plan's execution*. Research in legible planning [66, 168, 196] shows that optimal-cost plans to the goal are often less interpretable for communicating one's intents, and additional criteria need to influence the planner to account for a higher-cost, but more legible plan. In most cases, these criteria adjust the heuristic or constraints to reshape the search progression because the *path to the goal matters as much as the goal itself*.

In place of legibility, it is ideal for the responsive plan $\pi_{R_{ing}}$ to be assistive, independent, or adversarial throughout. A search state should not be considered ‘near a goal state’ simply because its cost is lower, but also because it is a state that exhibits some degree of helpfulness with respect to the interaction type. In assistive interactions, helpfulness can serve as a tie-breaking strategy between states on the frontier with the same lowest expected cost to the goal [12]—the state whose current plan (path from I_{now}) has greatest helpfulness should be considered first. For independent interactions, helpfulness might not matter as much as long as R_{ing} does not interfere with R_{ed} 's expected goals. However, to ensure that R_{ing} 's plan is minimally invasive, helpfulness can serve as a tie-breaking strategy by selecting the state whose current plan's helpfulness has the least absolute value. As the final extreme, adversarial interactions should prioritize tie-breaking towards states whose current plan has the least helpfulness. To ensure that helpfulness is not ignored when tie-breaking is unnecessary during search, we can also consider the measurement as an additional evaluation of states and adjust the heuristic to account for both expected cost and helpfulness as either a linear combination [111] or a lexicographic preference for multiple objectives [289].

As an analogy to the correspondence between preferred helpfulness values and the type of interaction, consider the utility/evaluation function in the minimax algorithm [237]. For traditional adversarial search, the function involves a difference of player scores such as $score_{max} - score_{min}$. The minimizing agent *min* prefers to take actions that yield states with smaller values (ideally negative) and the maximizing agent *max* prefers to take actions that yield states with greater values (ideally positive). Due to the function's range including positive and negative values, both agents are *maximizing the absolute value of the function when they have the greater score* and *minimizing the absolute value of the function when they have the lesser score*. This approach favors increasing the difference between their scores while in the lead, which increases the chances of maintaining the lead. However, the approach also favors decreasing the difference between their scores while behind, which increases the chances of taking the lead from the other agent in the adversarial scenario. If both players are maximizing agents with an evaluation function that is a single score $score_{max}$, then they both want to increase the score and take actions with this mutual benefit; this is assistive behavior. Lastly, the traditional game tree for turn-based game-theoretic situations [63] without minimax assigns each player personal payoffs [$score_{max}$ $score_{min}$] to maximize with respect to a mutual understanding that other agents also intend to maximize their own payoff. The subgame Nash Equilibria solutions for such games are analogous to independent behavior.

In a turn-taking-style interaction such as the above game-tree-represented interactions, R_{ing} should apply the search modifications on their turn and only use the traditional cost-based heuristic h on R_{ed} 's turn unless their interaction type is known. Unfortunately, evaluating states in simultaneous-move-style interactions might be a bit more complicated due to contrasting heuristic values between R_{ed} and R_{ing} .

However, the greatest foreseeable challenge in this research direction has to do with *approximating helpfulness* given the plan up to a search state and the *necessities*

rather than R_{ed} 's real goal. As a post-interaction computation, helpfulness is easy to compute because the goal is known when searching for R_{ed} 's single-agent plan *and* we know what actions R_{ed} took to accomplish their goal. Without knowing the true goal or R_{ed} 's future actions, we must approximate both of these in the search. This is prone to compounding error if the necessities are incorrect, and the computational efforts involved to predict R_{ed} 's remaining actions could be high unless we make trivializing assumptions such as R_{ing} no longer acting (like measuring RIKER's helpfulness incrementally in Section 5.3.1.3). This assumption would reduce predicting the cost of R_{ed} 's remaining actions to the traditional cost heuristic from the current search state s_{cur} . Then $cost_{\pi_{G,\geq now}^*}$ would be the traditional cost heuristic from I_{now} to some goal state satisfying the necessities $\widehat{G^{R_{ed}}}$ and $cost_{\pi_{R_{ed}\leftarrow R_{ing}}}$ would be the sum of the predicted cost for R_{ed} 's remaining actions and R_{ed} 's actions' costs in the plan up to s_{cur} , similar to the breakdown of $g(s) + h(s)$ in A* search (Section 2.1):

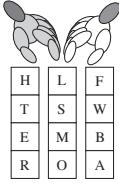
$$H\left(\pi_{I_{now}\rightarrow s_{cur}\rightarrow \widehat{G^{R_{ed}}}}\right) \approx h(I_{now}) - \left(cost_{\pi_{I_{now}\rightarrow s_{cur}}}^{R_{ed}} + h(s_{cur})\right).$$

Will these simplifications and possible errors still yield appropriate responsive behaviors for the interactions?

5.4 An Illustrated Example of Responsive Planning

To demonstrate the process of responsive interaction, we create a two-agent problem based on the BlockWords domain from Ramírez and Geffner's dataset³ [227]. Similar to the traditional Blocksworld toy problem, each block contains a letter so

³Note that the legacy code borrowed for our implementation does not scale to cases larger than the problems used in the International Planning Competition for which it was designed. The competition's problems turned out to be too limited to illustrate our work due to the types of problems available (grid world-like problems do not introduce many interaction opportunities) and their typical solution lengths (R_{ing} would not have sufficient information to interact until R_{ed} completes the short tasks).



HER (6, 2)	FATHER (18, 8)	FOSTER (12, 6)
OTHER (16, 6)	LATHER (18, 8)	HATER (12, 6)
BOTHER (22, 10)	MASTER (16, 8)	LATER (12, 6)
MOTHER (18, 6)	FASTER (16, 8)	WATER (12, 6)

Figure 5.1. Initial state and goals for our example problem with tuple $(cost_{\pi_G^*}, cost_{\pi_{R_{ed}+R_{ing}}^*})$. The specific problem instance’s goal word is bold-faced.

that stacks of blocks spell words from top-to-bottom. Figure 5.1 illustrates the initial state and lists the 12 possible ‘goal words’ that will be the hypothesis set \mathcal{H} for our example. To extend BlockWords for two interacting agents, all actions in $A_{R_{ed}+R_{ing}}$ are performed in parallel when different blocks are involved, and actions that share blocks are modified to account for joint actions: putting down the same block that is picked up is considered a handover, and two consecutive blocks might be picked up or put down in a stack simultaneously. We omit pairs of actions that have race conditions, such as placing a block on top of one that is being picked up.

5.4.1 Assistive Interaction Example

We assume that R_{ing} only computes one responsive plan without replanning; so the decision of when to join can be evaluated as well. R_{ed} uses a planner to find an initial optimal plan π_G^* that solves their assigned goal G : spelling MASTER. As R_{ed} performs each action a_t in π_G^* , R_{ing} observes it and then computes $P(G | \mathcal{O}, r)$ where $\mathcal{O}_{\leq t} = \pi_{G, \leq r}^*$. Figure 5.2 presents these recognition results for several choices of foresight parameter ϵ . We observe marginal differences for smaller ϵ , but larger ϵ exaggerate the probability of more costly goals as each action is taken. We will use $\epsilon = 3$ because it maximizes the necessities, but the threshold $\tau = 0.3$ (slightly less than the greatest probability mass given to a goal at the beginning) yields the same propositions for all $\epsilon \leq 5$.

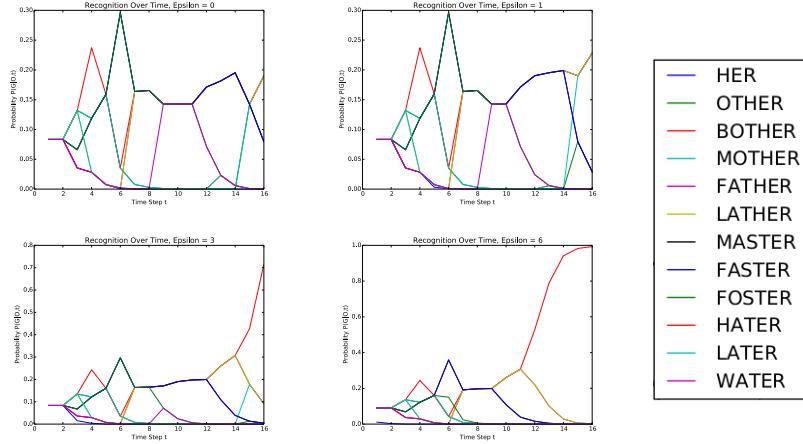


Figure 5.2. Recognized distribution over \mathcal{H} as each action of π_G^* is taken. The value of foresight parameter ϵ increases from left to right, top to bottom. The most costly goal, spelling BOTHER, becomes more likely as ϵ increases to look further ahead.

Because all the goal words share the same last two letters ER, we note that the necessity for ‘stack E on top of R’ and ‘place R on the table’ are always 1. So for any τ , these two conditions will be part of $\widehat{G_{R_{ed}}}$. When the distribution is more uniform at the earlier timesteps, any goal condition that frequently appears in \mathcal{H} has greater necessity. For example, observing the first two actions (R_{ed} takes H off T and places it on the table) does not disambiguate any of the goals so that the conditions ‘stack H on top of E’ and ‘stack T on top of E’ both have necessity 0.5; half the goals end in HER and the other half end in TER. However, both conditions cannot be true simultaneously so that $\widehat{G_{R_{ed}}} = G_{R_{ing}}$ has no solution and R_{ing} is unable to join in yet. After performing the sixth action (R_{ed} places S on top of T in Figure 5.3), the recognition algorithm identifies MASTER and FASTER as the most likely candidates and their shared goal conditions become the only ones with sufficient necessity—greater ϵ also deem their conflicting conditions necessary. The red lines show the conditions that are already satisfied, and the green dotted lines show the unsolved condition: ‘stack A on top of S’. $\pi_{R_{ing}}$ and $\pi_{R_{ed} \leftarrow R_{ing}}$ complete this together in 4 steps rather than R_{ed} completing it alone in 8 steps. R_{ed} then stacks M

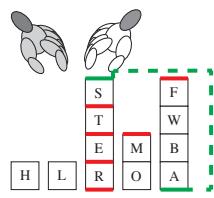


Figure 5.3. The state after R_{ed} performs the sixth action in its plan. The identified necessities are shown in red (satisfied) and green (unsatisfied) overlaid lines.

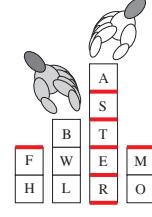


Figure 5.4. The state after R_{ing} performs its assistive responsive plan alongside R_{ed} 's plan. All the necessities are satisfied so that R_{ed} just needs to stack the final block and complete its spelling task.

on top of the stack in Figure 5.4 to complete G . Hence $H(\pi_{R_{ing}}) = 4$, indicating a helpful assistance in solving the task.

5.4.2 Adversarial Interaction Example

In the case that the assistive interaction example above was instead adversarial, we would need to consider the scenario for several durations d . Recall that R_{ing} 's goal is to ensure that G is never complete within d actions after the current time step. Thus, using the same example and again waiting until R_{ed} performs the sixth action, R_{ing} easily accomplishes their goal with any arbitrary plan if $d < 2$ and *almost* any arbitrary plan if $d < 10$. These cases are simpler because the quickest R_{ed} can accomplish the task is $\text{cost}_{\pi_G^*} = 16$ alone and $\text{cost}_{\pi_{R_{ed}+R_{ing}}^*} = 8$ with an assistive partner. We emphasize ‘almost’ because any $\pi_{R_{ing}}$ that helps to spell G 's goal word can complete G within the reduced duration if two agents can solve the task together within that time.

However, once d is great enough that R_{ed} can solve the plan on its own, then R_{ing} has to execute plans that prevent at least one of G 's conditions from being satisfied at every time step. For our BlockWords example, there is a ‘trivial plan’ $\pi_{R_{ing}}^{\text{hoard}}$ where R_{ing} picks up one of the required blocks, such as S , and then performs no-ops indefinitely. R_{ed} is unable to obtain the block held by R_{ing} and can never complete

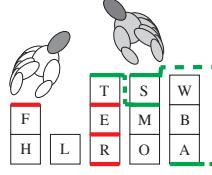


Figure 5.5. The state in Figure 5.3 two actions later, modified with respect to R_{ing} 's adversarial responsive plan $\pi_{R_{ing}}$. Stacking S on top of M undid two satisfied necessities while R_{ed} began to uncover the A block.

the task; thus $H(\pi_{R_{ing}}^{\text{hoard}}) = -\infty$. As most off-the-shelf automated planners use search in place of such logic, we instead expect $\pi_{R_{ing}}$ to continue to pick up and rearrange the blocks that will undo R_{ed} 's progress. Such $\pi_{R_{ing}}$ will usually have a finite negative helpfulness so that it is possible to find a (less optimal) solution $\pi_{R_{ed} \leftarrow R_{ing}}$. Figure 5.5 displays the state after unstacking S from the goal word and stacking it on top of M, but R_{ed} can still complete the task if this is the entire $\pi_{R_{ing}}$. R_{ing} will eventually need to replan and resume taking the tower of blocks apart. This back-and-forth interaction of R_{ing} 's deconstruction and R_{ed} 's reassembly emphasizes the need for planning and recognition to continuously update each other.

5.4.3 Independent Perspective

Lastly, let us consider two independent interactions where G'_1 is to spell the word HOWL and G'_2 is to spell the word WOLF, both using the same blocks. G'_1 is the simpler case because none of its blocks are involved in the necessities after R_{ed} performs the first six actions. Thus $\pi_{R_{ing}}$ will eventually pick up W and put it on top of L, which helps R_{ed} by reducing the number of actions needed to reach the A block. Then $\pi_{R_{ing}}$ will later pick up O in order to place it on top of W, but R_{ing} will first need to move the M block. R_{ing} will not place M on top of the stack of blocks spelling G 's goal word because that will prevent *success* from becoming true (see Figure 5.6) unless A was already put on top of S. Likewise, if R_{ing} had to move other blocks afterwards (in this case, they just pick up H and place it on top of O),

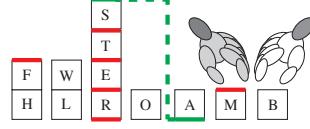


Figure 5.6. The state in Figure 5.3 four actions later, modified with respect to R_{ing} 's independent responsive plan $\pi_{R_{ing}}$ with a personal goal of spelling HOWL. R_{ed} does not use responsive planning and thus covers H with F, but R_{ing} also moves M instead of helping R_{ed} uncover A as they did during assistive interaction.

then they would not place the blocks on top of M because $\widehat{G}_{R_{ed}}$ contains a condition to leave nothing on top of the M block (in case G spells MASTER).

Because $\widehat{G}_{R_{ed}}$ also contains a condition to leave nothing on top of the F block in case G spells FASTER, G'_2 is more difficult to accomplish. The most R_{ing} can do is uncover the W, O, and L blocks and start to stack them. However, stacking them is not practical because R_{ing} will have to undo the stack in order to place them on top of F later. Thus more observations may be necessary before R_{ing} interacts in this case so that it can later update $\widehat{G}_{R_{ed}}$ and confirm that ‘stack nothing on top of F’ is not a necessity. Unless R_{ed} stacks a block on top of F, then this would unfortunately require R_{ing} to wait until G is already complete due to the ambiguity of the two goals. This presents an interesting path of future work where *both agents are simultaneously observing each other* for independent interaction.

5.5 Concluding Remarks: Responding through Observation

The majority of decision-making systems today define *tasks as concrete goals*, telling the agent(s) exactly what they must do in order to accomplish the assigned task. Though humans do not always describe the goal conditions or reward functions correctly [130, 183] such that the intelligent agent might execute unexpected actions that people will question or deem unsafe [9] (or, like the classic programming adage, do what the person says instead of what the person wants), their descriptions are of the form “affect/change the environment in this way.” Such a format does not

readily address *tasks with abstract goals* such as the three types of interaction discussed throughout this chapter: assistive, independent, and adversarial. Such tasks have descriptions of the form “help us”, “do not get in the way”, and “stop them”, respectively.

Although these task descriptions are vague, they usually become concrete over time once the agent knows the *means of accomplishing the abstract task*:

- “Help us *carry the groceries to the kitchen*.”
- “Do not get in the way *while we vacuum the carpet*.”
- “Stop them *from making too much noise*.”

Even if goal-driven agents R_{ed} do not communicate the concrete portion of the task directly, it is possible for the interactive intelligent agent R_{ing} to observe them both before and during the interactive experience. Using various recognition algorithms, including the ones described throughout Chapter 3, R_{ing} can take advantage of these observations to predict R_{ed} ’s goals and upcoming actions. Even when there is uncertainty between goals at the earlier stages of the interaction, the reason for this ambiguity is the overlapping satisfaction criteria between those goals and their correspondence to R_{ed} ’s most recent actions. Until further observations are available to disambiguate the concrete task, the *overlapping criteria likely need to solved regardless of R_{ed} ’s true goal*. This yields the necessities of R_{ed} ’s current task, which generate a concrete, intermediate goal that allows R_{ing} to initiate interaction while continuing to observe R_{ed} . Automated planning methods, including the ones described throughout Chapter 2, can find a solution to this intermediate goal.

The abstract nature of the task is generally evaluated more holistically than the associated concrete task as well. It is often easy to be successful with respect to the interaction’s description, but R_{ed} could deem R_{ing} ’s *level of success* lackluster:

- R_{ing} carries *only* a single bag of pretzels to the kitchen to help, but R_{ed} carries the remaining groceries including heavy milk cartons and canned food.
- R_{ing} treads around the carpet's perimeter to avoid getting in the way, but trips over the cord and disconnects the vacuum from the wall outlet. So R_{ed} has to stop and plug the vacuum back into the wall outlet.
- R_{ing} hides all the musical instruments and the television remote in another room, which delays R_{ed} from creating noise until they find the hiding spot.

We introduce helpfulness as a means of measuring the degree to which R_{ing} reduced R_{ed} 's efforts, and ideal values for this measurement depend on the type of interaction. It is likely that helpfulness can play a deeper role in the decision-making process so that the responsive plan adheres to the abstract task for the entirety of the interaction, rather than as a consequence of achieving each intermediate concrete task.

While this seems sufficient for closing the interaction loop because the interactive intelligent agent makes decisions to act with respect to their perceptions of others, there are a few more steps involved. Unlike closed-loop control in an environment devoid of decentralized agents, other agents such as R_{ed} *have their own autonomy*. This means they can observe R_{ing} and might change their own plans in an attempt to accommodate those predictions. R_{ed} can also become distracted or change their goals mid-execution, which will invalidate some, most, or all of R_{ing} 's expectations that guided the responsive planning so far. We explore these remaining issues for closed-loop interaction in more depth in the following chapters of this dissertation, but they ultimately tie into responsive planning.

CHAPTER 6

THE PRETCIL FRAMEWORK AND AN IMPLEMENTATION

I study how machines can play with people.

— Richard (Rick) G. Freedman,
elevator pitch opening

We just need to implement code for a planner and a recognizer, and then we write a little more code to connect them as they share information indefinitely. If we agree on the data structures beforehand to synchronize their representations, then this will not take long at all. It should be done before you submit grad school applications!

— Richard (Rick) G. Freedman,
famous last words when mentoring Roman Ganchin and Yi Ren Fung

We have covered enough topics to finally introduce the Planning and Recognition Together Close the Interaction Loop (PRETCIL) framework, an integrated approach for building interactive intelligent systems in which automated planning and recognition algorithms share information as previewed in Figure 1.1 and now detailed in Figure 6.1. The key element of the PRETCIL framework is that each step is not connected as a pipeline, but instead running concurrently and *the arrows show the flow of information shared between steps*. Open-loop interaction uses a pipeline architecture to repeatedly complete a step and pass its outputs as inputs to the next step. This causes the limitations to the interactive experience that Chapter 1 describes. Closing the interaction loop through integrating each step will not only improve the dynamic qualities of the interactive experience, but also allow each step to provide useful information that guides the others. Continuing from Chapter 5, we investigate a more complete integration of PAIR algorithms (see Chapter 3) and automated planning

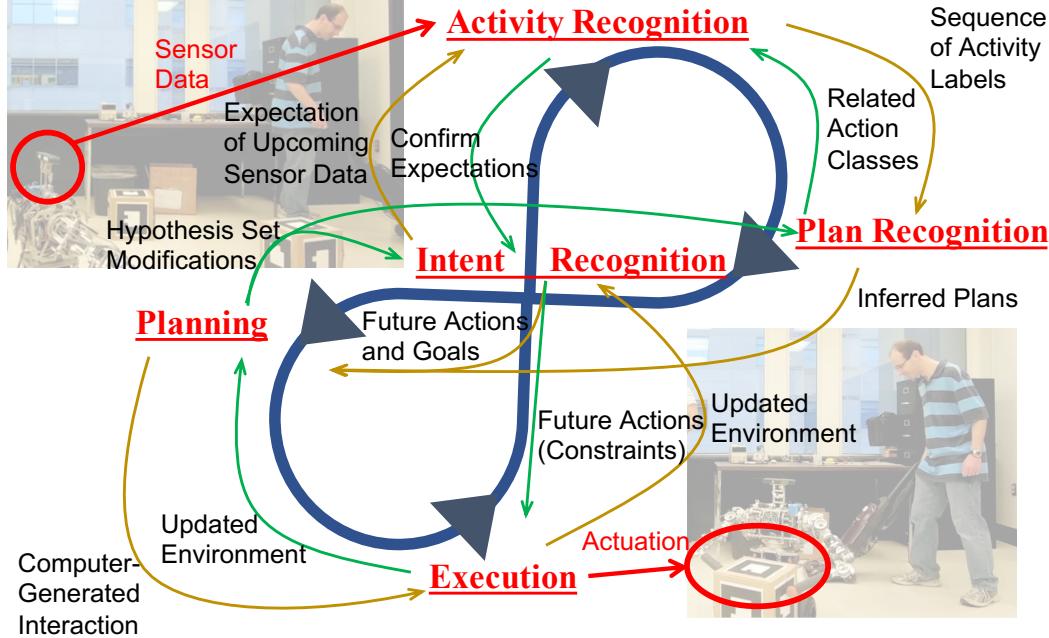


Figure 6.1. The PRETCIL framework’s planning and recognition interaction loop. Components pass information forward for processing (gold) or backwards to improve performance in later iterations (green).

algorithms (see Chapter 2) that closes the loop of an interactive intelligent system R_{ing} . Besides identifying and deciding *how to reach the currently perceived goals* of an observed user R_{ed} from the currently perceived state, this includes *predicting how others will respond to R_{ing} ’s responses* as well as *monitoring how well the interaction is going*.

To connect all the components, it is crucial that their inputs and outputs work together with a shared representation. Hence the choice of sensor data representations, intermediate data structures, and world representations must relate to the characteristics of the integrated algorithms that will produce or process them. Otherwise, an implementation of the PRETCIL framework can use any algorithm and/or model that is appropriate for the corresponding component(s). We chose to integrate recognition as planning (RaP, see Section 3.1.3) and responsive planning (see Section 5.2) *without* a dynamic prior in our preliminary implementation. In order to

make this implementation functional and feasible, this chapter also develops a state-space variation of these approaches, provides details about the program, and includes a postmortem describing the tribulations encountered.

6.1 Related Closed-Loop Interaction Approaches

Closed-loop interactive intelligent systems act with respect to the users' actions and intents, creating an ongoing dynamic and adaptive interactive experience. While many of today's commercial digital assistive agents appear to exhibit closed-loop interactive behavior, either the set of user inputs or agent outputs are often constrained to some preprogrammed behaviors. The primary applications of artificial intelligence in these systems include processing the input (for example, speech-to-text) or performing the output (for example, motion planning), but they do not actually understand the user better and make decisions based on such an understanding.

Several closed-loop interaction frameworks have recently been proposed, each integrating some approach for plan, activity, and/or intent recognition (PAIR) with another approach from the decision making literature. No one approach, including the PRETCIL framework, is superior in any way. In fact, each has trade-offs that support different scenarios and use cases. To provide a source of comparison, we briefly introduce the two other approaches of which we are aware.

6.1.1 Integrated Execution Monitoring and Automated Planning

As far as we know, Levine and Williams introduced PIKE [174, 176] as the first algorithm for closed-loop interactive agents. Designed for *cooperative team applications where all agents share the same global task*, such as coworkers in a factory domain, PIKE assumes that the agents involved in the interaction all share the same goal(s) and will be able to coordinate prior to the task execution. However, it is possible that agents have unique action niches based on their design—people can perform

dexterous tool-related tasks more effectively than a robot, but an industrial robot is generally strong enough to pick up very heavy objects.

Following the assumption that all agents share the same set of goals, PIKE first runs offline with the set of goals to generate a *library of centralized multi-agent plans* that solve each goal, including alternative solutions. Rather than store this plan library as a collection of action sequences, though, PIKE compiles all the plans into a single data structure called a Temporal Plan Network (TPN) [159]. Due to PIKE’s use of classical planners such as OPTIC [22] that account for durative actions, this data structure schedules the actions in the plan so that agents know *within which time intervals to execute actions* in order to *avoid timing-related plan execution failures*. PIKE shares the TPN with all other agents, both human and machine, because the target domains know that all agents will work together from the start without any ad-hoc team transformations [94].

The unique feature of PIKE’s TPNs is that they include *choice nodes* where an agent selects one of several tasks to perform. This addresses the alternative approaches for solving a single goal and allows overlapping subsequences of actions between solutions accomplishing different goal conditions. People have the most flexibility in these TPNs because they can select any of their available actions at corresponding choice nodes. The computational agents employing PIKE instead use their choice nodes as an *opportunity to complement the humans’ choices* and ensure that at least one action is applicable at their next choice node. PIKE annotates the TPN with *causal links* to indicate which actions’ effects satisfy preconditions of later actions. By *monitoring R_{ed} ’s actions* throughout the plan execution, particularly at choice nodes, PIKE then uses the annotations to refine the applicability of future actions and *resolve missing causal links through deciding how to act at R_{ing} ’s own choice nodes* while accounting for time constraints.

Levine later extended PIKE to RIKER [175], which extended the TPN to include probabilistic transitions at the choice nodes. Although this extension sounds simple, it encourages significant changes to interactive decision making that PIKE could not do. Reasoning over external factors, such as the weather and personal habits, RIKER has greater *foresight about teammates' future choice node selections* and dynamically adjusts the choice nodes' distributions. This enables computational agents employing RIKER to prepare their own actions further in advance, which is crucial in time-sensitive plan execution. Furthermore, RIKER has a *risk threshold parameter* that allows R_{ing} to select actions whose long-term effects may or may not guarantee successful plan execution—the probability distributions affect RIKER's risk analysis towards success. Empirical results indicate that increasing RIKER's risk threshold parameter correlates to R_{ing} deciding to take actions that more often result in plan failure, but also increases their helpfulness in the cases where the team is successful because R_{ing} is willing to participate sooner in the interactive experience. RIKER's plan libraries typically contain plans with forced start-time delays for R_{ing} , which guarantees no-op as an option at their early choice nodes.

6.1.2 Confirmation and Negotiation between Recognition and Planning

Around the same time that we introduced responsive planning [82], Geib et al. introduced a framework for interactive intelligent agents [90] that combines the ELIXIR plan recognition algorithm from Section 3.1.2 [89] with the Planning using Knowledge and Sensing (PKS) [220]. Similar to PIKE and RIKER, agents employing this framework are *strictly involved in assistive-type interactions*. Discussions with Geib confirmed that ELIXIR is now the name of the overall framework, and the plan recognition component is renamed LEX_{rec} while the component running PKS is called LEX_{gen}. We thus use the same naming convention, despite whatever names appeared in the original work.

LEX_{rec} begins recognition in its typical iterative parsing fashion. As users perform actions, the algorithm updates the probabilistic likelihoods of various parses as explanations according to the combinatory categorical grammar (CCG). The assistive agent does not involve themselves in the interaction until some set of explanations have a likelihood above a specified threshold. Unlike RIKER’s risk threshold parameter, this threshold accounts for confidence in explaining R_{ed} ’s plan.

Confidence matters because the step after LEX_{rec} is actually not LEX_{gen} , but an unnamed *negotiation component*. To our knowledge, the negotiation component is the first *explicit form of communication in a closed-loop interactive system*. Agents employing Elixir use this component for two purposes:

1. To confirm the recognized task is correct and
2. To request permission to perform remaining tasks in the explanation associated with the confirmed task.

Sorting the threshold-surpassing explanations in order from most likely to least likely, R_{ing} asks R_{ed} whether the name of the explanation’s complex category (this is not always the same as the root-result of the explanation) is their intended task. R_{ed} simply responds to either confirm or deny the query, and denial causes R_{ing} to propose the next most-likely explanation in its sorted list. Failure when R_{ed} denies all explanations surpassing the confidence threshold is thus less severe. The plan’s execution does not necessarily fail because R_{ing} continues to run LEX_{rec} , but R_{ing} did impinge on R_{ed} ’s cognitive load.

When R_{ed} confirms one of R_{ing} ’s proposed explanations σ , then the second phase of negotiation begins where R_{ing} investigates the names of incomplete tasks in σ . To avoid any challenges of multi-agent coordination, Elixir assumes that R_{ing} and R_{ed} act on their own for all tasks without overlap of actions and resources. For each remaining task that satisfies this assumption, R_{ing} requests permission from R_{ed} to perform the

task. Like in the first phase of negotiation, R_{ed} simply responds to each request with confirmation or denial. Once this phase of negotiation concludes, R_{ing} knows the list of permissible tasks that are assigned to them. R_{ed} should complete the denied tasks, which gives R_{ing} the freedom to run LEX_{gen} . Elixir translates the assigned task names into goal conditions for LEX_{gen} , which then computes a corresponding sequence of actions. R_{ing} performs this plan in parallel with R_{ed} 's continued plan execution, and the two agents no longer need to interact with each other.

Although LEX_{gen} ends the interaction framework such that Elixir appears to be open-loop during plan execution, PKS happens to account for sensing actions that still make it aware of the environment and indirectly aware of R_{ed} 's future actions. For example, if R_{ing} has a task that involves taking items from a drawer, then the PKS-computed plan should enforce a sensing action to determine whether the drawer is open. Whether R_{ed} has been opening or closing the drawer throughout the interaction, R_{ing} is able to act accordingly. Furthermore, the *negotiation step establishes a mutual understanding* between both R_{ed} and R_{ing} ; each agent knows which task(s) the other will accomplish so that R_{ed} can at least plan around R_{ing} through expectations. Humans are still far better at adaptive decision making than machines, and *giving people sufficient information to prepare to work around the machines is a very effective, often low-cost approach to creating an acceptable interactive experience.*

6.2 Overview of PRETCIL Framework

The primary feature of the PRETCIL framework is the *integration of perceiving* interaction partners *and making decisions* about how to respond such that these two aspects influence each other throughout the interactive experience. Activity recognition abstracts low-level observations, such as raw sensor information, into higher-level action labels/descriptions for both plan recognition and intent recognition. These two forms of recognition estimate R_{ed} 's goals and predict what actions they will take to

achieve them. With these predictions, a planner selects R_{ing} 's actions to respond to the estimated goals with respect to R_{ed} 's expected actions. If the decisions are too high-level, then execution determines how R_{ing} can perform the chosen actions. Given this response, intent recognition can also predict how R_{ed} will respond to R_{ing} 's action. Lastly, activity recognition completes the interaction loop by again abstracting what the interaction partner does, which also confirms whether or not the predicted response is correct.

6.2.1 Example: Perception via Planning as Recognition

When the interactive experience begins, R_{ing} has no model of the interactive partners R_{ed} . This means that R_{ing} is not aware of what they want to do and must first observe them in order to make any informed decisions. Virtual environments encode information about the program, including the interface, to easily describe R_{ed} 's performed actions at a higher-level representation. In real-world environments, R_{ing} will likely observe everything through raw sensor data, which contains *no semantic information* without some degree of data processing. Activity recognition should process the data stream to identify R_{ed} and how they are affecting the world, generating these higher-level action representations.

The plan and intent recognition components receive these actions as observations. For probabilistic RaP algorithms, the plans generated for each hypothesis's comparisons serve as the output for plan recognition. The plans specifically provide information about *what the user is expected to do by themselves* when satisfying each set of completion criteria. The costs of these recognized plans determine the distribution over the different hypotheses through probabilistic RaP as well. This distribution is the output for intent recognition because the distribution identifies *how likely each criteria motivates the user's actions*.

6.2.2 Example: Decision Making via Responsive Planning

When deciding how to respond to R_{ed} 's possible intents, it is important to consider the long-term interaction as much as the current action being taken. This is especially important at the beginning of the interactive experience because their initial actions are often relevant to completing multiple criteria—this ambiguity is the worst-case distinctiveness [156] that measures the maximum number of actions that the start of two optimal plans can share despite solving different goals. Furthermore, if R_{ing} ‘assists’ R_{ed} towards completion criteria that do not belong to R_{ed} 's intents, then R_{ing} might hinder the experience and reduce R_{ed} 's trust and willingness to work with them.

Our implementation of the PRETCIL framework identifies the necessities (see Section 5.2) to account for this concern. With respect to the distribution over the possible goal criteria that the intent recognition component provides, the necessities are the weighted sum over the parts of each completion criteria. It is rarely the case that different intents are mutually exclusive of each other; so R_{ing} can complete the common tasks that progress towards all the likely completion criteria and ideally begin to assist R_{ed} until they perform some action that further disambiguates their intent.

The necessities generate an intermediate goal for R_{ing} , and we can reuse probabilistic RaP’s planner to find a sequence of actions that will accomplish this generated goal. The planner generates the plan from the current state as output, assigning actions to both R_{ing} and R_{ed} until they both accomplish the intermediate goal. Although R_{ed} is not aware of this joint plan between the agents, R_{ing} follows the plan and has an expectation of what R_{ed} might do.

6.2.3 Example: Simple Execution Monitoring

Although R_{ed} does not know the joint plan, R_{ing} assumes that R_{ed} will perform the actions that the joint plan assigns to them. Through this assumption, R_{ing} 's future actions will execute successfully without uncertainty. Our implementation of the PRETCIL framework thus uses this plan for the second purpose of intent recognition: to *predict how R_{ed} will respond to R_{ing} 's actions*. If R_{ed} 's action returned from the activity recognition component matches the joint plan's, then we assume that the interaction is going smoothly and R_{ing} executes their next action according to the joint plan. If R_{ed} 's action does not match, then there is a chance that R_{ing} recognized incorrectly and reassesses the completion criteria with the newest observation(s). We call this trivial execution monitoring system that compares R_{ed} 's actual actions to the expected ones Single-Plan Action Matching (SPAM).

SPAM completes the interaction loop because it continues to observe R_{ing} 's behavior and use those observations to influence what R_{ed} decides to do during the interaction. From a metareasoning perspective, SPAM determines whether to rerun the plan recognition, intent recognition, and automated planning components (when the match fails) or to skip directly to the execution component (when the match succeeds). The former case allows R_{ing} the *flexibility to adapt* to any response from R_{ed} , and the latter case *avoids spending seemingly unnecessary resources* when the stored plan (and the predictions on which it is established) still hold. However, increasing the set of joint plans for alternative matching options could make the latter case apply more often—this supports using PIKE [174, 176] and RIKER [175] from Section 6.1.1 above as ideal replacements for SPAM in future iterations of our implementation.

6.3 Overview of Implementation

The above example of the PRETCIL framework integrates probabilistic RaP [227] with responsive planning [82], actively searching for possible plans during the

interaction and then getting the weighted average over the plans’ corresponding goal conditions to create an “intermediate goal”. Although the weighted average formulation in Definition 49 assumes that the states have a STRIPS-style representation, many real-world problems are not easy to represent as a set of logic predicates. Furthermore, *using off-the-shelf planning software exchanges the ease of implementation with the need to daisy-chain multiple pieces of software together and sacrifice efficiency* from separate processes in the system not taking advantage of shared information. We¹ developed a software library in C/C++ that addresses these issues for our initial implementation of the PRETCIL framework based on the example. The library’s more generic state-space representation allows developers to write customized code and represent their domains that are more difficult to capture in STRIPS alone, but this feature comes at the price that developers need to implement some additional functions on their own. In this section, we explain our software library’s general structure and show what developers need to implement for their specific application.

6.3.1 Library Structure

Figure 6.2 illustrates the primary components of the library that we developed for our implementation of the PRETCIL framework. The library’s built-in classes are all defined using C++ templates for state and action representations. In general, developers need to implement their own domain information and application. This grants them the *freedom to use data structures that best suit their representation* in addition to the freedom of expressing how the world changes using traditional code.

¹The team at early stages of development included Richard G. Freedman, the author of this dissertation, alongside student mentees Roman Ganchin and Yi Ren Fung. Upon joining SIFT, Freedman completed the library and implementation independently using internal research and development funding.

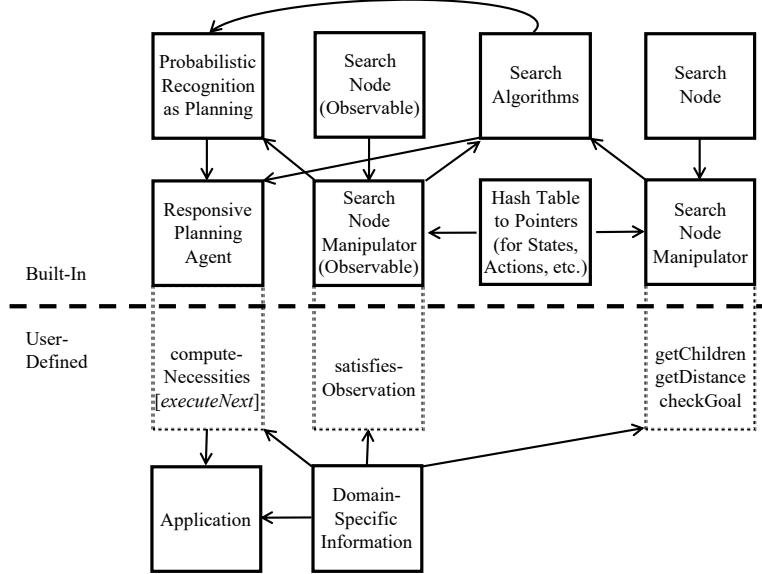


Figure 6.2. Current components in the library and their relations to each other. The arrows show the flow of information and/or use of an instance of the class. Extended dotted-border boxes contain the current functions that the developer needs to define.

6.3.2 State-Space Planning

Both probabilistic RaP and responsive planning require the use of an underlying planner, which we implemented as a suite of heuristic forward-*search algorithms* (see Sections 6.4 and 6.5 for more details). Although many search-based off-the-shelf classical planners rely on optimized domain-independent heuristics for specific state representations that correspond to some subset of PDDL specifications, we instead pass the responsibility of *heuristic implementation* to the developers because there are no constraints to their state representations. Furthermore, there are typically more accurate domain-specific heuristics.

In addition to computing distances between states, the developer needs to *both identify whether a state satisfies the goal criteria and generate the successor states* during node expansion. These functions are all defined within the *search node manipulator*-derived class, which performs operations given *search node* inputs. Search nodes store the state and parent information, which does not require domain-specific

modifications—developers can simply use `typedef` for search nodes with the instantiated templates rather than create a subclass.

6.3.3 Probabilistic Recognition as Planning

To account for observation sequences, RaP algorithms modify the domain to include a “progress bar” that fills as each observation is satisfied. STRIPS-represented domains do this through the inclusion of new fluents that count observation succession, and our library augments the observation progress bar in the *search node (observable)* and *search node manipulator (observable)* classes. These are subclasses of the *search node* and *search node manipulator* classes, respectively. The developer needs to implement one new function for the manipulator to properly adjust the progress bar: *evaluating when an observation is satisfied*. Besides observing actions like the original algorithms, we also include the previous and current states to allow *observations of the state-transitions and/or sensed features* [252]. It is up to the developer to decide how the observed states in the transition and/or action match an observation, based on both their representation and which information matters for their application.

6.3.4 Responsive Planning Agent

The *responsive planning agent* manages the interactive experience through the integration of automated planning and recognition that defines the PReTCIL framework. A single responsive planning agent stores an observation sequence and state-space planner per agent in the application’s environment, which serves as their models (that is, how R_{ing} internally describes R_{ed}). This information allows the *probabilistic recognition as planning* component to compute a distribution over the hypothesized sets of goal conditions per agent. Due to the developer’s freedom to implement states (and thus goal conditions) as they see fit, it is also necessary for the developer to

implement a *function that uses this distribution to generate a set of “intermediate goal” conditions* called necessities.

The responsive planning agent also has a joint-multi-agent planner that assumes all agents act in a centralized fashion, which plans out all their actions. As a trivial execution monitoring system, SPAM (see Section 6.2.3) simply checks whether other agents follow the joint plan. If they do, then the responsive planning agent continues to perform their next action in the joint plan and does not need to rerun recognition as planning, necessities computation, or joint-multi-agent planning. If the expected action does not match, then the agent uses the updated observation sequence to rerun these components and generate a new joint plan. Because the developer defines the domain’s actions, we recommend they also *override the function defining which action to perform next*. If the joint-plan does not exist or is exhausted, then it throws an error, but the developer can provide a default action to perform instead.

6.4 Recognition as Planning using Heuristic Search

Heuristic search serves as the underlying process of most classical planning methods and is a viable approach for finding solutions to problems that cannot be represented classically. However, the domain-independent classical planning heuristics have been the key to *simultaneously handling the two objectives in recognition as planning algorithms*—they solve the original problem and find plans that do (not) contain the observed agent’s action sequence. Referring to the equations throughout Section 3.1.3, the new pairs of planning problems $\mathcal{P}_{\mathcal{O}}^G$ and $\mathcal{P}_{\neg\mathcal{O}}^G$ encode two tasks into their set of goal conditions. Preserving G indicates the first task: solving the original problem. Including $\{p_m\}$ or its negation introduces the second task: following or disregarding the observation sequence. We can only add p_m to the current state upon performing all the actions in \mathcal{O} in order throughout the plan; each p_i cannot be removed once it is added to the state. This means that solving the original problem

$\mathcal{P}^G = (F, A, I, G)$ will inherently solve one of these two new planning problems, but which one would be arbitrary.

The domain-independent heuristics developed for classical planning representation languages can naturally address solving both tasks because the problem itself does not affect the estimation algorithm. Both the original and new problems' STRIPS representations provide all the information to create the heuristic. Simply including p_m or $\neg p_m$ as a goal condition is enough to alter how the heuristic evaluates each node in the search space. To implement RaP algorithms without assuming a STRIPS-like representation language, this means that we must adjust the heuristic that guides the search to target goal states that *address both objectives simultaneously* for each new planning problem. This is not the same as a multi-objective search problem [187] because it is a mandatory conjunction of task criteria rather than a choice of cost trade-offs to condition the solution set.

This section introduces two approaches that we designed in an attempt to address this challenge while searching in a generic state space. Although there are flaws in each approach, exploring their pros and cons provides new insights. The scientific community generally frowns upon negative results, and this dissertation is one of the few opportunities to disseminate the findings. We hope the reader learns something useful or identifies how to build off this work to create something successful. For the actual approach in our library's implementation, we direct the reader to Section 6.5.

6.4.1 Notation

When searching over a state space that is not constructed from a classical planning representation language, we must manually implement the search space and heuristic. Specifically, we only assume that the following are available from the initial planning problem implementation that is encoded in some programming language \mathbb{L} :

- The set of states S is the set of all assignable values to a set of variables² (primitives, instances of classes/structures, data structures, etc.),
- Each action $a \in A$ is a deterministic function mapping $a : S \rightarrow S \cup \{nil\}$, which changes the input state when it can be applied and returns *nil* otherwise,
- The initial state $I \in S$ is a specific assignment of values to all the variables in the state definition,
- The goal check function $G : S \rightarrow \{\text{true}, \text{false}\}$ evaluates to *true* if and only if the input state satisfies the task criteria,
- And the heuristic function $h_G : S \rightarrow \mathbb{R}^{\geq 0}$ estimates the cost of a plan satisfying G starting from the input state.

When encoding \mathcal{O} in \mathbb{L} , we allow components of the transition function's input and output to be observed as a single observation.

Definition 55. *An observed transition is a tuple*

$$(\text{before}(s, a), \text{doing}(a), \text{after}(a(s), a))$$

*that represents an agent performing action $a \in A$ while in state $s \in S$. before and after are abstraction functions that map states to some subset of their features that are relevant to the performed action. If nothing about a state is relevant, then *nil* is returned. doing is an abstraction function that returns the relevant portion of the name of the performed action (which allows obfuscated observations [167]) or *nil*.*

²This is similar to defining a SAS+ [16] state because computer memory is finite, and SAS+ variables may be assigned any value within a finite domain. However, classical planning representation languages do not create new objects; all possible values are predefined per variable's domain. So this is more general unless every possible binary encoding within a memory address space is enumerated.

This representation not only accounts for Sohrabi et al.'s inclusion of observable state features [252], but also addresses the fact that classical planning representation languages often embed relevant before/after information about the state in the action name. To account for the observation sequence in our heuristic search solver, we modify the search space using a counter similar to the $\{p_0, \dots, p_m\}$ fluents:

Definition 56. *The observation space is an augmented state space $\mathcal{D}_{enum}^{\mathcal{O}}$ that accounts for \mathcal{O} through vertices $S_{\mathcal{O}}$ and edges from instances of elements of $A_{\mathcal{O}}$ where:*

- $S_{\mathcal{O}} = \{(s, i) \mid s \in S \wedge i \in \{0, 1, \dots, m = |\mathcal{O}|\}\}$
- $a_{\mathcal{O}} \in A_{\mathcal{O}}$ is a deterministic function mapping $a_{\mathcal{O}} : S_{\mathcal{O}} \rightarrow S_{\mathcal{O}} \cup \{nil\}$ such that

$$a_{\mathcal{O}}(s, i) = \begin{cases} nil & \text{if } a(s) = nil \\ (a(s), i+1) & \text{if } o_{i+1} \text{ abstracts} \\ & \text{from } (s, a, a(s)) \\ (a(s), i) & \text{otherwise} \end{cases}$$

The observation space effectively makes $(m + 1)$ copies of the original search space $\mathcal{D}_{enum,0}, \mathcal{D}_{enum,1}, \dots, \mathcal{D}_{enum,m}$ and transitions from copy $\mathcal{D}_{enum,i}$ to $\mathcal{D}_{enum,i+1}$ whenever the observed transition matches the next $o_{i+1} \in \mathcal{O}$. It follows that the initial state, similar to the traditional recognition as planning definition, is $(I, 0)$. Likewise,

$$G + O : (s \in \{s \in S \mid G(s) = \text{true}\}, |\mathcal{O}|) \mapsto \text{true} \text{ and}$$

$$G + \overline{O} : \left(s \in \{s \in S \mid G(s) = \text{true}\}, i \in \mathbb{Z}_{<|\mathcal{O}|}^{\geq 0} \right) \mapsto \text{true}$$

are the new goal check functions in the observation space for \mathcal{P}_O^G and $\mathcal{P}_{\neg O}^G$ respectively.

6.4.2 Approach 1: Sequence of Heuristic Searches

As a baseline approach to solving problems \mathcal{P}_O^G and $\mathcal{P}_{\neg O}^G$ for each $G \in \mathcal{H}$, we propose an algorithm that runs multiple instances of heuristic search over the observation space.

The modifications made to solve $\mathcal{P}_{\neg O}^G$ are very minimal aside from the conversion to observation space because it is likely that many solutions to \mathcal{P}^G will not contain O as a subsequence [189]. Rather than precompute a plan for \mathcal{P}^G and assume that it will have the same cost as some other plan that will never contain O , we introduce a trivial extension to the original heuristic function that we call BOOM (Block Off Observation m):

$$h_{G+\overline{O}}(s, i) = \begin{cases} \infty & \text{if } i = |\mathcal{O}| \\ h_G(s) & \text{otherwise} \end{cases}.$$

BOOM avoids states in $\mathcal{D}_{enum,m}$. This naïve search *lazily pursues each solution to \mathcal{P}^G sequentially until it violates being a solution to $\mathcal{P}_{\neg O}^G$* . In the case that many solutions happen to contain O as a subsequence, BOOM will be inefficient because it does not abandon undesirable paths to the goal until the last possible opportunity (when the observation sequence becomes a subsequence).

The modifications made to solve \mathcal{P}_O^G are not quite as simple and involve a two-part search algorithm to *address each implicit task independently*: matching the observation sequence and reaching a goal state. As the observation sequence must be a subsequence of the plan before reaching a state that satisfies the goal conditions, there is a *priority for matching the observation sequence first*. This first planning problem \mathcal{P}^O in the observation space uses the traditional initial state $(I, 0)$, but replaces the goal check function with

$$O : (s \in S, |\mathcal{O}|) \mapsto \text{true}$$

that only cares about finding some state in $\mathcal{D}_{enum,m}$. To guide this search process, we introduce a trivial heuristic that only pays attention to the counter

$$h_O(s, i) = |\mathcal{O}| - i.$$

The solution to \mathcal{P}^O is the prefix of the plan solving \mathcal{P}_O^G as it addresses the first implicit task of containing the observation sequence.

The second planning problem $\mathcal{P}_O^{G'}$ conceptually continues from the end of the prefix to find a state that satisfies G . However, we perform heuristic search in the *original search space with the given goal check G and heuristic function h_G* because we already satisfied the criteria of matching the observation sequence. The only difference is the new initial state I' , which is the goal state $(I', |\mathcal{O}|)$ that the prefix plan reaches. The solution to $\mathcal{P}_O^{G'}$ is the suffix of the plan solving \mathcal{P}_O^G ; thus the two sequences of actions are appended to obtain the actual plan $\hat{\pi}_{G+O}^*$.

When all action transitions in the state space \mathcal{D}_{enum} are *bidirectional/reversible*, we *do not need to worry about the Sussman Anomaly* despite addressing the two subproblems sequentially because we *restart the heuristic search algorithm between them*. This means that backtracking to a previous state s in the original search space is possible, but it would technically correspond to distinct states $(s, i < |\mathcal{O}|)$ and $(s, |\mathcal{O}|)$ in the observation space. However, caution is necessary when there exists a *unidirectional/irreversible action $a_{//}$* in \mathcal{D}_{enum} because the solution to \mathcal{P}^O might include this action. If all the goal states satisfying G that are reachable from I are not also reachable from I' , then the *Sussman Anomaly applies* because $\mathcal{P}_O^{G'}$ will not have a solution—the search cannot undo $a_{//}$ from the prefix plan to reach the goal states.

The overall FIREWORKS (Find an Initial state REcognized With the Observations, then Resume K Searches) algorithm optimizes this two-search procedure by only solving \mathcal{P}^O once because it is the same for all $G \in \mathcal{H}$. Figure 6.3 illustrates FIREWORKS over a search space. Because there might be multiple optimal solutions to \mathcal{P}^O that would place $\mathcal{P}_O^{G'}$'s initial state closer to or further from a respective

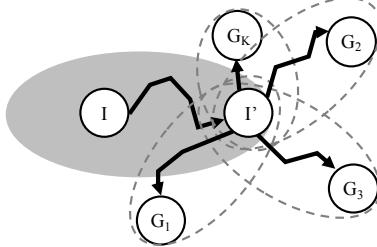


Figure 6.3. An illustration of the FIREWORKS algorithm overlaid on the original search space. The prefix path ($I \rightarrow I'$) solving \mathcal{P}^O requires a single search (solid shaded), but each goal check must search (dotted borders) for its own suffix ($I' \rightarrow G_{1 \leq i \leq K=|\mathcal{H}|}$) solving \mathcal{P}_O^G .

goal state, there is *no longer a guarantee that the solution to \mathcal{P}_O^G is optimal*, which is a key assumption for the traditional RaP algorithms. FIREWORKS further fails to guarantee optimal solutions to \mathcal{P}_O^G because the prefix of an optimal solution to \mathcal{P}_O^G is not necessarily an optimal solution to \mathcal{P}^O , which is possible when there are missing observations in \mathcal{O} .

6.4.3 Approach 2: Extended Heuristic for Traditional Search

In contrast to the amalgamation of various search problems above, we also propose a more uniform approach to solving \mathcal{P}_O^G and \mathcal{P}_{-O}^G . In particular, we only perform a single heuristic search per problem and introduce a pair of heuristics that can address both implicit tasks simultaneously. Although we will use best-first search where the priority queue *sorts the frontier using only the heuristic value*, we assume that the original priority value function is like the one used in A* search $f_G(s) = g(I, s) + h_G(s)$ rather than just $h_G(s)$. In this case, $g(s_1 \in S, s_2 \in S)$ is the exact cost of the path from state s_1 to state s_2 .

The Distance-to-Observation Progress Linear Ratio (DOPLR) balances between the two tasks of reaching the goal and (not) matching the observation sequence

$$h_G^O(s, i) = \frac{f_G(s)}{i+1} \text{ and } h_G^{-O}(s, i) = \frac{f_G(s)}{|\mathcal{O}|-i+1}.$$

The numerator approximates the remaining or total cost to the goal, and a cheaper cost is preferred. The denominator counts the index of the observation sequence up to which it is a subsequence of the current path. As the index increases, h_G^O decreases to show progress towards satisfying the subsequence goal condition (solving \mathcal{P}_O^G), and h_G^{-O} increases to show progress away from satisfying the non-subsequence goal condition (solving \mathcal{P}_{-O}^G). These denominators are similar to the heuristic that FIREWORKS uses to find the prefix plan.

The DOPLR heuristics have some useful properties to ensure that both tasks are addressed in tandem. The first property shows that one task will properly handle tie-breaking for the other. The proofs clearly come from the design of the ratios as explained above.

Lemma 3. *Let $(s_1, i_1), (s_2, i_2) \in S_O$ be two states in the observation space. The following statements hold:*

- If $i_1 = i_2$ and $f_G(s_1) < f_G(s_2)$, then $h_G^O(s_1, i_1) < h_G^O(s_2, i_2)$ and $h_G^{-O}(s_1, i_1) < h_G^{-O}(s_2, i_2)$.
- If $f_G(s_1) = f_G(s_2)$ and $i_1 < i_2$, then $h_G^O(s_1, i_1) > h_G^O(s_2, i_2)$ and $h_G^{-O}(s_1, i_1) < h_G^{-O}(s_2, i_2)$.

The second property reveals that we can provide a *relative* bound on the set of states that are explored before reaching a particular state. We emphasize the word ‘relative’ because we *constrain the states considered to those on the frontier rather than in the those reachable from the initial state*, which is a less traditional bound in state space exploration.

Theorem 2. *Let $(s, i) \in S_O$ be a state in the observation space. Then before exploring (s, i) during heuristic search from $(I, 0)$ with heuristic h_G^O , all states $(s', i') \in S_O$ on the frontier satisfying*

$$f_G(s') < f_G(s) \cdot \frac{i' + 1}{i + 1}$$

must have been explored first.

Theorem 3. Let $(s, i) \in S_{\mathcal{O}}$ be a state in the observation space. Then before exploring (s, i) during heuristic search from $(I, 0)$ with heuristic $h_G^{-\mathcal{O}}$, all states $(s', i') \in S_{\mathcal{O}}$ on the frontier satisfying

$$f_G(s') < f_G(s) \cdot \frac{|\mathcal{O}| - i' + 1}{|\mathcal{O}| - i + 1}$$

must have been explored first.

The constraint considering only states on the frontier *does not guarantee the generation of some states in observation space* because one of those state's ancestors must have been explored first. There is no guarantee of this happening either due to the ordering that Theorems 2 and 3 impose. In particular, the denominator in the ratios can allow a “runaway” situation where the search pursues a single path without considering others. In particular, $h_G^{\mathcal{O}}$ can become too greedy for satisfying the next observation and $h_G^{-\mathcal{O}}$ can become too averse to satisfy the next observation. However, this is not always the appropriate action to take. $\mathcal{P}_{\mathcal{O}}^G$'s solution accepts actions outside those in \mathcal{O} such that $h_G^{\mathcal{O}}$ should not always assume the first action satisfying an observation is the one observed, and $\mathcal{P}_{-\mathcal{O}}^G$'s solution accepts some incomplete subsequence of \mathcal{O} such that $h_G^{-\mathcal{O}}$ should consider some satisfied observations from time-to-time.

Corollary 1. The path found using heuristic search with heuristic $h_G^{\mathcal{O}}$ is not always optimal even if h_G is admissible.

Proof: (by example) Suppose we have the observation space shown in Figure 6.4 based on a 3×3 GridWorld instance with an obstacle in the center and observation sequence $\mathcal{O} = up$. The only state that satisfies G is g and the initial state is I . Let h_G be the oracle heuristic that always outputs the optimal cost to reach a goal state (this heuristic is admissible because the exact cost is never an overestimate). Then

the optimal solution to \mathcal{P}_O^G is the path $(I = (2, 2), 0) \rightarrow ((1, 2), 0) \rightarrow ((0, 2), 0) \rightarrow (g = (0, 1), 1)$ that *matches the only observation as the last action in the plan*. However, exploring $(I, 0)$ adds both states $((1, 2), 0)$ (via action *left*) and $((2, 1), 1)$ (via action *up*) to the frontier with heuristic values:

- $h_G^O((1, 2), 0) = \frac{1+2}{0+1} = 3$
- $h_G^O((2, 1), 1) = \frac{1+4}{1+1} = \frac{5}{2}$

Heuristic search selects the state on the frontier with the least heuristic value to explore next, which is $((2, 1), 1)$. This expansion adds two more states to the frontier, $((2, 0), 1)$ and $((2, 2), 1)$, with heuristic values:

- $h_G^O((2, 2), 1) = \frac{2+3}{1+1} = \frac{5}{2}$
- $h_G^O((2, 0), 1) = \frac{2+3}{1+1} = \frac{5}{2}$

As both of these states have a lesser heuristic value than $((1, 2), 0)$, and according to Theorem 2, the heuristic search explores them first. These expansions add the following states and heuristic values to the frontier:

- $h_G^O((1, 2), 1) = \frac{3+2}{1+1} = \frac{5}{2}$
- $h_G^O((1, 0), 1) = \frac{3+2}{1+1} = \frac{5}{2}$
- $h_G^O((2, 1), 1) = \frac{3+4}{1+1} = \frac{7}{2}$ (graph search would ignore this duplicate state)

Selecting either of $((1, 2), 1)$ or $((1, 0), 1)$ will similarly add the following states and heuristic values to the frontier:

- $h_G^O((0, 2), 1) = \frac{4+1}{1+1} = \frac{5}{2}$
- $h_G^O((2, 2), 1) = \frac{4+3}{1+1} = \frac{7}{2}$ (graph search would ignore this duplicate state)
- $h_G^O((0, 0), 1) = \frac{4+1}{1+1} = \frac{5}{2}$

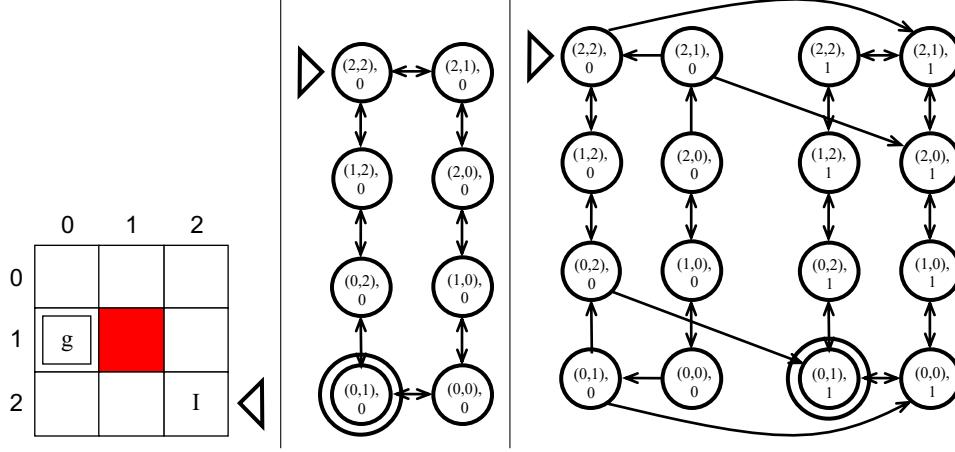


Figure 6.4. An observation space (right, original state space is in the center based on the grid on the left) that illustrates how to find a non-optimal path from $(I, 0)$ to $(g, 1)$ first using heuristic h_G^O when h_G is the oracle heuristic.

- $h_G^O((2, 0), 1) = \frac{4 + 3}{1 + 1} = \frac{7}{2}$ (graph search would ignore this duplicate state)

Lastly, selecting either of $((0, 2), 1)$ or $((0, 0), 1)$ will similarly add the following states and heuristic values to the frontier:

- $h_G^O((0, 1), 1) = \frac{5 + 0}{1 + 1} = \frac{5}{2}$
- $h_G^O((1, 2), 1) = \frac{5 + 2}{1 + 1} = \frac{7}{2}$ (graph search would ignore this duplicate state)
- $h_G^O((1, 0), 1) = \frac{5 + 2}{1 + 1} = \frac{7}{2}$ (graph search would ignore this duplicate state)

Regardless of the order in which heuristic search selects the states with heuristic value $5/2$, it will expand the goal state $((0, 1), 1)$ and thus find a path of cost 5 before considering the state $((1, 2), 0)$ that lies along the optimal path above of cost 3. \square

Corollary 2. *The path found using heuristic search with heuristic h_G^O is not always optimal even if h_G is admissible.*

Proof: (by example) Suppose we have the observation space shown in Figure 6.5 based on five states $S = \{I, x, y, z, g\}$ and observation sequence $\mathcal{O} = o_1, o_2$ where the observations are the state transitions $o_1 = I \rightarrow y$ and $o_2 = z \rightarrow g$. The only state that

satisfies G is g and the initial state is I . Let h_G be the oracle heuristic that always outputs the optimal cost to reach a goal state (this heuristic is admissible because the exact cost is never an overestimate). Then the optimal solution to \mathcal{P}_O^G is the path $(I, 0) \rightarrow (y, 1) \rightarrow (g, 1)$ that only matches the first observation (not the second one), and the *first action of the plan satisfies that observation*. However, exploring $(I, 0)$ adds states $(x, 0)$ and $(y, 1)$ to the frontier with heuristic values:

- $h_G^O(x, 0) = \frac{1+2}{(2-0)+1} = 1$
- $h_G^O(y, 1) = \frac{1+1}{(2-1)+1} = 1$

Heuristic search selects the state on the frontier with the least heuristic value to explore next, but both states have the same heuristic value. If the tie-breaking strategy [12] selects $(x, 0)$ first, then its expansion adds the state $(y, 0)$ to the frontier with heuristic values:

- $h_G^O(y, 0) = \frac{2+1}{(2-0)+1} = 1$

The tie-breaking strategy again selects the next state in the frontier to explore. In the case that it selects $(y, 0)$ first, then heuristic search adds the following states and heuristic values to the frontier:

- $h_G^O(z, 0) = \frac{3+1}{(2-0)+1} = \frac{4}{3}$
- $h_G^O(g, 0) = \frac{3+0}{(2-0)+1} = 1$

According to Theorem 3, the heuristic search will explore state $(z, 0)$ after the other states currently on the frontier. However, the tie-breaking strategy again selects between states $(g, 0)$ and $(y, 1)$. If it selects $(g, 0)$, then the heuristic search ends finding a path of cost 3 rather than the optimal path above of cost 2. \square

Corollaries 1 and 2 are both important to consider because the computation steps in RaP generally assume that the planners are optimal. It is not a requirement, and

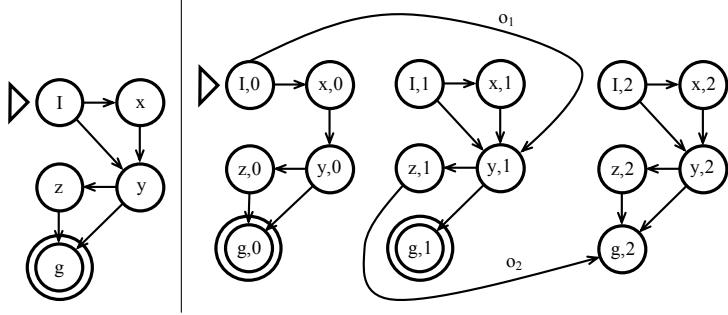


Figure 6.5. An observation space (right, original state space is on the left) that illustrates how to find a non-optimal path from $(I, 0)$ to $(g, i \neq 2)$ first using heuristic h_G^{-O} when h_G is the oracle heuristic.

even the original RaP experiments tried a planner that did not guarantee optimality [227]. However, it is an issue worth considering because it can impact the recognition algorithm's performance. The proofs both show that the order of exploration matter so that Theorems 2 and 3 are more limited than one might expect. However, even without admissibility, techniques such as opticizing search [147] exist that use the plan to refine the search process as it continues and eventually find an optimal solution.

In our early use of the DOPLR heuristics, we erred the interpretation that the theorems applied to the entire observation space rather than just the frontier. Doing so alters one of the corollaries, and we present the following misconception that arises from this misinterpretation to ensure that the reader does not repeat our mistakes.

Misconception 1. *The path found using heuristic search with heuristic h_G^O is optimal if h_G is admissible.*

Proof: (by contradiction) Suppose not. That is, let the path found with heuristic h_G^O to goal state $(s, |\mathcal{O}|) \in \mathcal{S}_\mathcal{O}$ have total cost $f_G(s) > f_G^*(s')$ where $f_G^*(s')$ is the optimal path cost to goal state $(s', |\mathcal{O}|)$. Then because h_G is admissible, $h_G(s) = h_G(s') = 0$ since it cannot overestimate the distance to the goal state. So $f_G(s) = g(I, s)$ and $f_G^*(s') = g(I, s')$, which implies $g(I, s) > g(I, s')$. This means we explored state $(s, |\mathcal{O}|)$ before state $(s', |\mathcal{O}|)$ despite its cost from the initial state being less. By

incorrectly interpreting Theorem 2 and substitution, we also find that heuristic search should have explored $(s', |\mathcal{O}|)$ before $(s, |\mathcal{O}|)$ because

$$g(I, s') = f_G^*(s') < f_G(s) \cdot \frac{|\mathcal{O}| + 1}{|\mathcal{O}| + 1} = f_G(s) = g(I, s).$$

This yields a contradiction where we should have found the optimal path first. \square

6.4.4 Experiments

To evaluate our hypothesis that we can perform probabilistic RaP using heuristic search, we re-implemented a popular PDDL domain from the RaP benchmarks [227] as a heuristic search problem with a customized heuristic for the domain. Besides comparing the outputs between the traditional approach (see Section 3.1.3) and our approaches above, we also compare the performance between our new approaches to investigate their trade-offs.

6.4.4.1 BlockWords Domain Implementation

Section 5.4 describes the BlockWords planning domain for one and two agents; these experiments consider the version with one agent. To encode the states and actions, we used the programming language C++ = \mathbb{L} . We represent stacks of blocks as vectors of characters, and another vector stores all the stacks of blocks to represent the collection of all stacks of blocks on the table. We store the block that the hand holds after a *pick-up* action as another character. In order to parallel the STRIPS-based representation, we define each action as a string for the grounded action name with its parameters and index them in an array. Due to the action containing all the information like its STRIPS counterpart, our observed transitions are simply of the form $(nil, a \in A, nil)$.

However, to take advantage of the customized code that is not run through a generalized, domain-independent classical planner, we *generate children directly from*

the current state. For example, if the hand is empty and there exists a stack with an R block on top of an F block, then we generate a state that moved the R block from that stack to the hand via the action $pick_up(R, F)$. This is a domain-specific approach that is much more computationally efficient than checking whether every pair of blocks satisfies some $pick_up$ or put_down action, taking linear time with respect to the number of stacks $O(s)$ compared to quadratic time with the number of blocks $O(b^2)$ where $s \leq b$ since multiple blocks can be in a single stack.

We also implemented an admissible domain-specific heuristic that accounts for the facts that (1) all the blocks needed to spell a word must be retrieved from each stack, (2) a stack that contains the spelled word must have no irrelevant blocks on top, and (3) $pick_up$ and put_down actions occur in pairs:

$$h_G(s) = \sum_{stack \in s} 2 \cdot remove(stack, G)$$

where $remove(stack, G)$ is the number of blocks on the top of the stack that must be removed. If the bottom of the stack matches the end of G , then $remove(stack, G)$ counts the first block not matching from the end of G and upwards. Otherwise, $remove(stack, G)$ counts the bottom-most block in the stack that G 's spelling needs and upwards.

6.4.4.2 Procedure

We selected twenty BlockWords problem instances from the benchmark dataset that share the same initial state configuration I of eight blocks and hypothesis set \mathcal{H} of twenty-one goal words to possibly spell. Of these twenty problem instances, no more than two of them have the same true hypothesis so that there are a variety of observed plans. Furthermore, there are four sets of five problem instances per number of missing observations: 30%, 50%, 70%, and fully observed. We omitted the five 10% observed problem instances because they contained only one observed action;

our domain-specific implementation of the heuristic search does not contain a way to check for no solution to $\mathcal{P}_{\neg O}^G$ (see Section 6.5.2 for a discussion of this phenomenon) and all these problem instances timed out while exhausting the search space.

We ran RaP on the selected twenty problem instances using our implementations for BOOM and FIREWORKS as well as the DOPLR heuristics. For the comparison with existing probabilistic RaP software, we used Ramírez and Geffner’s code with the FastDownward classical planner [116]. For our methods, we logged the computational resources (nodes explored, nodes expanded, and time) as well the distribution over the hypotheses. For the existing methods, we logged only the distribution over the hypotheses because the differences in state representation and implementation do not permit a fair comparison of resource consumption.

6.4.4.3 Results

Out of the twenty problem instances, both approaches successfully ranked the correct hypothesis as the most likely in sixteen instances. However, the instances in which they were each correct were not the same. In comparison, the original probabilistic recognition as planning algorithm ranked the correct hypothesis as most likely in all twenty problem instances. Because our heuristics are not guaranteed to be optimal except for BOOM, we hypothesize that these instances’ computations involved non-optimal plan costs. To investigate the differences in the distributions, we computed the KL Divergence

$$D_{KL}(RG \parallel \cdot) = \sum_{G \in \mathcal{G}} P_{RG}(G | \mathcal{O}) \cdot \ln \left(\frac{P_{RG}(G | \mathcal{O})}{P_{\cdot}(G | \mathcal{O})} \right)$$

where \cdot is one of the proposed heuristic search approaches and RG is the traditional version that uses a classical planner. The KL Divergence indicates the “surprise factor” if \cdot ’s distribution replaced RG ’s distribution, noting to what extent the distributions differ as one observes events sampled from each. Figure 6.6 shows that both

Table 6.1. Resources Consumed (BOOM and FIREWORKS)

Resource	Min	Q ₁	Median	Q ₃	Max	Mean
Nodes Explored	1321	2278.25	3574	4352	7944523	402780.5
Nodes Generated	5939	8893.25	12731.5	15586.25	7956346	414987.5
Time (seconds)	0.4601	0.6566	0.8450	1.1480	1114.7107	57.0987
Accuracy	16 / 20					

Table 6.2. Resources Consumed (DOPLR Heuristics)

Resource	Min	Q ₁	Median	Q ₃	Max	Mean
Nodes Explored	2491	7516.25	12661	24650	7961566	417514.1
Nodes Generated	10439	27900	38398.5	70550.75	7998340	452375.95
Time (seconds)	0.8777	2.4987	3.9017	6.7297	1123.9013	61.8347
Accuracy	16 / 20					

approaches' distributions are quite similar except for the final problem instances (the fully observed cases) and problem instance 7.

Tables 6.1 and 6.2 present the five-statistical summary and mean of resource consumption for the proposed approaches over the twenty problem instances. The maximum value is an outlier on the same problem instance, but the BOOM and FIREWORKS method generally uses less resources than the DOPLR heuristics. This provides evidence that the number of resources is reduced when we use a less thorough heuristic. However, there seems to be *no trade-off between extra computational resources and accuracy* because their performances are almost identical. Despite this, they both produce *distributions that often make similar predictions* to traditional RaP, confirming our primary hypothesis.

6.5 Algorithms for Faster Recognition as Planning

Real-time applications present constraints on a resource that is more difficult to work around than others: *time*. Memory hardware has gradually decreased in size and cost to allow reasonable scale-up, but people do not have the ability to manipulate the passage of time outside of developing hardware with more clock cycles and employing multiple cores or GPUs to take advantage of parallel computation. Even with greater-

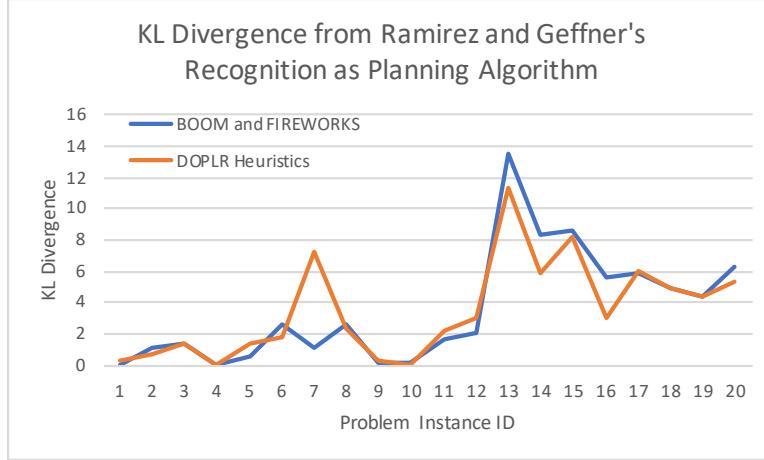


Figure 6.6. KL Divergence comparing the distributions from our approaches to the original.

frequency processing units, the rate of increased speed-up is decreasing due to physical limitations. Likewise, the flow of information in an algorithm can restrict how much of the procedure is parallelizable, and the scale-up is still restricted to the number of processing units available (compared to a nondeterministic Turing Machine that can run infinitely many tapes in parallel).

Many algorithms, including the ones for artificial intelligence, have both off-line and on-line variants to address this concern. The on-line versions usually sacrifice some quality or metric, such as performance or soundness, in exchange for having some solution within the specified time limit. Automated planning began to address this a while ago, beginning with anytime algorithms [300] that gradually improved the plan’s cost if more time was available after finding an original plan. On the other hand, the younger, inverse research area of plan, activity, and intent recognition has only begun to investigate this challenge for some of its currently employed algorithms.

RaP algorithms have a tendency to be relatively slow and run off-line with as many observations as possible because they require multiple calls to the off-the-shelf planners. As such, there has been research on approaches for speeding up the original algorithm [189, 281, 68, 217]; we reviewed these algorithms in Section 3.3.2.

Though these methods do improve the runtime of RaP algorithms, the majority of them employ some assumptions that lead to other potential complications during on-line performance. This effectively makes them *faster off-line algorithms unless the application uses the same assumptions in its real-time setting*.

In contrast to these speed-up approaches that apply assumptions to create algorithmic variations or take advantage of some trick, we propose a simple approach that takes advantage of a special subset of off-the-shelf planners, which still performs the original RaP algorithm. Namely, we remove overhead and redundant operations between some consecutive planner calls through *continuing to find solutions with a single planner call*. Shirin Sohrabi and Michael Katz together identified a very similar approach independently of Richard G. Freedman, the author of this dissertation, around the same time. They are now working together to formally disseminate the results, but Sohrabi and Katz kindly granted Freedman permission to include his original findings before the collaboration in this dissertation.

Before revealing these findings in Section 6.5.2, our implementation builds on earlier attempts to speed-up RaP using an algorithm that seems to have become forgotten in the literature: multiple goal heuristic search [59]. Roman Ganchin and Yi Ren Fung worked under Freedman on this earlier attempt to revive the algorithm³ and use it for faster RaP. Section 6.5.1 explains the algorithm and our empirical observations, including realizations of when it is better to use multiple goal heuristic search and when it is better to avoid using it.

6.5.1 Multiple Goal Heuristic Search

When running off-the-shelf automated planning software for RaP algorithms, it is important to note that *only the task's goal conditions change* between planner

³We found out that the original implementation was lost following the unfortunate death of Dmitry Davidov, the lead author of the multiple goal heuristic search paper.

executions. The initial state, set of actions, and set of states are only modified once from the original planning domain, and they do not change between iterations. This makes sense because RaP simulates the agent performing *each task* within the *same world* while enforcing the *same observed actions*. We hypothesize that this leads to redundant computation because independent searches can explore the same state multiple times.

We further hypothesize that multiple goal heuristic search (MGHS), an algorithm for web crawling applications that finds as many relevant websites as possible for a given search query, can help with this issue [59]. Although the concept of continuing the search process after exploring any goal state (a condition that terminates many traditional search algorithms) sounds trivial, MGHS focuses on developing new heuristics that better guide the search process to *find all the goals more efficiently*.

The MGHS heuristic has many responsibilities, including guiding the search progress in multiple directions (per goal) at once, choosing to explore nodes that make progress towards larger collections of goals, and handling multiple goal states that satisfy the same task conditions (biasing the search progress away from goal states for unsolved tasks' conditions).

Davidov and Markovitch discuss several theoretical MGHS heuristics that describe the perfect search cases, but these heuristics are usually impossible to find in reality without already exploring the entire search space. However, they use those properties to propose several heuristics for one who already has a distance approximation function $h_{dist}(s, g)$ for traditional heuristic search, similar to h in the remainder of the dissertation except that the goal conditions are now an additional parameter:

Sum considers how close a state is to finding each goal state $g \in S_G$ in the search space

$$h_{sum}(s) = \sum_{g \in S_G} h_{dist}(s, g)$$

where $S_G = \{s \in S, g \in G \mid s \text{ satisfies } g\}$ is the set of all goal states. S_G 's definition makes more sense for web crawling domains because each state is a web page over a network, which is likely smaller than the set of possible states in the reachable automated planning state space (up to $2^{|F|}$ for STRIPS-like state representations). We thus consider S_G to be the set of goal condition sets instead.

Progress accounts for the search progress to avoid rapidly switching its focus between different ‘directions’ (clusters of goal states), which allows the search progress to focus its limited resources on one cluster at a time. The progress heuristic first evaluates each state $s \in S$ on the frontier with respect to $G_P(s)$, the set of goal states that are closest to it compared to other states on the frontier:

$$G_P(s) = \left\{ g \in S_G \mid h_{dist}(s, g) = \min_{s' \in frontier} h_{dist}(s', g) \right\}.$$

Then the progress heuristic computes the ratio of

$$D_P(s) = |G_P(s)|^{-1} \cdot \sum_{g \in G_P(s)} h_{dist}(s, g),$$

the average distance to the elements of $G_P(s)$ (preferring to find the closest cluster of goal states), to the actual number of goals (preferring to find as many goal states as possible in larger clusters):

$$h_{progress}(s) = \frac{D_P(s)}{|G_P(s)|}.$$

If $G_P(s) = \emptyset$, then $h_{progress}(s) = \infty$. Unlike traditional heuristic search, this requires *updating every state's heuristic value* whenever the frontier changes.

Marginal Utility is a resource-bounded heuristic that tries to optimize the search tree to find as many goal states as possible with the remaining r resources (nodes explored, time, etc.). Although the marginal utility heuristic involves an algorithm that frequently updates tables for every state, which estimates the value of the theoretical heuristic ratio

$$MU(s, r) = \max_{T \in T(s, r)} \frac{|T_g(s) \cap T|}{r},$$

it performs best when applying many modifications to the search process such as clustering similar states. These modifications require learning features that indicate which states are similar to each other in the search space. Conceptually, $T(s, r)$ is the set of search trees with root node s that spent r resources on the search process, and $T_g(s)$ is the set of all goal states in S_G that s can reach in the search space. Thus $MU(s, r)$ compares the ratio of retrievable goal states to the number of resources that must be consumed. This is similar to the progress heuristic, but it does not use the distance estimate and allocates how many resources to spend on progress before switching to another ‘direction’.

A single search with MGHS might avoid the redundancy of RaP computation when $S_G = \{G + O, G + \overline{O} | G \in \mathcal{H}\}$. If the domain’s representation happens to be STRIPS-like, then there are plenty of domain-independent heuristics to consider for h_{dist} that are currently implemented in off-the-shelf planning software [123, 116]. Off-the-shelf classical planners will account for the adherence (or lack thereof) to \mathcal{O} due to its compilation into the state space’s augmented fluent set $F_{\mathcal{O}}$. Otherwise, the developer needs to define observation matching (see Section 6.3.3) and computing the distance between two states or a state and a set of goal conditions (see Section 6.3.2).

MGHS cannot re-explore the same states that individual RaP searches repeatedly explore, but MGHS should ideally explore only states that at least one of those

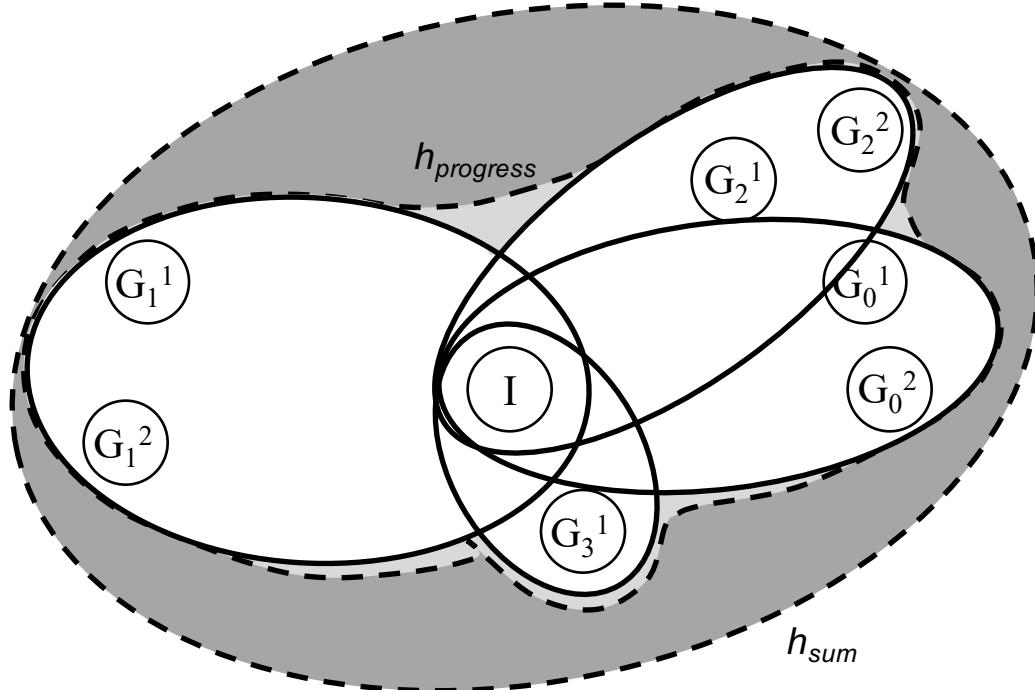


Figure 6.7. An illustration of MGHS’s search progression (dotted borders) with the sum and progress heuristics compared to individual heuristic searches per goal condition set G_i (solid borders). Overlapping solid-bordered regions indicate redundant exploration that MGHS saves. MGHS unnecessarily explores regions exclusively within the dotted border and not within a solid-bordered region (shaded).

searches would explore. If the tasks’ goal conditions correspond to states within a local region of the state space (that is, each $G \in \mathcal{H}$ is similar enough that $h_{dist}(g_1, g_2) \leq k$ for all $g_1, g_2 \in S_G$ and some relatively small $k \in \mathbb{R}^{\geq 0}$), then this is not a large concern because many states in each individual search overlap. However, tasks with more diversity might have fewer overlapping states in their individual searches. This would be the same as performing each RaP search, but there is a chance that MGHS will explore additional states in regions of the search space between tasks that are never explored during any individual search—Figure 6.7 illustrates a layout of search progression for these cases. We *hypothesize that the progress heuristic will help with this* because it focuses on sets of goal states to minimize wandering between multiple tasks.

Representing tasks as goal conditions introduces a potential side effect that Davidov and Markovitch discuss as an additional consideration (Section 4.5 of their work [59]). It is usually the case that multiple goal states satisfy a single set of goal conditions, which means that MGHS *still factors alternative goal states for a solved task into heuristic calculations*. Because we only need one pair of plans per goal for RaP’s likelihood approximation⁴, spending resources on additional plans for a solved task should not be a priority.

Although we can simply remove the solved task’s goal states from S_G and stop collecting alternative plans, this might reduce RaP’s performance because the *MGHS heuristics are not guaranteed to be admissible* like h_{dist} often is for traditional heuristic search. Coincidentally finding additional plans during search is beneficial if they happen to have a lower cost than the ones currently found. RaP’s equations prefer optimality when approximating the likelihood, but we repeat from Section 6.4.4.3 that experiments with a *satisficing* (finding any solution, not necessarily optimal) planner still showed reasonable performance [227]. The satisficing planner was faster than the optimal one, and MGHS’s anytime design is conveniently satisficing.

To postpone S_G ’s refinement until more tasks have at least one solution, we use Davidov and Markovitch’s heuristic modification suggestions. The modification considers the distance between a state and the set of already-found goal states

$$h_{dist}^{MGHS}(s, g) = h_{dist}(s, g) \left(1 + c_1 e^{-c_2 d(s, \text{found}(g))}\right)$$

with parameter scalars $c_1, c_2 \in \mathbb{R}^{\geq 0}$. We instead count the *number of goal states found for the specified task*

$$h_{dist}^{RaP}(s, g) = h_{dist}(s, g) \cdot e^{|\text{found}(g)|}.$$

⁴Even if we chose to violate the single-plan/max-replaces-sum assumption, we can recover multiple plans from a single goal state if we keep track of *all states’ parents* while they are on the frontier.

Table 6.3. Features of MGHS Variations

Variation	Heuristic	h_{dist}	Weighted	Update Per Goal
SUM_A*_UNW	sum	A*	X	X
SUM_A*_W_NU	sum	A*	✓	X
SUM_A*_W_U	sum	A*	✓	✓
PROG_A*_UNW	progress	A*	X	✓
PROG_A*_W	progress	A*	✓	✓
SUM_BF_UNW	sum	best-first	X	X
SUM_BF_W_NU	sum	best-first	✓	X
SUM_BF_W_U	sum	best-first	✓	✓
PROG_BF_UNW	progress	best-first	X	✓
PROG_BF_W	progress	best-first	✓	✓

The motivation for this difference is that web crawling aims to find a diverse set of relevant websites and *each site is still unique*, but diverse goal states for a *single task are redundant for RaP computations*. Thus searching for goals that satisfy a specific task are less favorable when other tasks have not found as many goal state instances, but the priority increases as other tasks find more respective goal state instances.

6.5.1.1 Empirical Evaluation

We implemented MGHS using both the sum and progress heuristic, including a few basic variations. Table 6.3 lists the features distinguishing each variation. The primary classes of features involve defining h_{dist} with respect to A* search ($g(s) + h(s)$) or best-first search ($h(s)$), weighting h_{dist} based on the number of found goals (h_{dist}^{RaP}), and updating the computation of h_{dist}^{RaP} for all nodes on the frontier whenever MGHS finds a new goal. Because the progress heuristic requires frequent updates based on the states in the frontier, we note that these variations update whenever MGHS finds a new goal by definition.

For our empirical evaluation, we implemented RaP where a single iteration of some variation of MGHS retrieved plans solving all the pairs of goal condition sets: $S_G = \{G + O, G + \overline{O} | G \in \mathcal{H}\}$. The current implementation instructs MGHS to terminate when either:

- A plan exists per set of goal conditions or
- The entire search space is exhausted.

Because the search space is often far too large to feasibly explore, we include an optional ‘give-up’ threshold that throws away all states $s \in S$ whose current plan’s cost $g(s)$ exceed a scalar multiple k of the expected optimal plan’s cost:

If $\left(\max_{g \in S_G} h_{dist}(I, g) \right) \cdot k < g(s)$, then do not add s to the frontier.

This reduces the size of the search space for *quicker exhaustion at the risk of missing a solution when the heuristic is inaccurate*. MGHS returns the most optimal plan found per goal condition set when it meets a termination condition, and we assume that all goal condition sets without a plan (the second termination case) have no solution. Theorem 6 (not introduced until Section 6.5.2) provides a weak justification for the assumption in this case because MGHS does not guarantee that the most optimal plan it finds is truly the most optimal solution to the problem.

We ran this RaP implementation and two versions using single-goal searches (with A* and best-first for h_{dist}) on the same BlockWords domain and suite of problem instances from Section 6.4.4’s experiments. Rather than use the BOOM and FIRE-WORKS or DOPLR heuristics from those experiments, we used the original heuristic function as an experimental control—Section 6.5.2.1 provides evidence that this heuristic is useful despite ignoring the observation simulation progress. Tables 6.4 through 6.9 include the results with ‘give-up’ threshold $k = 2$ to compare the variations with respect to their resource consumption. The tables do not include any variations with the “update per goal” feature because they took more than six hours to run per problem instance. Although it might be a consequence of a poor implementation without much optimization, the time to update all the states on the frontier’s heuristic values far outweighed the time exploring unnecessary states with the other

variations' less accurate heuristics. To get a better sense of this scaling trade-off between runtime and visited states, Table 6.10 presents the resource consumption when searching over smaller, more trivial BlockWords problem instances.

Regarding the empirical results, the recognition accuracy is generally reasonable for all the variations. However, it is greater when using A*'s notion of distance to consider the current plan cost as much as the estimated remaining plan cost, which is an issue that Davidov and Markovitch also mentioned being worth future consideration. Amongst the MGHS variations that do not update per goal, we notice that weighting the heuristic upon finding goals generally improves resource consumption both in the number of visited nodes and runtime. This indicates that MGHS with the sum heuristic was finding *multiple solutions to the same goal condition set in a nearby region of the search space*. Increasing the weight of the newly expanded states' heuristic value (compared to updating those already on the frontier) after finding the first such goal dissuades MGHS from continuing to investigate nearby solutions that are likely redundant for RaP computation. This empirical result supports Davidov and Markovitch's conjecture and their proposed solution.

Regardless of the resource savings from applying weights to the sum heuristic, it is still evident that *running RaP with multiple single searches is far more efficient than running RaP with a single MGHS*. While this appears to be a dead-end direction and reason to abandon MGHS, we revisit Figure 6.7's conceptual illustration of search progression. Specifically, the sum heuristic searches the largest region of the state space in order to *find at least one state that satisfies each set of goal conditions*. The sum heuristic considers *each goal state's distance with respect to each unique goal condition set*:

$$h_{sum}(G_i^j) = \sum_{g \in S_G} h_{dist}(G_i^j, g),$$

which appears to contribute to this issue. For example, $h_{sum}(G_1^1)$ includes the summand $h_{dist}(G_1^1, G_0)$, which are on opposite ends of the search space from the initial

Table 6.4. Resources Consumed (RaP with Single-Goal A*)

Resource	Min	Q ₁	Median	Q ₃	Max	Mean
Nodes Explored	7102	88119	324233	579397	1283969	424814
Nodes Generated	22962	194414	618263	838837	1429749	636698
Time (seconds)	2.1282	25.3894	91.0205	144.8106	313.1597	110.8889
Accuracy	20 / 20					

Table 6.5. Resources Consumed (RaP with MGHS SUM_A*_UNW)

Resource	Min	Q ₁	Median	Q ₃	Max	Mean
Nodes Explored	657720	988145	1168908	1505058	2801828	1291983
Nodes Generated	940983	1291734	1484203	1700347	2758316	1544703
Time (seconds)	331.1712	503.1685	572.4558	713.0611	1153.2303	627.3257
Accuracy	17 / 20					

Table 6.6. Resources Consumed (RaP with MGHS SUM_A*_W_NU)

Resource	Min	Q ₁	Median	Q ₃	Max	Mean
Nodes Explored	267597	396689	458022	1415841	2231816	878432
Nodes Generated	523910	649228	751477	1258422	1866089	962897
Time (seconds)	201.7343	249.0654	282.8890	615.0121	1052.3603	461.7122
Accuracy	16 / 20					

Table 6.7. Resources Consumed (RaP with Single-Goal Best-First)

Resource	Min	Q ₁	Median	Q ₃	Max	Mean
Nodes Explored	19793	129884	222656	417098	795702	290038
Nodes Generated	22001	119523	183876	360523	616948	232589
Time (seconds)	5.0836	34.2357	59.7817	109.2426	195.2136	73.6835
Accuracy	14 / 20					

Table 6.8. Resources Consumed (RaP with MGHS SUM_BF_UNW)

Resource	Min	Q ₁	Median	Q ₃	Max	Mean
Nodes Explored	547175	913979	1066322	1410752	2260153	1160935
Nodes Generated	653779	1109590	1281681	1360916	2156495	1276209
Time (seconds)	306.1243	472.4070	534.0588	660.0212	954.9943	560.9642
Accuracy	15 / 20					

Table 6.9. Resources Consumed (RaP with MGHS SUM_BF_W_NU)

Resource	Min	Q ₁	Median	Q ₃	Max	Mean
Nodes Explored	124033	336739	650518	1125279	2086922	783285
Nodes Generated	255600	594422	853630	971261	1720481	849092
Time (seconds)	95.9842	225.1612	391.5334	557.2130	1009.3003	419.8096
Accuracy	13 / 20					

Table 6.10. Resources Consumed (5-Block→6-Block Domain)

Variation	Nodes Explored	Nodes Generated	Time (seconds)
Single-Goal A*	393→619	890→1782	0.0490→0.1413
SUM_A*_UNW	384→1372	656→2684	0.0855→0.3414
SUM_A*_W_NU	617→1660	873→3044	0.1014→0.4436
SUM_A*_W_U	900→4906	1074→5594	0.5972→14.5931
PROG_A*_UNW	576→1276	645→1565	1.5588→10.0335
PROG_A*_W	516→1274	600→1563	1.3531→10.0948
Single-Goal BF	323→353	558→743	0.0427→0.0513
SUM_BF_UNW	352→1177	602→2178	0.0709→0.2939
SUM_BF_W_NU	820→2218	919→3224	0.1537→0.4846
SUM_BF_W_U	805→3527	892→3576	0.4350→7.5354
PROG_BF_UNW	348→1546	475→2057	0.6719→15.0347
PROG_BF_W	375→3740	523→3238	0.7824→75.8171

state. Because MGHS still explores the frontier in order of least heuristic value to greatest, this means that it must explore all states whose heuristic values are less than such large summations (or averages because they multiply the sum by a scalar constant, which is the number of goal condition sets).

MGHS does not encounter this issue in web crawling applications because the *set of goal criteria are disjunctive*; any websites that match at least one goal condition set is acceptable and satisfies the search results even if none of the found websites match some specified goal condition set. On the other hand, the *set of goal criteria are conjunctive* for RaP; the search results remain unsatisfied until MGHS finds at least one goal state per goal condition set. This means that we must *take caution when assigning multiple goals to the same MGHS*. If the goal states for different goal condition sets happen to be nearby in the state space, then the sum heuristic between those goals will remain relatively small and not extend the search progression into unnecessary regions.

While we leave extensive work on *defining the proximity of goal conditions* to future work, we note one obvious case where two goal condition sets are clearly similar to each other: $G + O$ and $G + \overline{O}$ for some $G \in \mathcal{H}$. We provide a deeper

Table 6.11. Resources Consumed (RaP with MGHS SUM_A*_UNW, Goal Pairs)

Resource	Min	Q ₁	Median	Q ₃	Max	Mean
Nodes Explored	21024	95067	236797	416710	1020378	310706
Nodes Generated	46667	163212	375175	519549	1340438	413242
Time (seconds)	6.0025	29.3887	72.4838	123.7634	283.3005	90.0523
Accuracy	17 / 20					

Table 6.12. Resources Consumed (RaP with MGHS SUM_A*_W_NU, Goal Pairs)

Resource	Min	Q ₁	Median	Q ₃	Max	Mean
Nodes Explored	39295	307727	438189	942138	2266736	669430
Nodes Generated	69837	351367	484399	950422	2207108	698362
Time (seconds)	11.0762	86.1807	121.6704	247.7545	544.1348	175.1491
Accuracy	17 / 20					

theoretical justification for this in Section 6.5.2 below, and Tables 6.11 through 6.14 show the resource consumption in the previous experiments when we reallocate the goal condition sets into pairs for $|\mathcal{H}|$ MGHS searches instead of one. These empirical results present an interesting trend that the weighted sum heuristic now performs less efficiently than the unweighted sum heuristic. However, if we consider that the set of goal condition sets in each MGHS are much closer together than before, then this reversal makes sense. The incentive to abandon the current portion of the search space is less ideal when *we can guarantee that goal states that satisfy alternative goal condition sets are nearby*. With this in mind, we point out in a comparison between Table 6.4 and Table 6.11 that RaP with MGHS using the unweighted sum A* heuristic, given the goal pairings rather than all the goals, is *generally faster and visits fewer states* than RaP with single-goal A* searches. This provides evidence supporting our hypothesis, and *better allocating nearby goal condition sets into fewer MGHS searches will likely improve this performance further*.

6.5.2 Continuing the Search Process

As a consensus with the current speed-up techniques in Section 3.3.2, our approach also identifies that the overhead for augmenting the domain to create two new

Table 6.13. Resources Consumed (RaP with MGHS SUM_BF_UNW, Goal Pairs)

Resource	Min	Q ₁	Median	Q ₃	Max	Mean
Nodes Explored	23141	149044	241952	469283	1018513	313381
Nodes Generated	23898	122024	192020	347512	766294	247756
Time (seconds)	5.9818	41.4213	68.0177	128.4404	255.8661	84.0761
Accuracy	16 / 20					

Table 6.14. Resources Consumed (RaP with MGHS SUM_BF_W_NU, Goal Pairs)

Resource	Min	Q ₁	Median	Q ₃	Max	Mean
Nodes Explored	38294	395098	707105	1414980	2169304	890261
Nodes Generated	60729	408550	760000	1404691	2024358	902211
Time (seconds)	11.2258	113.4534	192.2762	331.8237	521.5126	223.8063
Accuracy	16 / 20					

planning problems is not necessary to perform RaP. As mentioned at the start of Section 6.4, solving the original planning problem $\mathcal{P}^G = (F, A, I, G)$ inherently solves one of RaP’s two new problems \mathcal{P}_O^G or $\mathcal{P}_{\neg O}^G$, but which one is arbitrary.

Masters and Sardina’s first assumption that it will often solve $\mathcal{P}_{\neg O}^G$ [189] makes sense when *we can permute many actions in the plan and still have a solution or there are multiple distinct ways to accomplish the task*, but this does not always hold—the problem instance’s initial state and goal conditions play a large role in the number of optimal plans and their prefix diversity. As a simple example, let us revisit the GridWorld example in Corollary 1’s proof. There is a reason that we did not reuse this example to prove Corollary 2⁵. Figure 6.8 slightly differs from Figure 6.4 because the goal state for $\mathcal{P}_{\neg O}^G$ cannot have the single observation *up* satisfied. However, *all solutions to the automated planning problem must contain a single up action*. The same issue thus occurs when O contains one or two *left* actions. Although these assumption-violating cases seem obscure because O is so short, this is an extremely relevant concern at the *beginning of an interactive experience when R_{ing} just begins observing R_{ed} and needs to calculate initial responses*.

⁵It is easier for the reader to follow examples that are consistent, which is why we frequently use BlockWords throughout this dissertation whenever possible.

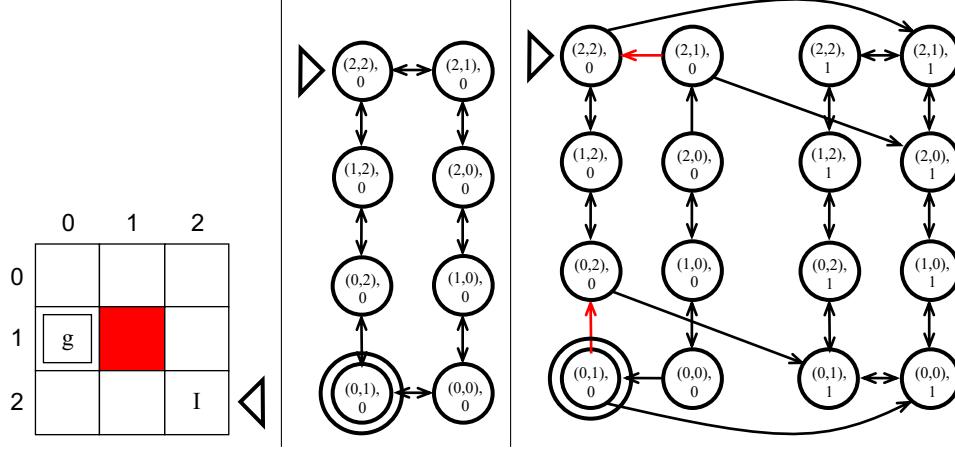


Figure 6.8. An observation space (right, original state space is in the center based on the grid on the left) that illustrates the lack of any solution from $(I, 0)$ to $(g, 0)$. The red edges in the observation space show the bidirectional edges in the state space that would transition to the goal state if the *up* action was not observed.

We similarly consider a BlockWords domain where every block is on the table and has a unique letter. Because nothing is stacked, there is *exactly one optimal solution*: pick up and put down each block sequentially to build a stack that spells the goal word. If R_{ed} solves the problem optimally, then $\mathcal{P}_{\neg O}^G$ has no solution for all possible O that R_{ing} could observe (accounting for missing observations). On the other hand, if we have an initial state where

1. A block b_1 's letter is not part of the goal word,
2. b_1 is on top of a block b_2 whose letter is part of the goal word,
3. There is some block b_3 on the table whose letter is also not part of the goal word, and
4. b_1 and b_3 are both the top-most blocks of their stacks,

then we can trivially branch the observation sequence after picking up b_1 , placing it either on the table or on top of b_3 . If there are more stacks like the one that step 3. describes, then there are additional optimal plans that place b_1 on those stacks.

Furthermore, additional stacks like the one that step 2. describes allow even more plan flexibility because the first action can pick up either stack’s top-most block.

These plans are not necessarily diverse, but the fact that they are all so similar indicates that they involve similar regions of the state space. That is, *the computational effort to find another plan after the first one should be far less than the computational effort required to start yet another search from scratch*. There are many trivial and off-the-shelf ways to collect these plans:

- Perform breadth-first or A* search up to some stopping condition (resource-bound, number of goals found, etc.), storing the path to each goal state found.
- Off-the-shelf software for diverse planning [208] can find sets of unique plans for a single problem.
- Off-the-shelf software solving the top- k -shortest paths problem [6] find the k -most optimal plans.
- Off-the-shelf software solving the top- k -quality paths problem [148] find all the plans with cost at most k .

Each plan π that this extended search finds belongs to the set Π_G , which means π also belongs exclusively to one of its partitioned subsets Π_{G+O} or $\Pi_{G+\bar{O}}$. As we continue to collect more plans, the likelihood that there is at least one plan in each partition increases—the computations for RaP’s distribution over the hypotheses use *the most optimal plan in each partition*. This continued search is faster than running the planner over the pair of modified problems as long as its consumed resources do not exceed the resources that two slightly different search problems consume (on average, the bottleneck of search occurs near the optimal cost due to the exponential branching factor [12]). Because most off-the-shelf planners use domain-independent heuristics, we point out that the runtimes and heuristic functions between the above approaches vary.

Theorem 4. If k satisfies

$$k > \max_{G \in \mathcal{G}} \max(|U_G|, |V_G|)$$

such that

$$U_G = \left\{ \pi \in \Pi_{G+O} \mid \text{cost}_\pi \leq \text{cost}_{\pi_{G+\bar{O}}^*} \right\}$$

and

$$V_G = \left\{ \pi \in \Pi_{G+\bar{O}} \mid \text{cost}_\pi \leq \text{cost}_{\pi_{G+O}^*} \right\},$$

then the solution to the top- k -shortest paths problem contains the plans that return the same probability distribution as the original probabilistic RaP.

Proof: For each $G \in \mathcal{G}$, let $x_G = \max(\text{cost}_{\pi_{G+O}^*}, \text{cost}_{\pi_{G+\bar{O}}^*})$ be the cost of the less optimal of the optimal solutions to the pair of new planning problems in RaP. By setting k to a value strictly greater than the larger of U_G and V_G for each $G \in \mathcal{G}$, we are guaranteed to find at least one plan with cost x_G in each partition Π_{G+O} and $\Pi_{G+\bar{O}}$. Furthermore, because k is large enough to include all plans with cost x_G , we are guaranteed that there does not exist a more optimal plan in $\Pi_G - (U_G \cup V_G)$ waiting to be found in the search space. This means that the search results contain an optimal solution for both \mathcal{P}_O^G and $\mathcal{P}_{-\bar{O}}^G$, whose costs are the values for the original probabilistic RaP computations. \square

Theorem 5. If k satisfies

$$k \geq \max_{G \in \mathcal{G}} \max \left(\text{cost}_{\pi_{G+O}^*}, \text{cost}_{\pi_{G+\bar{O}}^*} \right),$$

then the solution to the top- k -quality paths problem contains the plans that return the same probability distribution as the original probabilistic RaP.

Although Theorems 4 and 5 both guarantee computing the same distributions as the original probabilistic RaP, the challenge is identifying the correct value of k . Precomputing k is not ideal because it requires finding the plans that effectively perform the original probabilistic RaP algorithm. Likewise, assuming a larger value than necessary for k is risky because it could waste a lot of resources as the search continues to find plans that are not involved in the probabilistic RaP computation. If these extraneous resources are too great (that is, the extended search explores more states than two vanilla searches starting from the initial state), then it will also take longer than the original algorithm.

There are two possible approaches that address this issue of not knowing the correct value of k . The brute-force approach is to modify such planners with an *alternative stopping criteria* that terminates when the collection of found plans contains one for solving each of \mathcal{P}_O^G and $\mathcal{P}_{\neg O}^G$. Without modifying a planner as extensively, the second approach applies a *bound of allowable error*. In particular, continuing to search with any of the non-diverse planning approaches proposed above collects the plans in order of increasing cost (though breadth-first search and A* might miss some if using graph search rather than tree search). If the extended search finds all the plans up to some cost \bar{c} , then the remaining plans clearly have a cost greater than \bar{c} . This fact allows us to conclude the error of stopping probabilistic RaP prematurely under the *assumption that the complementary RaP problem has no solution*.

Lemma 4. *If there does not exist any solution with cost \bar{c} or less to \mathcal{P}_O^G , then*

$$P(\mathcal{O}|G) \leq e^{-\beta\bar{c}}.$$

If there does not exist any solution with cost \bar{c} or less to $\mathcal{P}_{\neg O}^G$, then

$$P(\neg\mathcal{O}|G) \leq e^{-\beta\bar{c}}.$$

 **Theorem 6.** If the optimal solution to either \mathcal{P}_O^G or $\mathcal{P}_{\neg O}^G$ has cost $cost_{\pi^*}$ and there does not exist any solution with cost $cost_{\pi^*} + d$ or less for some $d \in \mathbb{R}^{\geq 0}$ to the other problem, then the error of assuming that there is no solution to the other problem is at most

$$1 - \frac{1}{1 + e^{-\beta d}} \text{ for } \max(P(\mathcal{O}|G), P(\neg\mathcal{O}|G))$$

$$\text{and } \frac{e^{-\beta d}}{1 + e^{-\beta d}} \text{ for } \min(P(\mathcal{O}|G), P(\neg\mathcal{O}|G)).$$

Proof: A problem with no solution's optimal solution has cost ∞ . Thus when we know that there is a solution to one of \mathcal{P}_O^G or $\mathcal{P}_{\neg O}^G$ and assume that there is no solution to the other, we are effectively assuming that

$$\max(P(\mathcal{O}|G), P(\neg\mathcal{O}|G)) = 1$$

$$\text{and } \min(P(\mathcal{O}|G), P(\neg\mathcal{O}|G)) = 0$$

because $\lim_{x \rightarrow \infty} e^{-\beta x} = 0$, which is used as substitution for its respective probabilities in Equation 3.1. However, our assumption is wrong if there is a solution to the other problem with cost greater than what we explored in the search space. We bound our error using Lemma 4 for substitution into Equation 3.1:

$$\begin{aligned} \max(P(\mathcal{O}|G), P(\neg\mathcal{O}|G)) &\geq \frac{e^{-\beta cost_{\pi^*}}}{e^{-\beta cost_{\pi^*}} + e^{-\beta(cost_{\pi^*} + d)}} \\ &= \frac{e^{-\beta cost_{\pi^*}}}{e^{-\beta cost_{\pi^*}} (1 + e^{-\beta d})} = \frac{1}{1 + e^{-\beta d}} \text{ and} \\ \min(P(\mathcal{O}|G), P(\neg\mathcal{O}|G)) &\leq \frac{e^{-\beta(cost_{\pi^*} + d)}}{e^{-\beta cost_{\pi^*}} + e^{-\beta(cost_{\pi^*} + d)}} \\ &= \frac{e^{-\beta cost_{\pi^*}} \cdot e^{-\beta d}}{e^{-\beta cost_{\pi^*}} (1 + e^{-\beta d})} = \frac{e^{-\beta d}}{1 + e^{-\beta d}}. \quad \square \end{aligned}$$

6.5.2.1 Variations of Implementation

The insights above explain the motivations for our proposed approach to do the search on-line with similar results to the original probabilistic RaP algorithm, using only $|\mathcal{H}|$ planner calls rather than the traditional overhead and $2|\mathcal{H}|$ planner calls. Though this is the goal of our method, there are cases where we can speed the approach up further. This is analogous to how other state-of-the-art approaches described in Section 3.3.2 apply if their assumptions are appropriate for the situation.

The first variation performs off-line bootstrapping for faster on-line use. When the set of hypothesized goal states \mathcal{H} is *known before the observing agent begins to perform recognition*, then we can precompute a large set of plans that solve each $G \in \mathcal{H}$. Rather than waste resources, our $|\mathcal{H}|$ searches can still perform the off-line precomputation more efficiently. If we ensure that the set of plans Π_G contains all plans up to cost \bar{c} , then the intuition of our approach still applies without needing to do the search component on-line. Instead, we only need to scan Π_G on-line in order of increasing cost to identify the most optimal plan solving each of \mathcal{P}_O^G and $\mathcal{P}_{\neg O}^G$ as R_{ing} observes R_{ed} in real time. As long as the least-optimal plan in Π_G is sufficiently large, then Theorem 6 guarantees that the error is acceptably small when all the plans in Π_G only satisfy one of the two problems. The worst-case time complexity involves scanning all the plans in their entirety because none of the plans satisfy \mathcal{P}_O^G , which is $O(|\Pi_G| \cdot \max_{\pi \in \Pi_G} \text{cost}_\pi)$.

To avoid the linear scan of each plan on-line, regardless of whether we collect Π_G on-line or off-line, we can take advantage of the original RaP algorithm's augmented versions of the state space F_O and A_O . That is, we *search in the observation space rather than the original state space*. However, instead of separate searches to solve each of \mathcal{P}_O^G and $\mathcal{P}_{\neg O}^G$ in this altered state space, we perform our approach's ongoing single search to solve for the original goal without the augmented condition of (not) satisfying O . Because the augmented states include the progress through simulating

the observation sequence, we only need to store each goal state that is found rather than the entire plan reaching it. The check for each augmented goal state g is trivial: if $p_m \in g$, then g 's plan solves \mathcal{P}_O^G ; if $p_m \notin g$, then g 's plan solves $\mathcal{P}_{\neg O}^G$. Thus, the scan over found solutions is linear in the worst case that all the goal states only satisfy one of the two problems, which is $O(|\Pi_G|)$. This variation of our proposed approach assumes that the *overhead computation for the generation of the augmented state space does not outweigh the computation saved from the reduced plan scanning* because F_O and A_O are updated on-line as \mathcal{O} increases. That is, the search cannot be done off-line because the search space depends on \mathcal{O} .

Our implementation of RaP for the PRETCIL framework *searches through the observation space* as described above. Each of the $|\mathcal{H}|$ MGHS calls *searches for both goal states $G + O$ and $G + \bar{O}$* for $G \in \mathcal{H}$ as described in Section 6.5.1.1. The heuristic assigned to both goal conditions in a single MGHS (sum heuristic, A* search, unweighted) run is simply the *original A* heuristic used to search through the state space*. This takes advantage of the case where MGHS's set of *goals are similar enough to explore the same region of the search/observation space* so that we avoid both redundant and unnecessary state exploration.

6.5.2.2 Connection to Plan Libraries

When Ramírez and Geffner introduced planning domains as an alternative to plan libraries [226], the primary advantage they provided was an increase in flexibility for recognition. Plan libraries are static due to precomputation and lacking sufficient information to generate new plans that might satisfy unexpected goal conditions or observation sequences. On the other hand, the *planning domain dynamically computes sets of plans based on the current set of hypothesized goal conditions, initial state, and observation sequence*.

However, the majority of RaP approaches use a fixed initial state and set of hypotheses while at most pruning the set of possible goals; *they do not generate novel goals* with the exception of E-Martín, R-Moreno, and Smith’s approach that returns the most-likely fluents to be satisfied without a clear notion of a goal [68]. Many of these approaches also apply the observation sequence to the search method much like the original RaP algorithm’s augmented state space. As we explained in our variations above, this reliance on the observation sequence, which cannot be known until recognition time, prevents the ability to use much precomputation. The only speed-up techniques so far that use off-line computation are Masters and Sardina’s [189], which precomputes the optimal plan that is assumed to not satisfy any possible observation sequences, and Pereira, Oren, and Meneguzzi’s [217], which precomputes a data structure based on landmarks for each hypothesized goal.

When one *decouples the observation sequence from the search process* as in our approach, then our first variation reveals that precomputation is possible when we *assume that the set of hypotheses \mathcal{H} is complete and the initial state is fixed*. Pruning a hypothesized set of goal conditions G is equivalent to ignoring the precomputed Π_G . That is, the set

$$\mathcal{L} = \bigcup_{G \in \mathcal{H}} \Pi_G$$

is a plan library that serves the same purpose as the plan domain for current RaP algorithms. In the case that our approach generates some \mathcal{L} and it is missing a plan, then Theorem 6 bounds the error of the recognition probabilities.

As HTNs are also used for recognition [92] and represent plan libraries in a more compact form (as well as have their own scalable approaches [146]), we propose the option of storing each $\Pi_G \subseteq \mathcal{L}$ as a trie data structure. Rather than prefixes of letters (one per node) per word overlapping, the *prefix of state (one per node) sequences overlap with edges labeling the actions* that make the state transition possible. Due to the fixed initial state, all plans in Π_G overlap at the root node at a minimum.

Then *matching the observed transitions in \mathcal{O} for recognition reduces to traversing the trie*. Additional bookkeeping is necessary to account for missing observations in \mathcal{O} , namely branching pointers at nodes to explore each subtree for the next observation in the sequence. If one of these pointers reaches a leaf node without matching some observation in \mathcal{O} , then that traversed plan solves $\mathcal{P}_{\mathcal{O}}^G$. If one of the pointers is at some node in the trie when it satisfies all observations in \mathcal{O} , then the set of all plans in its subtree solves $\mathcal{P}_{\mathcal{O}}^G$. This recognition algorithm can run on-line, moving pointers further through the trie with each new observation—the precomputed trie can also *agglomerate all the information about each node’s subtree’s plan costs* for the sake of computing probabilities for RaP.

Due to the abstractness of tasks rather than directly representing states, HTN-based plan libraries are traditionally more flexible than these plan domain-generated-plan libraries. That is, changing the initial state will likely require recomputing the trie. Both versions of plan libraries require updating when the set of considered hypotheses adds a novel hypothesis. Ultimately, this implies that a *whole class of RaP algorithms remain unexplored that actually take advantage of plan domains’ untapped potential*.

6.6 Concluding Remarks: Influences of Automated Planning and Recognition on Each Other

From entertainment to personal assistance, intelligent systems are interacting with people in a variety of applications. However, even when these systems appear to act autonomously and allow the user free will, there is usually extensive back-end development to engineer the interactive experience. Though not as restrictive as expert systems with hand-coded tables of what to exactly do in every considerable situation, there is usually a fixed set of inputs or outputs that maps from or to artificial intelligence algorithms. For example, natural language interfaces might perform speech-to-

text and then map that text to a set of expected inputs through parsing or machine learning. Likewise, embodied agents might have a preprogrammed finite state machine that specifies what output behavior to perform, and task and motion planning algorithms determine how to execute those behaviors given the current environment’s configuration.

Even though these intelligent systems exhibit artificial intelligence and account for the environment and stimuli, *they are not actually interacting with an understanding of the user*. People act with purpose, explore their environment, make mistakes, and sometimes change their mind in the middle of doing something. Closed-loop interaction addresses this by *modeling users and making decisions with respect to those models*.

We thus introduced the Planning and Recognition Together Close the Interaction Loop (PRETCIL) framework as a cognitive architecture. Unlike existing closed-loop interaction architectures, the PRETCIL framework iterates indefinitely to *update the recognized intents and plans using perception and expected user responses* while also *revising decisions about how to act based on these updates*.

As a general framework, one can apply any appropriate algorithms to the PRETCIL framework. In this chapter, we implemented it using responsive planning [82] and recognition as planning [227] as the primary components. We reviewed the specific implementations for each component conceptually, theoretically, and empirically, including versions that did not work out. As such, the approach outlined above is by no means the only or best way to implement closed-loop interaction. Instead, it serves as a *first step* towards accomplishing this important research challenge by considering the *role different AI challenges play in interactive intelligent systems*. The goal of this dissertation is to provide evidence that various areas of AI are mature enough for integration so that we can begin to *address more complex challenges that*

require their intersection. We continue this discussion in Chapter 8 with a roadmap to possible extensions, revisions, and future challenges.

CHAPTER 7

HUMAN PERCEPTIONS OF ASSISTIVE AGENTS USING THE PRETCIL FRAMEWORK

RGF: You know, it will be nice to apply research to something other than spherical cows.

MCJ: Like cuboid cows? They aren't equidistant from the center all over.

RGF: No, no. I mean working with people.

MCJ: Oh, spherical humans seem fine. Carry on.

— Richard (Rick) G. Freedman,
in a typical conversation with Marvin C. Jones

Our primary goal behind the PRETCIL framework's conception is to better understand how intelligent systems can interact with others, especially people, in the same way that people interact with each other. This means that theoretical findings and computational results are not sufficient to evaluate our *overarching hypothesis that the PRETCIL framework is a first-step* towards this goal. We did run an empirical evaluation of two computational agents, one using Chapter 6's PRETCIL framework implementation and the other just using a vanilla A* search, for comparison against RIKER in Levine's dissertation [175]. We encourage the reader to read that dissertation's experiments if they are interested in such results, but we consider them insufficient for this goal because *A* alone does not appropriately model human decision making*.

To inspect how our agent interacts with people, we must allow people to interact with an agent employing the PRETCIL framework and *receive their feedback and impressions*. Unlike the Turing test, we are not interested in a binary question of the

form ‘does interacting with this system feel like interacting with a human?’ People serve as *potential interactive partners* who decide whether to interact with the system and for what reasons. They (R_{ed}) generally avoid others (R_{ing}) who are less beneficial when seeking an interactive partner for a task. We use the term ‘beneficial’ instead of ‘helpful’ because the novelty effect [264] is still widespread—people might interact with intelligent interactive systems because they are new enough to be ‘cool’ and/or ‘fun’ even if they are not helpful.

In preparation for this human subjects experiment, there were several phases that each had unique challenges and setbacks. For those who plan to run such studies in the future, we discuss these phases and the evolution of our approach in order to address the challenges along the way. Following these preparations, we present and discuss the outcomes of our small, preliminary study. As far as we are aware, this is the *first study with human subjects that involves people’s experience with a closed-loop intelligent interactive system*.

7.1 Human-Subjects Studies with Intelligent Systems

Testing interaction in multi-agent systems when all the agents are machines is *conceptually simple to set-up and run*: place the machines together in a controlled environment, let them interact, collect data, and pause experiments whenever a bug occurs or a machine malfunctions (resuming/restarting after fixing the issue). The experiments throughout Chapter 6, which focus on single-agent testing, follow a similar procedure. We emphasize the word ‘conceptually’ because this is still easier said than done—controlled environments can be difficult to design and implement (for example, search and rescue scenarios [132, 161]), data collection methods can fail or miss important information (for example, a wall-mounted camera will not record a robot’s activities when it faces the opposite direction), and fixing malfunctions alone

can be an intense challenge when a machine’s physical parts need replacement or the bug corresponds to an obscure edge case.

In comparison, testing interaction in multi-agent systems where some subset of the agents are humans can be far more difficult. At the moment and likely the near future, machines lack autonomy at the level of a will of their own and do exactly as they are told¹. On the other hand, people have the ability to decide whether they want to participate in research studies and determine the extent of their involvement. Putting machines together into the controlled environment and turning them on is as simple as the efforts to move the machines into place, but adding people into these environments is not as trivial as shepherding them into place. Ethics issues also play a role in defining recruitment strategies and fair compensation practices for participation. People’s right to withdraw from a study also makes *pausing experiments much less desirable* if there is an error. That interaction’s data point is often lost, and later trials might become delayed such that other people no longer have availability and cancel rather than reschedule.

The fields of human-computer interaction and human-robot interaction have been dealing with these challenges for a while and further identified various types of human-subjects studies. There are far too many to list in this dissertation, but we briefly introduce the three that are most related to our study in this chapter: Wizard-of-Oz, human performance change, and human emotional evaluation.

Going beyond a paper prototype to investigate people’s expectations and interpretations of an interface mock-up, *wizard-of-Oz* (WoZ) studies [232] expose people to a machine that lacks an implementation of the desired intelligence algorithm. To compensate for the missing functionality, a *human expert hides outside the controlled*

¹The author is skeptical that computational systems will have such levels of autonomy and/or intelligence anytime soon based on experience and seeing the progression of state-of-the-art research. However, an anomalous discovery that could change this belief, however unlikely, is always possible.

environment and operates the machine remotely; however, the human subjects are *told that the machine has the missing intelligence algorithm and asked to interact naturally with the machine*. The human expert thus perceives and interacts with the human subjects via the machine interface, and researchers inspect how people behave in order to identify their expectations and interpretations of machines employing the algorithm. This is very helpful for algorithmic design, particularly knowing what people expect the machine to do and in which ways people will engage with it [198].

As a more traditional case-control experiment, *human performance change* studies inspect *how people perform with and without the machine's intervention*. Unlike WoZ studies, the machine already has the desired functionality so that there is no degree of deception. Short-term studies usually involve human subjects performing the task both with the test machine and on their own/with a comparative machine, and researchers compare relevant metrics between these instances. Short-term studies often inspect the immediate benefits of intelligent tools [100], human-enhancement/augmentation devices [172], or interactive partners [101]. Longitudinal studies span long periods of time with occasional sessions [140] or ongoing interactions with the machine [264], measuring the human subject's performance at various moments throughout the study. The evolution of the performance provides evidence for the intervening machine's impacts, which is compared (when possible) with the evolution of the performance of human subjects who lacked the intervention.

As a middle-ground between WoZ and human performance change studies, *human emotional evaluation* studies evaluate *how people perceive machines that exhibit certain properties* [165]. Researchers run most human emotional evaluation studies almost identically to human performance change studies. However, rather than measuring the human subjects' performance during the interactive experience, researchers provide surveys that ask them about their perception of the machine and/or experience. There is not always a before/after trial for comparison, depending on the

domain—a student is unlikely to learn the same lesson twice both on their own and with an intelligent tutoring system, but a student can participate in different lessons [102]. Emotional perspectives are helpful for iterative design, much like WoZ studies, because programmers can revise their system based on the acquired knowledge about how people engaged and felt about the machine. They are also a means of performance evaluation where the *performance is subjective to the human rather than objective to the task*.

7.2 Experiment Platform

For our human subjects experiment, people played a turn-based BlockWords game in a simulated environment shown in Figure 7.1. To influence the human subject’s goal-driven behavior, the interface displays a specific completion criteria that the interactive agent does not receive. A demonstration prototype (see Section 7.2.2 below) allowed people the freedom to select from a set of completion criteria instead. The human-controlled player (R_{ed} , for consistency with our theoretical notation in previous chapters) performs an action on the first turn, which provides some information to the interactive agent running our implementation of the PRETCIL framework (R_{ing} , for consistency with our theoretical notation in previous chapters) about which criteria they intend to satisfy. Then the interactive agent performs an action if they received sufficient information to decide how to respond.

Due to the simulated game setting, user inputs were limited to discrete key/mouse presses that the computer can easily interpret without raw sensor data. Thus this implementation of the PRETCIL framework simply *performs activity recognition as a mapping from the input to the game’s corresponding action* rather than employing the topic-modeling techniques from Chapter 4 or some other activity recognition method from Section 3.2.

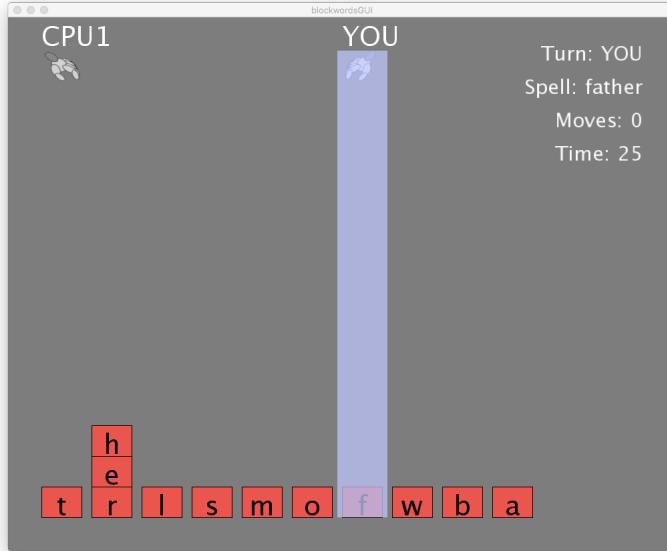


Figure 7.1. Screenshot of the BlockWords two-player game.

After each human-controlled player’s turn, the interactive agent adds the latest action to the observation sequence \mathcal{O} . On the interactive agent’s first turn and any subsequent turn that they do not have a joint-multi-agent plan, the PRETCIL framework implementation runs RaP with MGHS (pairs of goals, sum heuristic, A* distance; see Section 6.5), computes the necessities with respect to threshold τ , and then computes a joint-multi-agent plan that solves the necessities via a single-goal A* search defined within the MGHS code. Neither the interactive agent nor interface reveals this joint solution to the human-controlled player. While this plan is active and considered correct according to SPAM (see Section 6.2.3), the interactive agent executes the plan’s next action on their turn in response to the human-controlled player. Like the activity recognition component’s implementation, the simulated game setting allows us to simply *map the action name to an animation and update the state* without a complex execution algorithm.

7.2.1 BlockWords: Good for Human-Computer/Robot Interaction?

BlockWords was designed to be a challenging domain for machines, and finding solutions thus provided evidence of intelligent decision-making capabilities. People often engaged in similar domains since they were children, especially when playing with toy blocks spelling words. This gives people an advantage when performing a BlockWords task, which could interfere with measuring changes in their performance—*how can an intelligent interactive system help someone with a task whose solution is too trivial to require help?*

People are also very good at generalizing their plans and finding patterns that yield *domain-specific strategies*. In the case of BlockWords, the generalized strategy is to *uncover and stack each block in order to spell the goal word backwards*. For the next target block b , the player’s best strategy is to find the stack containing b , remove all blocks on top of b from that stack and place them on the table in their own stack, and then place b on top of the stack that spells the goal word (when b is the bottom-most block, then it goes on the table).

A hierarchical task network (see Section 2.3) easily captures this strategy, but our approach uses state-space search (see Section 6.4) without any domain information. Therefore, *most people will quickly realize and adopt the generalized strategy*, finding an obvious solution to each level while the *interactive agent continues to blindly search from scratch* on every level. Future iterations of this study will account for alternative interactive agent implementations that appropriately apply this domain knowledge, and we will also investigate other domains that are less trivial for people.

7.2.2 Revisions to Platform

We demonstrated an early implementation of the PRETCIL framework described above at the Twenty-Ninth International Conference on Planning and Scheduling (ICAPS) in July 2019. Approximately fifteen conference attendees watched others

interact or directly interacted with the demonstration over the one-hour period that it was on display. Based on the authors' observations of their experiences and feedback, new questions and research challenges emerged involving the integration of automated planning and recognition within an interactive domain. The majority of these appear in Chapter 8 as future work, but we addressed the ones that had simple platform revisions and were too important to ignore before running the human subjects experiment.

The quickest change addressed the “too few observations” issue where problem $\mathcal{P}_{\neg O}^G$ has no solution at the beginning of the interactive experience. As Section 6.5.2 mentions, BlockWords always has this issue if the *first block that is picked up happens to be on top of a stack containing any block whose letter spells the goal word* and that block exclusively displays the letter. We added a ‘head-start’ parameter ν where the human-controlled player took ν turns before the interactive agent ran the PRETCIL framework. For these ν turns, the interactive agent simply performs a no-op action to pass its turn. The human-controlled player’s accumulated actions greatly increases the chance that O is sufficiently long enough to not block some solution to $\mathcal{P}_{\neg O}^G$. A more effective solution would be to add a *domain-specific no-solution check that abandons a goal condition set* prior to searching, similar to how 8-puzzle instances in different permutation groups cannot reach each other [237].

The most critical issue was a surprisingly frequent case where the interactive agent stopped acting in the middle of the interactive experience. We identified two primary reasons for this phenomenon in our implementation:

1. O increased with many observations that were not part of any optimal solution because *people are not expected to solve the puzzles optimally*. Thus for all hypotheses $G \in \mathcal{H}$, all solutions to \mathcal{P}_O^G eventually have much greater costs than the solutions to $\mathcal{P}_{\neg O}^G$, which are the optimal plans for the original planning problem \mathcal{P}^G . Then the probabilistic RaP likelihoods approach 0 for all hypothe-

ses, but are not actually 0 so that they still normalize to a valid distribution over \mathcal{H} . Unfortunately, *computational underflow eventually rounds these small probabilities down to 0* such that the distribution over \mathcal{H} is all 0's. Then the set of necessities becomes \emptyset and the *optimal joint-plan happens to be a plan with no actions*.

2. If the WCD between multiple goal condition sets $\mathcal{G} \subseteq \mathcal{H}$ is great enough and one of them is the human-controlled player's true goal, then each hypothesis in \mathcal{G} will eventually become very likely without any disambiguation between them, sharing the greatest probability mass and contribution to the necessities. If these goal condition sets further vary by some small feature that cannot coexist (for example, placing both the T and H blocks on top of the E block), then the *updated necessities will include these contradictory conditions and no longer have a solution*.

Although each of these causes is an issue to consider fixing, the consequence of any of these cases is more severe. Without a plan to follow, the *interactive agent performs no-op, often for the remainder of the interaction*. People were a little disappointed, but not concerned, if this did not affect the human-controlled player's performance. However, if the interactive agent was initially helping and picked up a necessary block when this occurred, then they essentially *performed the adversarial hoarding strategy* $\pi_{R_{ing}}^{hoard}$ from Section 5.4 such that the human-controlled player could not complete their task. The solution to this issue was also simple: *redefining the 'default no plan' action based on the BlockWords domain*:

- If the interactive agent is not holding any block, then they perform a no-op action.
- If the interactive agent is holding any block, then they perform a put-down action that places the held block onto the table in its own stack.

With this default, the human-controlled player still has access to all the blocks to spell whatever word they desire.

The first reason for the above issue also played a role in the last problem. When \mathcal{O} was long enough that probabilistic RaP returned a distribution of all 0's over \mathcal{H} due to computational underflow, the program entered a *point of no return in ongoing iterations*. The human-controlled player continues to take actions, which further increases \mathcal{O} 's length so that probabilistic RaP continues to return a distribution of all 0's over \mathcal{H} . Running the probabilistic RaP algorithm is then feckless and wasting time in the interactive experience. People playing the game showed signs of irritation when *the computer-controlled agent consistently took time to decide what to do and then simply did nothing* (the default action during the demonstration). Because the distribution of all 0's was the trigger, our simple fix to this issue was to check for this case and mark a *give-up flag* when it occurred. When a later iteration of our implementation of the PRETCIL framework encounters the give-up flag, the computer-controlled agent immediately performs the assigned default action. Upon updating the default action, we removed this feature at the beginning of the experiments; Section 7.5 explains how this was not a good decision and the give-up flag was reinstated.

In hindsight of the above revisions, there are likely some better, equally quick fixes. We will consider them in future iterations, but list them below rather than in Chapter 8 due to their immediate relevance:

- **Log Probability** In other fields of computer science and mathematics, multiplying many probabilities together has already caused the above computational underflow problem. To avoid their products turning into 0's, they usually use the *logarithm of the probabilities* to convert the multiplication of small decimal values into the addition of large negative numbers. Likewise, high-cost

plans yielding small probabilistic RaP likelihoods would instead produce large negative log-likelihoods.

- **Theorem 6** To reduce probabilistic RaP’s runtime and avoid reaching a probability of 0 so quickly, we can add additional search stopping conditions besides the give-up threshold k (not to be confused with the give-up flag feature). We proved the error bound that justifies a ‘corner-cutting trick’ after finding a solution to one of \mathcal{P}_O^G or $\mathcal{P}_{\neg O}^G$ in Section 6.5.2.
- **Restart \mathcal{O}** When \mathcal{O} becomes so long that we can *guarantee that the human-controlled player is not acting optimally*, then *Masters and Sardina’s assumptions [189] both certainly hold* when the give-up flag is set. That is, the optimal plan solving the original problem \mathcal{P}^G is also the solution to RaP’s subproblem $\mathcal{P}_{\neg O}^G$ —then we simply store this plan’s cost and only solve \mathcal{P}_O^G . Furthermore, because we observe the human-controlled player without any missing observations, we can ignore \mathcal{O} and simply solve \mathcal{P}^G from the current state. Although we considered this approach, we did not choose to implement it due to the difference that the *computer-controlled player also influences the environment*, which could bias RaP when starting from the current state.

7.3 Participant Recruitment

When preparing the experimental procedures, we initially predicted that the time for reading/signing the consent form, interaction with the game, and completing the survey would last around one hour. More importantly, we did not offer any form of compensation to human subjects for their participation because

- The study was not funded by any grant and
- The researchers, including the author of this dissertation, were all students on a limited budget.

The Institutional Review Board (IRB) approved the lack of compensation due to the researchers' justifications that:

- The time commitment of one hour is small and
- Playing the game adds a novelty/fun factor for participants in the study.

With the recruitment methods primarily targeted towards undergraduate students, it seemed that these would be reasonable incentives. By the time that the code was ready and recruitment began, though, the lead researcher (and author of this dissertation) moved to another state and was unable to perform the study there due to a lack of IRB approval at nearby institutions. The approved recruitment methods did not generalize to the general public (flyers, announcements in front of classrooms, and e-mails over listservs), which meant that the lead researcher had to return to the institution with IRB approval in order to perform the study. Together with the time constraint for completing the work in this dissertation, this limited the study's duration to one week rather than ongoing until enough subjects participated. This also limited the time of recruitment to several weeks before the study and throughout the study's duration.

The recruitment phase had the most unforeseen challenges due to changes in the institution's bureaucratic processes² between the study's original submission to the IRB and the beginning of the study's execution. Specifically, e-mail listservs that broadcasted information to undergraduate students were now restricted to institutional use only, which eliminated the primary method of advertising the study and recruitment. A social e-mail listserv for the graduate students was still available, and some professors of graduate student-only courses were willing to do the classroom announcement before the study began. As one would expect, graduate students are

²The author of this dissertation has newfound appreciation for the relatively low bureaucratic burden at his current employer, Smart Information Flow Technologies (SIFT), LLC.

much busier and very few signed up to participate in the study with its lack of compensation for one hour’s worth of their time. The department did kindly distribute fliers in some limited places, but few students seem to pay attention to these bulletin boards.

Upon arriving at the institution to perform the study, only six participants signed up out of the goal of twenty subjects. Several participants had to reschedule due to unexpected events (the original session they gave up was not filled with a replacement in time), and a few other participants did not show up for their chosen session. During the week of the study, additional people signed up, but most of them chose time slots that overlapped with another person who signed up and did not select alternatives. By the end of the week, a total of seven human subjects kindly participated in the study.

7.4 Experiment Procedure

The experiments took place in a reserved, private conference room to create a controlled environment containing only the on-site researcher, laptop running the experiment platform described in Section 7.2, and current study participant. Each scheduled trial involved a single study participant, and they came to the reserved conference room to perform the experiment.

Before beginning the experiments, the on-site researcher thanked the participant for taking the time to be involved in the study, offered to answer any of the participant’s questions throughout the study, and then provided an IRB-approved consent form that the human subject read and signed (each subject received a copy of the form to keep for their own records). The on-site researcher only answered one question vaguely, “how does the computer-controlled agent work?”, because the *information could affect the human-controlled player’s behavior* and influence the results. To avoid this concern, the on-site researcher promised to answer the question following

the trial—all participants who asked this question seemed satisfied with the response, and the on-site researcher summarized the implementation of the PRETCIL framework following the trial’s completion.

After the participant signed the consent form, the on-site researcher seated them in front of the laptop and set the game up to run. The game itself is simply a graphical user interface (GUI) and server that displays the current BlockWords state, processes the human-controlled player’s actions, and then communicates with other programs over a socket for the computer-controlled player’s actions. The reason for this multi-process design was two-fold:

1. We wrote the game in the Processing programming language, which facilitated the GUI design. However, we wrote the computer-controlled agent in the C++ programming language because of our PRETCIL framework library implementation (see Section 6.3.1). Socket communication between the separate processes allowed us to pass information between them without needing to create a wrapper software package between languages.
2. In case we reuse the game to run a WoZ study in the future, the socket setup makes it easy to change who controls the computer-controlled agent(s) in the game. Specifically, we only need to create a client for human experts to play the game remotely (in another location away from the human subject). Then this client communicates with the game’s server in place of an intelligent interactive system.

After starting the program that ran the game, the on-site researcher executed a separate program for the agent running the implemented PRETCIL framework. Although the socket communication sometimes caused problems due to race conditions, the participant pointed out whenever the game seemed to freeze and the on-site researcher restarted the programs at the current level.

The first two levels of the game are a tutorial, and the on-site researcher joined the participant at the laptop to watch their performance and provide feedback about how the game works. The first tutorial level challenges the participant to spell a single word by themselves. This tutorial level *teaches the participant about the game’s controls and how to accomplish the goal*. The second tutorial level repeats the initial setup and goal word, but includes the computer-controlled player. This tutorial level *teaches the participant how turn-taking works with the computer-controlled agent*. However, to avoid any trouble with this agent during the tutorial, the *hypothesis set only contains the goal word*. Thus probabilistic RaP guarantees predicting the correct goal word with probability 1 and copies it into the necessities, which makes the assistive interaction problem almost trivial.

Upon completing the tutorial, the on-site researcher left the participant alone to continue playing the game. The on-site researcher remained in the room to answer any questions and restart the game whenever it seemed to stop working. Levels throughout the game provided different BlockWords problem instances and either challenged the participant to play on their own (one-player) or with the computer-controlled agent (two-player). As the participant played, *the game recorded each player’s action with a relative timestamp*. This log can replay a trial, which allows the researchers to *analyze the human performance change* with respect to runtime and number of actions to complete various levels.

After the participant finished all the game’s levels or chose to stop playing, the on-site researcher provided a survey for the participant to complete. The *survey’s statements address the participant’s emotional response* to the interactive experience:

1. It was more interesting to perform the tasks alone than with an interactive agent.
2. It was engaging to perform the tasks alone.

3. It was engaging to perform the tasks with an interactive agent.
4. I enjoyed the tasks when performing them alone.
5. I enjoyed the tasks when performing them with an interactive agent.
6. Performing tasks with an interactive agent was more frustrating than performing tasks alone.
7. I enjoyed working with the interactive agent.
8. At times, I became frustrated while working with the interactive agent.
9. At times, I became annoyed while working with the interactive agent.
10. My decisions were influenced by the interactive agent.
11. Additional feedback or comments:

The first ten statements use a 5-point Likert scale from -2 (strongly disagree) to 2 (strongly agree). The last statement is free-response for the human subjects to optionally provide any feedback, ideas, or other responses. When the participant finished the survey, they returned it to the on-site researcher. The on-site researcher then thanked the participant again for taking the time to be involved in the study and offered to answer any remaining questions, including “how did the computer-controlled agent work?” After the participant left the conference room, the on-site researcher generated a random, unique identifier for the participant and used it to label the survey in place of their actual name.

To preserve the anonymity of the participants, we locked all signed consent forms and physical survey responses in a drawer in a secure office space. We then locked a key mapping each participant’s name to their unique, randomly generated identifier in a *separate drawer*. Both the IRB protocol and consent form described this *process for privacy maintenance*. Participants have a right to withdraw their information

following the trial, and this setup allows the researchers to be able to find the participant’s data for removal *without leaving a connection between them and their data that someone else could easily compromise.*

7.5 Experiment Results

Although seven participants is a small sample size for a study, we further divided them into two groups. Each group had a small change to the game: the available levels and the presence of the give-up flag feature (see Section 7.2.2 for details). The first group (four participants) had fifteen levels, and the last few were difficult for the computer-controlled agent employing probabilistic RaP and blind search. That is, *more difficult levels have much larger state spaces that require additional time to search for individual plans.* With optimal play, this additional time did not seem to be too worrisome, and we wanted the experiment to include a variety of performances to *honestly reveal the computer-controlled agent’s best-case and worst-case behaviors.* However, the human subjects did not always play optimally and these worst-case runtimes became unreasonable, especially under the promise of the study lasting at most one hour. Some participants requested terminating the interactive experience early when these long computational times began to occur, and we realized that *we needed to remove the difficult levels and allow the agent to give-up trying to interact.* Thus the second group (three participants) had a suite of thirteen levels, removing the two levels that were difficult for the computer-controlled agent’s search algorithms even with the give-up flag feature available.

Due to the small sample sizes in this experiment, *we cannot make any formal conclusions from these results.* However, from the perspective of a WoZ pilot study, these results serve as a *source of hypothesis generation for future studies.* That is, we use these results as observations that formulate testable hypotheses. Then, we can *formally test them using well-designed experiments that specifically focus on that*

particular feature, introducing a tighter controlled environment to avoid potential confounding factors. Unlike traditional WoZ studies that rely on a human actor and aim to understand how to better design the machine, our WoZ study directly used the machine itself in order to understand *what aspects of the human-computer/robot interaction are worth investigating*.

7.5.1 Human Performance Change

The small sample size and change to the game both hindered the initial design for testing hypotheses regarding human performance changes. We originally intended for only the tutorial levels to be fixed while the remaining levels, which would have a constant problem instance, randomly selected the number of players. Then there would have been two classes of participants, one playing the level alone and the other playing the level with the interactive agent, and we could *compare the differences in the two sample population's performance metrics*. Our hypotheses were that the execution time and number of actions (plan cost) would both decrease when the interactive agent running our implementation of the PRETCIL framework was involved.

These hypotheses appear to imply positive helpfulness, but they compare the *average population's performance change—helpfulness should be agent-specific*, similar to measuring a treatment's effects (compare ‘how many individuals improve?’ to ‘does the average improve?’). We wanted to measure helpfulness (hypothesizing it would be positive) as we did in the study comparing the PRETCIL framework implementation prototype against RIKER [175], but this is more difficult because the *human must perform the same problem instance both with and without the interactive agent*. When the human-controlled player was another computer simply running A* search for decision making, we could guarantee a consistent helpfulness measurement for two reasons:

1. For a given problem instance, A* will perform consistently, even with random tie-breaking strategies, to return an optimal plan. A *person is not guaranteed to perform exactly the same way in the same situation*. Accounting for this variance would likely require more than one run of the scenario, and people will likely get bored of such repetition.
2. Unlike an algorithm that constantly runs the same programmed procedure, *people can learn and adapt how they solve problems*. Besides accounting for reason 1. as the person repeatedly plays the same level, they might remember their solution from playing with/without the interactive agent when replaying the level without/with the interactive agent and update their plan. Even with many levels in between these two instances, we cannot guarantee what the person will forget and remember.

We thus worried that we could not accurately measure an individual's performance with a single-session study design. To *appropriately measure helpfulness* with a lower chance of the above reasons taking effect, we propose an alternative study where *a person must play the game over multiple sessions*, randomly choosing the number of players and permuting the levels each session.

Instead of testing the above hypotheses, we collected the execution time and number of actions per level per participant via the game's logs. The only hypothesis we can consider testing is that the inclusion of the give-up flag feature reduces the execution runtime, but it is better to *design an experiment that specifically tests this* rather than consider it ad-hoc based on the sudden experimental design change. As such, Figure 7.2 displays the consumed temporal resources per subject per level, separated by group. To allow some degree of consistency during the experiments, we preassigned the number of players per level and kept the assignments constant for all seven participants (instead of random assignment as mentioned earlier).



Figure 7.2. Temporal resources consumed per non-tutorial level during the human subjects experiment. Levels with no results are not shown—all Group 1 participants skipped them, and they were removed for Group 2.

The most obvious observation about these plots is that there generally appears to be a *positive correlation between the plan cost and time to execute that plan*. Assuming that all actions have uniform cost, this is not very surprising. Only Level 12’s plot appears to disagree with this trend, but three data points is not sufficient to make any conclusions.

We measure the execution time as the difference between the time stamps of the human-controlled player’s first action and last action, which means that ‘thinking time’ to generate an initial plan is not included. Otherwise, we would have considered that participants took longer to think of a more optimal solution in Level 12. We do not include any measurements of this ‘thinking time’ because they greatly varied between individual participants and levels with *no signs of correlation to the actual temporal resource consumption*. Perhaps a larger sample size could reveal different classes of people who put various amounts of consideration into their initial planning before starting execution. If we can identify how long someone thinks before they begin to act, then the interactive agent can *take advantage of that ‘thinking time’ for prudent off-line computation* rather than doing nothing until receiving an observation.

The second-most obvious observation about these plots is that there is generally variance across the x-axis (the plan cost). This confirms that *participants did not always execute optimal plans* because the set of optimal plans per level have a single cost. However, there are two further observations worth making about this variance in plan costs:

1. The variance in plan cost is greater in two-player levels. This might be a consequence of the interaction if the computer-controlled player makes any mistakes or violates the human-controlled player’s expectations. A larger number of participants is necessary to study the impact of such factors.
2. Amongst the one-player levels, the variance in plan cost decreases until everyone (the one-player levels did not change between groups) performs optimally. This

might be evidence that people are improving on their own as they continue to play, supporting our concerns in Section 7.2.1 about the simplicity of the BlockWords domain.

The final performance-related observation worth noting is that the execution times are generally lower for participants in Group 2 than those in Group 1. This provides evidence supporting *further investigation of the ad-hoc hypothesis*, but we cannot make any conclusions with the current results. Because we only logged information from the GUI and not from the agent running the PRETCIL framework implementation, we cannot easily identify when they should have given up during the Group 1 trials for a fair comparison.

7.5.2 Human Emotional Evaluation

A collection of seven survey responses, split into two groups of four and three, is far too small to summarize with a five-value summary like the data presented in previous chapters. Instead, we compute the mean μ and standard deviation σ over each survey statement's responses and present them in Tables 7.1 and 7.2 respectively. We designed our Likert scale to have the following properties:

- A score of 0 indicates indifference towards the emotional statement.
- A score less than 0 indicates disagreement with the emotional statement.
- A score greater than 0 indicates agreement with the emotional statement.
- A score's magnitude is proportional to the strength of agreement/disagreement with the emotional statement.

The Law of Large Numbers is clearly not satisfied to assume that these values describe a normal distribution. First, the majority of the standard deviations are between 1.5 and 2 when evaluating a 5-point Likert scale, which implies that *individual*

Table 7.1. Human Subjects Survey: Response Means

	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10
Group 1	0.25	0.75	-0.25	0.25	-0.75	1.00	-0.25	0.75	1.50	-1.25
Group 2	-0.33	-0.33	1.67	-0.33	1.67	0.67	1.33	0.33	-0.67	-1.00
Overall	0.00	0.29	0.57	0.00	0.29	0.86	0.43	0.57	0.57	-1.14

Table 7.2. Human Subjects Survey: Response Standard Deviations

	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10
Group 1	2.06	1.89	1.71	1.71	1.89	2.00	1.71	1.89	0.58	0.50
Group 2	2.08	1.15	0.58	1.15	0.58	0.58	0.58	1.15	1.53	1.73
Overall	1.91	1.60	1.62	1.41	1.89	1.46	1.51	1.51	1.51	1.07

scores are spread over most the spectrum of options and the mean fails to represent the majority of the population. We color Table 7.2's lower standard deviations red to indicate which statements' scores might be more focused about a particular value. The accumulated survey results in Figure 7.3 present the wide variety of responses between and within the two groups.

Table 7.3 recomputes the mean using the *absolute value of the survey responses*. If the absolute value of the means in Table 7.1 is close to the mean of the absolute values:

$$\frac{1}{N} \sum_{i=1}^N |r_i| - \left| \frac{1}{N} \sum_{i=1}^N r_i \right| < \epsilon,$$

then there is at least *evidence of consensus on (dis)agreement with the corresponding emotional statement*. We color Table 7.3's entries that satisfy this when $\epsilon = 0.3$ red. If these two means are reasonably different, then there is *evidence of a lack of consensus where some subjects agreed and others disagreed*, and the magnitude on each side is similar. The entries colored red in both tables are almost identical, which is not surprising because a *stronger consensus is expected to have a lower variance and spread of responses*. Thus, we only speculate based on the *possible implications* of each result because we cannot guarantee that our samples are a valid representation of the entire population.

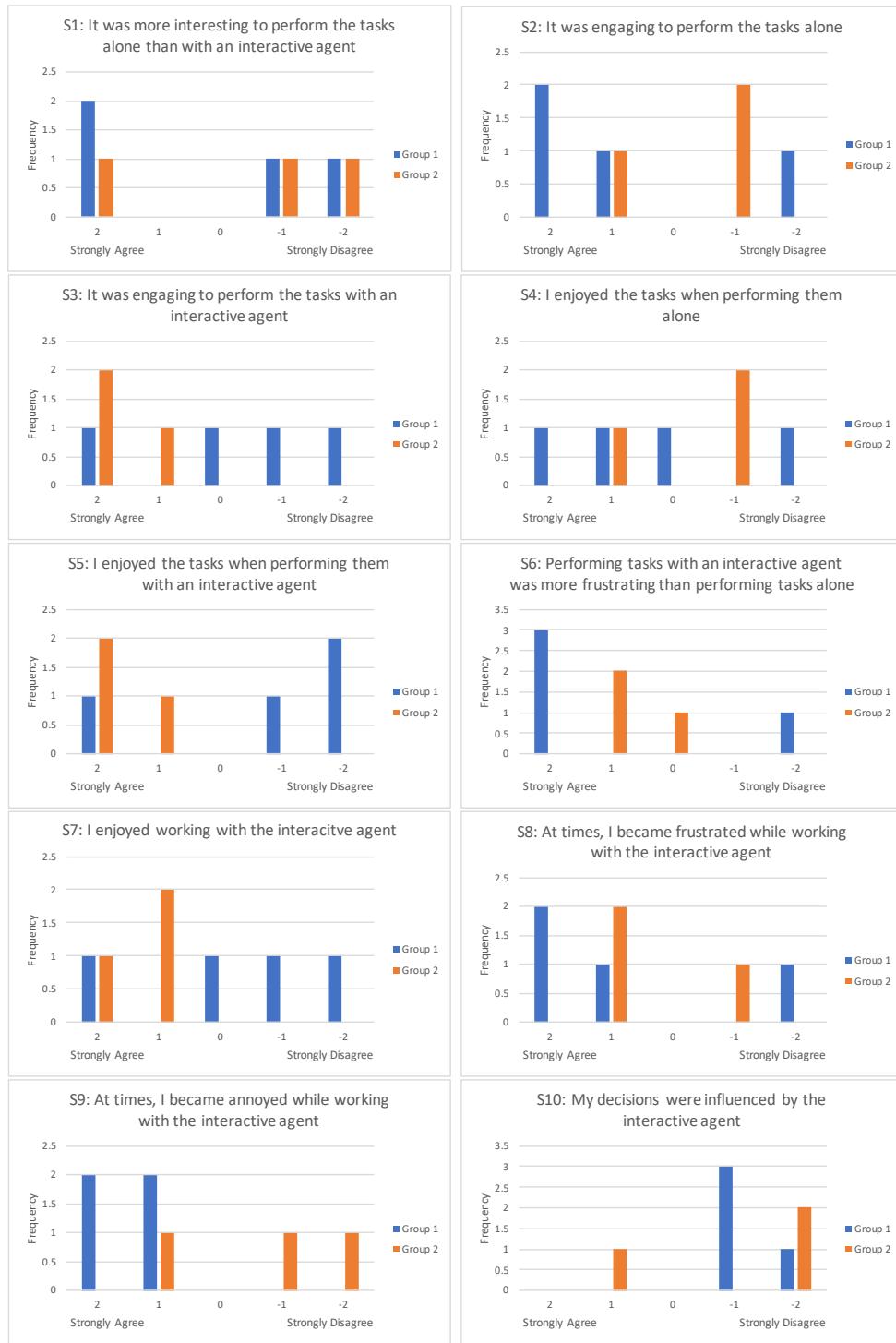


Figure 7.3. Survey responses from participants in the human subjects experiment. Participants completed the survey after finishing the game or choosing to terminate their play session.

Table 7.3. Human Subject Survey: Absolute Value of Response Means

	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10
Group 1	1.75	1.75	1.25	1.25	1.75	2.00	1.25	1.75	1.5	1.25
Group 2	1.67	1.00	1.67	1	1.67	0.67	1.33	1	1.33	1.67
Overall	1.71	1.43	1.43	1.14	1.71	1.43	1.29	1.43	1.43	1.43

We focus our initial observations on the responses whose distributions *appear to have a greater chance, but no guarantee, of representing the actual population*. These correspond to the values colored red in either Table 7.2 or Table 7.3, which relate to Figure 7.3's plots whose response frequencies primarily cluster on one side of the (dis)agreement spectrum:

Group 1 Participants who played the levels that were more challenging for the computer-controlled agent (and the agent could not give-up) . . .

- . . . became annoyed while working with the agent. (S9: $\mu \approx 1.5$, $\sigma \approx 0.58$)
- . . . did not feel that the agent influenced their decisions. (S10: $\mu \approx -1.25$, $\sigma \approx 0.50$)

Group 2 Participants who played the levels that were easier for the computer-controlled agent (and the agent could give-up) . . .

- . . . found performing the tasks with the agent engaging. (S3: $\mu \approx 1.67$, $\sigma \approx 0.58$)
- . . . enjoyed performing the tasks with the agent. (S5: $\mu \approx 1.67$, $\sigma \approx 0.58$)
- . . . were more frustrated performing the tasks with the agent than performing them alone. (S6: $\mu \approx 0.67$, $\sigma \approx 0.58$)
- . . . enjoyed working with the interactive agent (S7: $\mu \approx 1.33$, $\sigma \approx 0.58$)

Overall Participants who played either version of the BlockWords game with the computer-controlled agent . . .

- . . . did not feel that the agent influenced their decisions. (S10: $\mu \approx -1.14$, $\sigma \approx 1.07$)

The most curious of these observations is a trend that might be an anomaly: Group 2 participants were frustrated playing with the computer-controlled player, but they were engaged and also enjoyed playing with the computer-controlled player. In contrast, Group 1 did not have such a trend and were instead annoyed. That is, *do people accept working with a machine that responds sufficiently fast, even if their behavior is not helpful?* Previous research on human-machine teams found evidence that people are willing to work with their machine partners if they improve the task’s efficiency [99], but we doubt our agent is improving efficiency if it frustrates the human subjects. We worry that this observation is either an *anomaly of the sampled population* or a *novelty effect* where the currently unique experience of interacting with an intelligent interactive system amuses people [264].

Aside from the ten given statements, the majority of the participants’ free-response answers (S11 in the survey) in both groups stated that the computer-controlled player was slow. This indicates that despite all the research in Section 6.5, our *closed-loop interaction algorithms are still not fast enough to satisfy human expectations of response time*. This supports employing additional speed-up tricks such as pruning unlikely goals [281] and avoiding on-line computation, similar to PIKE’s [174, 176] and RIKER’s [175] approaches (see Section 6.1.1). However, one participant actually appreciated the slower computation time and anthropomorphically interpreted it as a contemplation behavior:

I thought the occasional “hang” was interesting whether it be intentional or due to computations. It gave me a sense of a “live” agent.

7.6 Concluding Remarks: Can PRETCIL Play with People?

Despite the convenience of sitting in an office with just a whiteboard and computer, theorizing about how machines can play with people is not enough to *actually create machines that can play with people*. The fields of human-computer and human-robot interaction have developed techniques for studying the relationships between people and machines, and the communities have begun to extend the definition of machines to include systems employing artificial intelligence [272, 114].

Using some of these techniques, we began to study real interactions between people and our implementation of an assistive interactive agent based on the research throughout this dissertation. Although there were unexpected challenges that prevented us from running the experiment as planned, this is still the *first time that people experienced an interaction with a closed-loop interactive intelligent system*. Between the data collected directly from this experiment and the feedback from participants at the ICAPS 2019 demonstration, there are *many new research opportunities, ranging from answering questions to testing hypotheses*. Chapter 8 ends this dissertation with a discussion about such topics, which we leave to future work both for ourselves and others interested in studying closed-loop interactive intelligent systems. *Many novel situations from interactive experiences and integrated frameworks take traditional artificial intelligence algorithms out of their original context*, and we need to address them as we continue to study and create intelligent interactive systems that actually interact with people. Whether using the PRETCIL framework or another model, the answer to this concluding remarks' title is, “not yet.”

CHAPTER 8

THE NEXT STEPS FOR DECISION MAKING FOR INTERACTION

Many of us have experienced the adage first-hand that closing any door leads to opening many new ones. Our acquired knowledge leads to new opportunities, and the answers we receive only lead to more questions.

— Richard (Rick) G. Freedman,
an excerpt from his submission to the University of Massachusetts
Amherst’s Graduate School Commencement speech contest

This dissertation’s initial investigation with the PRETCIL framework provides evidence *supporting the potential for integrating recognition and decision making to close the interaction loop for autonomous systems*. Deciding how to interact with others benefits from understanding what they are doing, and those decisions in turn influence the expectations over how they will respond. This creates an indefinite procedure for at least the duration of the interactive experience, but individual interactions are rarely one-shot, independent events over one’s lifetime. The integration of these interactions define who we are and how others perceive us, and hopefully interactive intelligent systems will one day be a part of this phenomenon as well. The hypotheses and novel avenues explored throughout this dissertation have many possible directions to consider next, and we present their general classes below as challenge topics for future research.

8.1 Challenge Topic: Sufficient Information to Interact

For the demonstration, we set two parameter values manually when initiating an interactive session: the necessities threshold and the number of turns that the

person has for a head-start. The former adjusts the sensitivity of feature selection when generating intermediate goals from the recognized distributions; the threshold $\tau \in [0, 1]$ requires a feature to appear in enough goal criteria that collectively represent at least τ of probabilistic RaP’s distribution over \mathcal{H} . The latter acts as a delay before the assistive agent begins responding, which allows it to have an observation sequence that is less ambiguous during recognition. Although parameter tuning is a common challenge in many algorithms, especially for machine learning performance, we identified some impacts for choosing different values.

If the necessities threshold is too low, then the intermediate goal adds more features unique to specific goal criteria. Although this sounds more robust to accommodate the uncertainty at the beginning of the interaction, the present-day norm of *conjunctive goal conditions* means that there is a greater opportunity for the goal to have contradictions. In our demonstration, at most one block can be on top of another. However, lower thresholds allowed words that shared one letter to require both of their preceding letters on top—“mother” and “father” could easily require both the A and O blocks to be placed on top of the T block (in addition to building the stack that spells “ther”) when τ is sufficiently small. The lack of a solution to this goal means that the interactive agent cannot find a plan and act that turn, which makes them appear less helpful to people. Likewise, if the necessities threshold is too high, then the necessities might not contain any features and the intermediate goal consequently changes nothing—“mother” and “father” may have a combined probability of 0.8 in some cases, but that does not identify any intermediate goal conditions when $\tau = 0.9$. In this case, the *solution of doing nothing has the same consequence as not finding a solution to a goal with contradicting conditions*.

The number of head-start turns can be more drastic. If it is low, such as 0 to begin interacting immediately, then the first few observations can be very ambiguous such that the assistive agent recognizes a near-uniform distribution over the subset of goal

criteria that use those actions at least once in their possible solutions. Even with a reasonable necessities threshold, this distribution can either be *spread too thin* to find no features for the intermediate goal or be *concentrated enough over almost-distinct goal criteria* that contradicting unique features are added to the intermediate goal. The latter scenario sometimes selected unique features that did not contradict others, but were *generally incorrect so that the interactive agent performed actions that did not make sense to people* interacting with them. An additional observation would often prune the extraneous goal criteria from the recognition algorithm’s output, which is why we added the head-start parameter. When it was set too high, though, the *person made enough progress alone that they found the interactive agent’s late response less useful*.

8.2 Challenge Topic: (Un)Intentional Communication

When the autonomous agent in human-computer/robot interactions has their own personal goals, they can communicate their intentions to people via legible planning with low-level motions [65, 196] or high-level actions [168]. However, our assistive agent’s personal goal is more abstract: “to help the user with their own goal.” So the interactive agent does not have a personal goal until they compute an intermediate one through recognition (see Section 5.5). Geib et al. [90] account for communicating the agent’s newfound goals during the negotiation step, but their assistive agent pipelines the interaction process so that there is no further recognition after negotiating tasks. We assumed that the *cognitive load of frequent negotiations would not be ideal* as the PRETCIL framework loops indefinitely.

However, our implementation’s planner assumes what people involved in the interaction will do, which sometimes reorders or includes extraneous actions compared to what each person actually plans to do. These *deviations between both agents’ expectations* could be enough to *confuse the person instead*. In one instance, someone

trying the demonstration looked at the debug data to read the sequence of assumed actions and mentioned that providing this information would have been a *useful explanation for the unexpected behavior*. Providing explanations for decision making systems [76] has been growing in popularity recently, but we need to be careful that these explanations do not constrain people’s freedom to act in accordance to what the machine does [43]. Social media platforms already force people desiring attention to be “slaves to the algorithm” [33, 124], and we do not wish to extend this trend to interactive experiences with intelligent systems.

Some people at the demonstration already succumbed to such constraints when selecting their own actions to ensure legibility to the recognition algorithm, viewing the demonstration as a puzzle rather than an open-ended interactive experience. Does this *defeat the purpose of closed-loop interaction* if people adjust their own behaviors *to satisfy the algorithms around them rather than act naturally?* Though we mentioned that Levine and Williams’s assistive agents [176] have more restricted interactions using a library of precomputed plans, this library often contains multiple plans that allow the interactive partner flexibility (this is the purpose behind their *choice nodes* where the human can take one of several actions). This leads to a research challenge for *finding the balance in a hybrid of closed-loop interaction frameworks*. If a joint-agent planner finds multiple plans to the intermediate goal, then which plans’ action should the agent use when there are multiple options? That is, when monitoring the execution, which plans are “going according to plan”?

In the opposite direction of behavior influence, it is also important to identify when people take actions for *signaling purposes* rather than goal satisfaction. In one instance at the demonstration, the interactive agent picked up a block that the person just put down on top of the goal stack because the necessities did not think it belonged. Specifically, R_{ing} predicted that R_{ed} wanted to spell ‘later’ or ‘water’ rather than ‘master’; implementing the dynamic prior (see Section 5.1) might have

avoided this issue because it is generally more optimal to spell shorter words. So the best action for R_{ing} to take after R_{ed} picked up the A block was picking up the recently-placed S block. The person calmly passed their turn with a no-op action, and the interactive agent then replaced the S block on top of the goal stack. From the perspective of probabilistic RaP, this means that the action of not putting the A block on top of the goal stack after the removal of the S block made spelling ‘later’ or ‘water’ less optimal than spelling ‘master’ or ‘faster’ (another option in the demonstration’s dictionary of possible goal words \mathcal{H}). Other people in this situation often put the A block down elsewhere rather than do nothing, and this usually began to confuse the recognition algorithm as described in Section 8.3 below because putting the A block down anywhere other than the goal stack was not optimal for any of the goal words in \mathcal{H} . Although people in both cases performed an action as a responsive signal (stating, “that is wrong” or “put it back, please”), the *interactive agent only interpreted one case correctly by mere coincidence of the underlying algorithm*.

8.3 Challenge Topic: What Information Actually Matters?

While most of the challenges discussed so far involve general issues that relate specifically to the interactive experience, it is also important to consider some algorithmic challenges. The most critical ones we identified during the demonstration relate to *using all the available information*. Some plan recognition algorithms already address noisy sensing [252] and irrelevant experimental actions while exploring the environment [195], but these methods still *assume that the observed agent R_{ed} is the only actor in the world*.

The *interactive agent R_{ing} ’s actions also change the world*, and these need to be acknowledged during recognition. We simply encoded them as observations because RaP algorithms handle missing observations by assuming that R_{ed} performs actions that can connect two consecutive observations. However, the potential for

poorly chosen intermediate goals threw off the recognition algorithms due to the R_{ing} 's sometimes incorrect actions and state modifications. Classifying these actions as noise or experimentation might work pragmatically, but they are conceptually different because *these actions have purpose and influence the interactive partner's later actions toward their goal*. Furthermore, for long-term interactive intelligent systems that people cannot reset as easily as our demonstration, how should they modify observation sequences over time for relevancy to the current interaction only?

When our demonstration's assistive agent computed a joint plan to solve their intermediate goal, the search algorithm used the same heuristics as probabilistic RaP's searches even though the state space and set of actions changed to address turn-taking. The above issues with incorrect intermediate goals present two things to consider with respect to this approach. First, when the intermediate goal contains contradicting conditions, is there a way to *find a plan that satisfies some largest possible subset of goal conditions* so that the agent can do something? Second, if the interactive agent is unable to find a plan, should they perform a default action (as we did in Section 7.2.2) or replan for some default set of goal conditions? We initially programmed our assistive agent to perform a no-op, but this led to several failed demonstrations where the human-controlled player needed a block that the computer-controlled player was holding before they failed to find a plan and started to execute no-ops. Even if some of these people intended to confuse the assistive agent with noisy observations, a default goal of not holding any blocks would at least allow people to complete the task on their own. We implemented this as a default plan because placing any block on the table cannot interfere with spelling a word, but the *goal variation is more robust in domains where there is no single plan that guarantees non-interference*.

8.4 Challenge Topic: Integrated Architectures of Algorithms

As domain-dependent heuristics can take advantage of specific information to improve search efficiency, customized code for integrating algorithms can take advantage of shared information more than simply gluing preexisting implementations together. While the PRETCIL framework is complex and allows a variety of algorithms to fill each component’s niche role(s), we hypothesize that implementing the framework as a *single architecture* can be more advantageous than pipelining the information between the algorithms separately. This was part of our inspiration for re-implementing search and probabilistic RaP algorithms in addition to the other reasons mentioned throughout this dissertation. For example, Figure 6.2 illustrates how all the algorithms that involve search share underlying data structures such as hash tables of states and actions—using pointers that reference a single hashed instance *avoids storing many duplicate object instances in memory between different search executions* within the same state space.

However, there are many additional opportunities to improve the design of our software library and potentially improve the performance of this PRETCIL framework implementation. One enticing feature of the original class of RaP algorithms was the ability to do most the work with off-the-shelf classical planning software, which reduced implementation efforts for recognition algorithms. However, more recent variations have *diverged from this reliance to create their own data structures and algorithms that take advantage of recognition-specific properties* that would never exist in an off-the-shelf classical planner [69, 217]. If we follow in their footsteps, then we can modify heuristic search in more ways than just searching for multiple goal states at once (see Section 6.5.1).

One such example involves *modification of the search algorithm and node implementation*. Searching through observation space requires augmenting the observation progress count to the original state space, which means *search typically encounters*

the same original state multiple times with a different progress count. This is not too worrisome memory-wise because we already hashed the original state, yet this novel state in observation space will generate the same original children states as the previous one with a modified observation count and increase the runtime for an *almost-redundant* search process. If we instead modify the search node to store a *bound on the observation progress* and *adjust these bounds per node throughout the search process*, then we can perform an ongoing heuristic search in the original state space (see Section 6.5.2)—this search process *only visits, expands, and goal-evaluates each state one time*. Another example, based on discussions with former student Roman Ganchin, would be *storing information from one search progression to speed-up the following ones*. MGHS’s marginal utility heuristic [59] uses this to identify how many goal nodes exist in a search subtree from a given node. Pattern databases [57] store the oracle heuristic values for subproblems between states, and we can fill in a table between searches (rather than precompute it all a-priori) so that future search processes can find the same goal more efficiently.

Beyond the integration of high-level recognition and planning into a single architecture, there is also the potential to *directly include low-level recognition* such as the activity recognition approaches in Chapter 4. The topic modeling-based algorithms associate sensor data to action labels via probability distributions, but is it possible to use the distribution more flexibly when matching observations in RaP? Rather than selecting the most-likely action label and then passing it to a RaP-like algorithm for independent analysis, the raw sensor data can provide a distribution over possible action labels as well as the approximately-recognized plan to guide the search process(es) solving \mathcal{P}_O^G and $\mathcal{P}_{\leq O}^G$. The challenge behind this approach is the additional uncertainty of whether a matched plan is truly a solution to either problem because the *observations themselves are probabilistic*.

8.5 Challenge Topic: Understanding People, Not Just AI

As of writing this dissertation, it personally seems to the author that many members of the AI research community under-appreciate the fact that people exist alongside intelligent systems. This in no way means that they disregard the existence of people or their greater competence than the typical intelligent system. Humans have served the role of experts to improve systems employing AI since the early days of the field—a few, but far from all, examples include:

- People analyzed case scenarios and proposed solutions during the development of expert systems [131],
- Apprenticeship learning/learning from demonstration [239] relies on observing how people do things and receiving their feedback, and
- Active learning [39] asks people for classification advice when there is uncertainty during training and/or testing.

Furthermore, human performance is often a goal for researchers studying some aspect of AI. The fact that researchers strive to develop computational agents that can effectively play games against people and win [105] is more than enough evidence that people are not being ignored.

However, acknowledging people for their intelligence is only considering *a fraction of their potential contributions to developing intelligent systems that co-exist in the world amongst them*. The term “human-aware AI” [143] became a recent buzzword in AI research that might address this issue, but also has a risk of averting it. At the moment, most (not all) the literature regarding human-aware AI seems to focus on one of several areas:

- **Personalized Algorithms** Applications using a machine learning algorithm collect data about the specific user and tune their training dataset accordingly. A popular example is recommender systems [230].

- **Explainability** The AI algorithm derives a human-acceptable justification for their actions [3] so that people can feel more comfortable around machines.
- **Cognitive Modeling** Intelligent systems observe an individual's or group's behavior and generate a model that can emulate the observed behavior. Recognition algorithms commonly fall into this area, especially when employing theory of mind [18].
- **Ethics** To avoid/lessen the risk of intelligent systems causing harm to people, AI researchers must take responsibility to design algorithms and domains in such a way that minimize adverse consequences. However, for situations that even people consider ‘gray areas’ [240], we also need to decide how intelligent systems will address them.

These AI approaches certainly acknowledge the fact that people exist around them, but *do the researchers creating these approaches acknowledge people any further than being the targets of human-aware AI algorithms?*

Rather than creating methods based on philosophical conjectures of what people do (providing data, asking for specific forms of justification, making certain classes of decisions, etc.), *there is a lot that AI researchers can learn from actually watching people do things and studying their co-existence with intelligent systems*. This idea is not novel; it has been studied throughout other fields of research including psychology, cognitive neuroscience, and human-computer/robot interaction. Rather than working independently such that *AI researchers human-agnostically create “human-aware AI”* and release it into the world for other researchers to study later, there is merit to working together so that the algorithms are *designed based on what people actually do* before dissemination. Furthermore, we can receive emotional and social feedback from people during these studies that inform researchers about *how they feel and what they do (not) like about the methods*.

Through Chapter 7, it has hopefully been clear that many conceptual ideas regarding how decision making for interaction worked did not properly play out as intended once people tried them out. Our experiments only started to explore people’s interactive experiences with closed-loop interactive intelligent systems, and there is much opportunity for co-studying how *both* machines and people make decisions about interacting with each other. As one example for future research, consider the look-ahead parameter ϵ and suboptimality parameter λ from our dynamic prior for probabilistic RaP in Section 5.1. We can sweep values and optimize the results for some contrived benchmarks, but why not study real people acting in relevant domains first? How suboptimal are people when they solve trivial problem instances compared to more difficult ones? How much foresight do people apply in various problem instances as time progresses?

Observing how people interact can benefit more than just recognition algorithms to guide decision making, but machines can only observe and interpret as instructed. A human researcher, though, is not restricted by such algorithmic code. They can observe additional information about people to further improve understanding and identify new research questions. While conducting the experiments in Chapter 7, the on-site researcher (who happens to be the author of this dissertation) noticed almost every participant remark surprise during the second tutorial level—people assumed the computer-controlled player was adversarial until they put the next block on top of the goal stack. Human-aware AI on its own would not realize the significance of this simple event, but we propose the following new research questions from it alone:

1. Why would a *person assume that the interaction was adversarial*? Does the nature of playing games affect their prior? Is it a cultural impact from science fiction’s common theme of adversarial intelligent systems?

2. How does an *interactive agent communicate its type of interaction to people?*

This is similar to legibility, but regarding an abstract task (helping, hindering, etc.) rather than a concretely-defined task (see Section 5.5).

3. How can an *interactive agent recognize another agent's interaction type?* That is, how can they tell if another agent is assisting, being adversarial, or simply acting on their own independent task?

Considering such questions provides evidence that the PRETCIL framework might not even have enough components to address such levels of reasoning yet. Much research remains, but it is clear that *machines interacting with people is not just the goal, but also an essential, iterative part of the research process.*

APPENDIX

FIRE EMBLEM HEROESTM IS NP-COMPLETE

A.1 Introduction

Fire Emblem HeroesTM (FEH) is a videogame for mobile devices released in February 2017 where two players strategically manipulate pieces with various statistics and special abilities in a manner similar to a hybrid of Chess and Magic the GatheringTM [211]. Unlike many classic strategy games where the collection of pieces are the same for each player and consistent between gameplay instances, FEH has a large collection of pieces (called “heroes” or “units”, but we will use the more general term for readers who are not familiar with the game’s vocabulary) from which players construct a team of pieces to use prior to each gameplay instance¹. Most matches take place on a six-by-eight-tile board (called a “map”, with some tiles containing obstacles that may possess their own attributes) and only allow teams of up to four pieces, but some modes use alternative board sizes or larger teams where new pieces are added to the board over time. In general, a player wins by successfully capturing all the opponent’s pieces from the board first; however, sometimes alternative criteria are presented such as offense and defense (a turn limit for offense to win, which is a turn limit for defense to avoid losing) and/or capturing special territory-marked tiles on the board.

Although FEH supports human players competing against each other, this currently extends to only selecting the team of pieces and occasionally the game board’s

¹A player’s personal collection of pieces will be some subset of all existing pieces in the game because they must be purchased via a gumball machine-like interface called “summoning”.

layout. These selections are uploaded to the game’s server and then downloaded to a client player’s device when the match begins. Thus the client player’s team of pieces is human-controlled (though there is an auto-battle feature that automates the client player), but the player’s team that is downloaded from the server is computer-controlled. Like most traditional videogames, the algorithm the computer follows for manipulating the team of pieces is an expert system. That is, there is a hand-coded list of instructions for what the computer should do based on the current board configuration. Some members of the FEH-playing community have invested time into reverse-engineering this expert system [282, 4], and it has also been concluded through strong evidence that the algorithm is purely deterministic. This drastically changes how the *gameplay* is perceived from an adversarial game to a puzzle game and guarantees replicable playthroughs if the board and pieces are identical [199], which allows players to share solutions that help guide others [222].

These properties are related to the Nondeterministic Polynomial (NP)-Complete class of problems that are verifiable in polynomial time (with respect to the problem size) and thus solvable in polynomial time if all possible solutions are guess-and-checked simultaneously. We show that the FEH decision problem, determining whether the human-controlled player can win within k turns given board configuration B , is in this complexity class after explaining the mechanics of the game.

A.2 Playing Fire Emblem HeroesTM Overview

Although FEH has much in common with other strategy games because players move pieces they control around the board and capture opposing pieces, the manner in which these steps are executed is not as traditional. In particular, the variety of statistics and special abilities play a large role in how to perform actions such as moving and capturing. We thus describe the features of pieces and then outline the general gameplay procedure.

A.2.1 Pieces and Their Features

In FEH, a piece represents a single character on the game board. A piece occupies a single tile, but has many properties assigned to it that vary its performance during gameplay. Some of these are constant and cannot be modified significantly through player investment:

Character Flavor Unique identifiers for a piece include its name, epithet, portrait images, and voice-acted sound clips. These do not affect gameplay.

Movement Type The number of adjacent tiles that a piece can traverse in a single movement and its obstacle restrictions. There are four types:

Infantry Traverses at most two tiles per movement; a forest obstacle tile counts as two tiles; and cannot traverse pit, water, or tall object obstacle tiles.

Cavalry Traverses at most three tiles per movement; a trench obstacle tile counts as three tiles; and cannot traverse forest, pit, water, or tall object obstacle tiles.

Flying Traverses at most two tiles per movement; and cannot traverse tall object obstacle tiles.

Armor Traverses at most one tile per movement; and cannot traverse pit, water, or tall object obstacle tiles.

Five Statistics Integer values that indicate a piece's proficiency at various aspects of the game. The sum of a piece's five integer values is generally equal to another piece with the same movement type (armor has the greatest sum while cavalry and flying have the least sum) and weapon type range (explained below, melee has a greater sum than ranged) if they were both introduced to FEH around the same time. To entice players to purchase newer pieces, this sum for each movement and weapon-type-range combination tends to increase over time;

however, alternative currencies awarded in the game allow players to increase these values to strengthen their older pieces. The five statistics are:

Hit Points (HP) The ‘stamina’ representing how well a piece can sustain before being captured. This number generally decreases after each capture attempt during a gameplay instance, and the piece is captured if its HP becomes zero or less.

Attack (Atk) The ‘strength’ representing how well a piece can capture other pieces. If a piece happens to have a special ability that makes it effective against the opposing piece that it tries to capture, then its attack value is multiplied by 1.5.

Speed (Spd) A piece attempts to capture twice in a single action if its speed value is at least five greater than the opposing piece that it tries to capture’s speed value.

Defense (Def) The ‘physical durability’ representing how well a piece can withstand capture attempts from pieces with physical weapon types (explained below).

Resistance (Res) The ‘magical durability’ representing how well a piece can withstand capture attempts from pieces with magical weapon types (explained below).

Weapon Type The type of weapon implies the range at which the piece can attempt to capture other pieces on the board as well as which statistics are used during the capture attempt. Melee weapon types may attempt to capture opposing pieces in an adjacent tile to any tile to which the piece can traverse. Ranged weapon types may attempt to capture opposing pieces exactly two adjacent tiles away from any tile to which the piece can traverse. If a piece attempts to capture an opposing piece with the same weapon type range, then the opposing piece

will attempt to capture it afterwards if the capture was not successful (called a “counter”). Weapon types are also either physical; including swords, lances, axes, beast attacks, daggers, and bows; or magical; including tomes, staves, and dragon attacks. Of these weapon types, only tomes, staves, daggers, and bows are ranged; the remaining weapon types are all melee. An additional special feature of the bow weapon type is that it is effective against pieces with the flying movement type. Pieces with the staff weapon type have access to special abilities that can recover another piece’s HP, but they are not allowed to use many offensive special abilities. Furthermore, pieces with the staff weapon type multiply their total damage by 0.5 when they attempt to capture another piece.

Weapon Color The color of a weapon is used to add a Rock-Paper-Scissors relationship between pieces called the “Weapon Triangle”. Of the weapon colors, red has an advantage over green, green has an advantage over blue, and blue has an advantage over red. There is also a colorless weapon color that has no advantages or disadvantages over other weapon colors unless changed by a special ability. Similar to how effectiveness works, a piece with a weapon color advantage over a piece that it is attempting to capture (or that is attempting to capture it) multiplies its attack by 1.2. On the other hand, a piece that has a weapon color disadvantage over a piece that it is attempting to capture (or that is attempting to capture it) multiplies its attack by 0.8. Of the weapon types, swords must be red, lances must be blue, and axes must be green, but all the other weapon types may be assigned any of the four weapon colors².

In contrast, a piece’s remaining features can be modified through a player’s investment of resources, including the payment of various currencies awarded in the

²As of the time of writing this, all pieces with the staff weapon type have the colorless weapon color. So it might be the case that staves cannot have any weapon color.

game or other pieces in the player’s inventory. Some of these simply modify the five statistics similar to a level-up mechanic (“merging”, “support”, and “dragon flowers”) or during specific modes of play (“blessings” whose benefits depend on the weekly “season” for arena modes). The rest are related to the special abilities that a piece possesses during gameplay (when there is more than one option for a single feature, the chosen one to use during the current game instance is “equipped”), and we will focus on these below:

Weapon For a piece’s weapon type, there are a variety of weapons available in the game with a might (Mt) statistic and some special abilities. The might statistic is an integer value that is added to the piece’s attack statistic (for computational purposes, FEH refers to this sum as the piece’s “attack”). The other special abilities either mimic a special ability from a slot skill (defined in a bullet below) or provide some unique ability. Many pieces also have an exclusive preferred weapon that can only that specific piece can equip. The list of actual abilities and preferred weapons is too exhaustive to list, and we will primarily use the generic weapons with no special abilities in the proof. The only non-generic weapon ability we will need is the ‘brave’ effect, which allows a piece to perform two consecutive capture attempts per initiated capture attempt (in exchange for lesser might than other weapons and a decrease in the piece’s speed statistic).

Assist Skill Besides attempting to capture the opponent player’s pieces, the player can instead have their own pieces interact with each other if the one moving has an assist skill and will be in the correct range to another piece on the same team. Pieces with the staff weapon type have their healing ability assigned as their assist skill, but other pieces can apply alternative helpful effects. These include increasing the values of some subset of another piece’s five statistics (‘rally’), exchanging some conditioned number of hit points (‘ardent sacrifice’ and ‘reciprocal aid’), moving itself and/or the other piece somewhere else on the

board despite the allowed number of spaces for the movement type ('reposition', 'shove', 'smite', 'draw back', 'swap', and 'pivot'—obstacles for each movement type still apply), and allowing a piece that already moved to move again ('dance' and 'sing' are limited to specific pieces in the game and these pieces cannot perform the assist skill on each other if they both have it equipped). A few pieces have exclusive assist skills as well. No assist skills will be used in the proof.

Special Skill Special skills are related to an ability that only occurs once in a while, using a countdown that 'charges' when some condition is met and then activating when the condition is met after completely charging. After activation, the countdown returns to its maximum value (called the 'cooldown count'). Whenever a piece is involved in a capture attempt, unless a special ability is involved, the charge decreases by 1. Additional events and special abilities can alter the cooldown count and charge rate, and some levels in the game (not player-vs.-player content) will charge the special skills of the pieces on the computer-controlled player's team before the round begins. The majority of the special skills increase the amount of damage that a piece performs during a capture attempt, but there are a few others that reduce the amount of damage a piece receives during a capture attempt, reduce the HP of the opponent's pieces outside of a capture attempt, heal HP during a capture attempt, grant another movement, etc. A few pieces have exclusive special skills as well. We will only use the special skill named 'miracle' in the proof, which negates a successful capture and sets the equipped piece's HP to 1 if (1) it is completely charged, (2) the piece equipped with this special skill has more than 1 HP before the opposing piece attempts to capture it, and (3) the opposing piece's capture attempt is successful after the damage calculation.

A/B/C/Sacred Seal-Slot Skills These are the primary special abilities assigned to a piece, and they have a wide variety of effects. In general, but not always the case, the skills allowed in each slot have a specific effect property. A-Slot skills typically provide a benefit to the piece with that skill equipped, B-slot skills typically alter some game rule such as capture-attempt-order or apply additional effects when certain events occur (using assist skills, being involved in a capture attempt, etc.), C-slot skills typically apply changes to nearby pieces on the gameboard (increasing/decreasing the values of the five statistics, adding additional effects, etc.), and sacred seal-slot skills can be any other slot skill so that two abilities pertaining to the same slot can be equipped simultaneously³. A few pieces have exclusive skills for a subset of their slots as well. The list of actual abilities per slot skill is far too exhaustive to list, but we will use the A-slot skill named ‘distant counter’ in the proof. This skill simply allows an equipped piece with melee weapon type to counter if a piece with ranged weapon type attempts to capture it.

A.2.2 Gameplay Flow

We will describe the general flow of gameplay, including how some specific events such as capture attempts play out for resolution. However, FEH has multiple phases for a variety of events that determine when special abilities activate. As none of these niche moments are needed for the proof, we will not discuss them to avoid overwhelming readers who are unfamiliar with FEH. We will only focus on the main mode of gameplay with the traditional victory criteria of capturing all of the opponent’s pieces first.

³The available sacred-slot skills are a subset of all the slot skills in the game, but is gradually expanding over time. They are usually awarded during gameplay and can only be assigned to one piece at a time.

A.2.2.1 Setup

The game contains a variety of boards that are six tiles horizontally by eight tiles vertically. Every board contains markers for where to place each type of obstacle tile as well as the pieces on both players' teams. These piece placement tiles enforce placement based on the player's ordering of their pieces in the team list; however, an updated version shortly after the game's release has since allowed the human-controlled player to rearrange their pieces' placement before the round begins⁴. The FEH problem's input encodes the turn limit k and the board after setup and the player rearranged their pieces (since the team list could have been reordered prior to setting up the board).

A.2.2.2 Turn Order and Actions

Although FEH is turn-based, the players' pair of turns together are referred to as a "turn". This is important for some special abilities that last 'until the end of the current turn.' A single turn is thus divided into two "phases". The human-controlled player always goes first during the "player phase", and then the computer-controlled player always goes second during the "enemy phase". A turn for each player is the duration of that player's current phase and the other player's next phase, which is effectively the time between that player beginning their consecutive phases.

During their respective phase, the player may perform up to one action with each piece on their team that is not captured⁵. This has some similarities to strategy

⁴This feature provides an advantage to the human-controlled player, but that is a game design-related discussion that is not relevant to this work. The advertised reason for the feature involves players no longer having to reset matches to alter their team ordering when playing non-player-vs.-player levels that require a time or financial payment per try, which is a common feature of mobile videogames.

⁵A very recent addition of "duo skills", which are exclusively to very few rare pieces called "Duo Heroes", allow a second action once per game instance that activates the skill. Some modes also allow "pair-up" actions to be taken anytime during a player's phase that swap a piece on the board with a corresponding piece off the board. Due to their late inclusion in the game's history and being augmentations to the original rules, we will omit these cases.

games such as Arimaa where a player takes multiple actions per turn, but the actions cannot be allocated between pieces freely (that is, a piece can only act once per turn rather than not move one piece in exchange for two actions with one piece). Like such strategy games, though, the order in which pieces act is up to the player so that the available actions per piece might change from the consequences of another piece’s actions.

A single action combines both moving and interacting with up to one other piece on the board, which depends on the piece’s movement type and range for weapon type (if interacting with an opponent’s piece or breakable obstacle) or assist skill (if interacting with another piece on the player’s team). A piece cannot finish moving into another tile that is occupied by any other piece. On the other hand, a piece may freely move through a tile with another piece from the same team (similar to letting an ally slide past). This benefit does not reciprocate with pieces on the opponent’s team, which serve as additional tall object obstacles to the ones placed during the board’s setup. When the interaction is an assist skill, then the acting piece’s assist skill’s effects simply resolve. Interacting with a breakable obstacle also simply reduces the obstacle’s hit points by 1, and the obstacle is removed when its hit points reach 0. Interacting with an opponent’s piece begins a capture attempt, which is formally part of a more complicated event called “combat”. We describe combat in the following section.

A.2.2.3 Combat Procedure

When two pieces on an opposing team interact, a combat begins and these pieces have opportunities to capture each other. Depending on the phase of the turn, one of the players “initiates” the combat by performing an action with their own piece R_{init} that interacts with an opponent’s piece R_{foe} . We will progress through combat assuming that no special abilities from any weapons or skills are active.

1. The piece that initiated the combat R_{init} attempts to capture the opposing piece R_{foe} . R_{foe} 's HP is decreased by the amount of damage it receives, which is the difference of R_{init} 's attack statistic minus R_{foe} 's defense statistic (if R_{init} 's weapon type is physical) or resistance statistic (if R_{init} 's weapon type is magical). If R_{foe} 's HP becomes 0 or less, then it is captured—the piece is removed from the board and combat ends.
2. If R_{foe} is not captured and it has the same weapon type range as R_{init} , then R_{foe} counters and attempts to capture R_{init} . R_{init} 's HP is decreased by the amount of damage it receives, which is the difference of R_{foe} 's attack statistic minus R_{init} 's defense statistic (if R_{foe} 's weapon type is physical) or resistance statistic (if R_{foe} 's weapon type is magical). If R_{init} 's HP becomes 0 or less, then it is captured—the piece is removed from the board and combat ends.
3. If R_{init} is not captured and its speed is 5 or greater than R_{foe} 's speed, then R_{init} “follows up” and attempts to capture R_{foe} again. R_{foe} 's HP is again decreased by the amount of damage it receives. If R_{foe} 's HP becomes 0 or less, then it is captured—the piece is removed from the board and combat ends.
4. If R_{foe} is not captured, has the same weapon type range as R_{init} , and its speed is 5 or greater than R_{init} 's speed, then R_{foe} “follows up” and attempts to capture R_{init} again. R_{init} 's HP is again decreased by the amount of damage it receives. If R_{init} 's HP becomes 0 or less, then it is captured—the piece is removed from the board and combat ends.
5. If neither R_{init} nor R_{foe} is captured yet, then combat ends and both pieces remain on the board with their updated HP values.

Any time one piece attempts to capture the other during combat, both pieces decrease their equipped special skill's countdown by 1 to charge it. For the special

skill ‘miracle’ that we will use in the proof, its maximum cooldown count of 5 means that the piece equipping it must be involved in 5 capture attempts as either R_{init} or R_{foe} before it can activate. To avoid it not being ready in time, we will take advantage of the fact that the computer-controlled player’s pieces exclusively equip it and that FEH has some levels where the computer-controlled player’s pieces have their specials fully charged before the round begins.

Though this is the common progression of combat, the reader should be aware that the order can change and additional events can frequently occur during a single combat based on other pieces’ special abilities (besides R_{init} and R_{foe} themselves, nearby pieces can affect them during combat). We will not have any skills with external effects, but we will equip one piece on the human-controlled player’s team with a weapon that has the ‘brave’ effect. This effect alters the progression above such that, when the piece is R_{init} , steps 1. and 3. repeat immediately before steps 2. and 4. respectively.

A.3 k -FEH Problem

Given turn limit k and a board B following its setup, we define FEH as a decision problem that returns \top if and only if the human-controlled player can win in k or fewer turns. However, if we simply limit the format of B and its setup as described in Section A.2.2.1, then the FEH problem does not scale between problem instances because the *board size is constant*. This also bounds the maximum number of pieces on the board because at most one piece can occupy a tile. Thus, even if the program would take a long time to run, these FEH instances can be solved in constant $O(1)$ time with respect to the set of possible values of B by enumerating all possible move combinations that might exist on every tile at each of the k turns.

We therefore generalize FEH to allow scaling the size of B as input to the problem instance: a board can have sizes of n tiles horizontally by n tiles vertically with any

arbitrary assignment of obstacles and players' pieces to the tiles that is valid (for example, an infantry piece cannot be placed on a tile with a water obstacle). Our generalization thus allows either player to have an arbitrary number of pieces on their team (but at least one), and these team sizes do not have to be identical. Furthermore, we do not restrict the players to the set of pieces that have been released so far in FEH—any possible tuple of non-negative integers may be assigned to the piece's five statistics, and any combination of movement type, weapon type, and weapon color is permissible. Due to skills not being as trivial to generalize as a linear combination of integers or cross product of enumerable sets, we still enforce them to be the set of skills currently released in FEH belonging to each respective category.

Definition 57. *The FEH problem is a decision problem to determine whether, for integer k and generalized FEH board B , there exists a sequence of actions such that the human-controlled player can capture all the computer-controlled player's pieces in k turns or less. That is, can the human-controlled player win a game within k turns starting with B 's configuration?*

For the proof, we will specifically use only skills that were available upon the game's release to confirm that FEH was always NP-complete. This represents the original boards in FEH because we can pad extraneous rows and columns of the board with tall object obstacle tiles over which no movement type can traverse.

With this generalization to varying board sizes, we now investigate how the process of solving the FEH problem scales with respect to n . Following a similar intuition to the original FEH game, the set of possible solutions for a single value of n is finite due to the constant turn limit k that is also provided as input. Enumerations are still *bounded to a polynomial quantity* with respect to n , though. Because no two pieces can occupy the same tile, there are at most n^2 pieces on the board and at most $n^2 - 1$ of them belong to the human-controlled player. In general, this means that the human-controlled player will take at most $n^2 - 1$ actions per turn. However, due to

the allowance of additional actions by some assist skills (e.g. ‘dance’ and ‘sing’) and special skills (e.g. ‘galeforce’), a single piece might be able to take some additional actions. Due to the game’s rule’s restrictions that

- the special skills that grant an extra action may only be activated once per piece per turn,
- assist skills that grant an extra action to another piece cannot be applied to pieces that have such skills equipped, and
- assist skills that grant an extra action to another piece can be applied to a piece even if it was already affected by such an assist skill earlier in the same turn;

we can bound the number of actions that a single piece can take on a single turn to at most n^2 . This includes the piece’s typical action, additional action from its equipped special skill, and all the remaining $n^2 - 2$ pieces having an assist skill that grants another action choosing to interact with this piece. This means that the player has at most $(n^2 - 1) \cdot n^2 = n^4 - n^2 < n^4$ actions per turn regardless of B and the pieces assigned to the player. Due to k being constant, this ultimately means that a solution for the human-controlled player to win given B is bounded by kn^4 actions. This is $O(n^4)$, which is a polynomial number of constant-time actions with respect to n , the size of B .

A.4 FEH is NP-Complete Proof

This proof is broken into several smaller proofs. In particular, we will show that:

Section A.4.1 *A program whose runtime is bounded by a polynomial with variable n can verify a proposed solution to a given FEH problem.* This part shows that the FEH problem is “simple enough” to check a solution within polynomial time. However, this only provides an *upper bound* on the complexity because we could

possibly generate a solution using an algorithm that is as complex or simpler than the verification program. The worst case, which is the case of problems in the NP-complete complexity class, is a guess-and-check approach that runs the polynomial runtime verification program in parallel over all possible solutions simultaneously. This is non-deterministic polynomial runtime.

Section A.4.2 The known NP-complete problem *Subset Sum* [88] is polynomial-time reducible to FEH. This means that any Subset Sum problem can be converted into some FEH problem such that the solution to that specific FEH problem can also be converted into a solution to that specific Subset Sum problem. Due to this conversion being done in polynomial runtime, the process of solving the specific FEH problem to find the solution to the specific Subset Sum problem takes no more than nondeterministic polynomial runtime. Thus if FEH has a simpler way to generate its solution than the guess-and-check worst-case approach, then there is now a simpler way to solve the Subset Sum problem that no longer involves guess-and-check with a polynomial-bound runtime verification algorithm. Such a statement is a contradiction because the Subset Sum problem is already proven to be NP-complete. This establishes the *lower bound* on the complexity because the FEH problem cannot be “so simple” that it inherently solves the Subset Sum problem more easily than the Subset Sum problem can solve itself.

These two parts essentially act like a computational complexity version of the Squeeze Theorem, bounding FEH’s complexity class from both directions as NP-complete.

A.4.1 Polynomial-Time Verification

Let B be some arbitrary generalized FEH board of size n tiles horizontally by n tiles vertically, and let k be some arbitrary positive integer. Then given a sequence π of at most $k(n^4 + 1)$ actions (including “end turn” actions), we can verify in polynomial

time with respect to n whether π solves B such that the human-controlled player wins in at most k turns.

Proof: Given B , k , and π as inputs, we construct the verification program found in Algorithm 3. In general, the verification program is simply an emulator that runs FEH with B and simulates the human-controlled player’s actions with each action sequentially specified in π . If any action in π is invalid and cannot be performed at that time (the piece cannot move or interact as instructed, a specified piece in the action is already captured, etc.), then the solution is invalid and the verification program returns \perp . This also handles the case where π provides too many actions on a given turn because there will not be enough pieces to move, rendering one of the later actions invalid. Likewise, if k turns lapse and the human-controlled player did not yet win, then the verification program returns \perp . The case where the human-controlled player loses is a combination of these two situations. Either π continues to specify actions that the human-controlled player cannot perform (and are thus invalid) because all their pieces are captured, or the human-controlled player loses their last piece on the enemy phase of the last turn so that the computer-controlled player still has at least one piece on the board after the end of the k^{th} turn. Thus the verification program only returns \top when every action in π is valid, the computer-controlled player has no more pieces on the board, and the turn number is k or less. This means that π may contain extraneous valid actions after the human-controlled player wins; we can trivially remove these by adding an additional victory-condition check step after performing each action and returning \perp when this condition is met while π is not finished, but we choose to consider these solutions acceptable because the human-controlled player still wins within the turn-limit and does not perform any illegal actions. Hence Algorithm 3 performs as expected.

Because k is a constant, the outermost loop runs a constant number of times and is a scalar multiple to the runtime. The innermost loop runs until π specifies the

Input: Generalized FEH Board B , Turn Limit k , and Proposed Action Sequence π

Output: \top or \perp

```

1 set  $n$  to  $\sqrt{\text{tileCount}(B)}$ ;
2 set  $\text{action\_index} = 0$ ;
3 for  $turn$  from 1 to  $k$  by 1 do
4   // $next\_action$  is set to NIL if the index is out of bounds
5   set  $next\_action$  to the  $\text{action\_index}^{\text{th}}$  element of  $\pi$ ;
6   increment  $\text{action\_index}$  by 1;
7   //Simulate the player phase with  $\pi$ 
8   while  $next\_action \neq \text{"end turn"}$  and  $next\_action \neq \text{NIL}$  do
9     if  $\text{validAction}(next\_action, B)$  then
10       update  $B$  by performing  $next\_action$ ;
11       set  $next\_action$  to the  $\text{action\_index}^{\text{th}}$  element of  $\pi$ ;
12       increment  $\text{action\_index}$  by 1;
13     end
14     else
15       | return  $\perp$ 
16     end
17   end
18   //Simulate the enemy phase with  $\xi$ , which is a sequence of actions
19   set  $\xi$  to  $\text{enemyPhase}(B)$ ;
20   foreach action  $a$  in  $\xi$  do
21     | update  $B$  by performing  $a$ ;
22   end
23   end
24   // $k$  turns are over, and  $\pi$  should be exhausted
25   set  $next\_action$  to the  $\text{action\_index}^{\text{th}}$  element of  $\pi$ ;
26   if  $next\_action \neq \text{NIL}$  or  $\text{numberEnemies}(B) > 0$  then
27     | return  $\perp$ 
28   end
29   else
30     | return  $\top$ 
31   end

```

Algorithm 3: Verification of a Proposed Solution for the FEH Problem

“end turn” action or π is completely traversed, whichever occurs first. Exhausting π before all k turns are complete thus terminates the innermost loop for the remaining turns. The sum of all the iterations of the innermost loop across the k iterations of the outermost loop is $|\pi| \leq k(n^4 + 1)$, which is polynomial runtime in $O(n^4)$.

This means that the black-box function `enemyPhase(B)` is the only part of Algorithm 3 whose runtime needs to be determined. This step is literally running FEH’s enemy phase code to determine what actions to take given the current board state after the human-controlled player ends their turn. This is where the fact that FEH operates the computer-controlled player under a hand-coded expert system becomes important. The order of moving pieces and their actions are guaranteed by known sets of conditions including prioritizing the performance of ‘rally’ assist skills (increasing the value of some subset of another piece on the same team’s five statistics), attempting to capture the human-controlled player’s pieces whenever possible⁶, etc. Although this expert system is not revealed to players, members of the FEH-playing community have heavily studied and reverse-engineered it (if not completely, then mostly) [282, 4]. It is clear from these investigations that the number of tiles, which can also affect the number of pieces, does not have a significant effect on the computer-controlled player’s action choices. That is, without any exponential branching factor to consider ordering of piece actions and options for which actions to perform per piece, `enemyPhase(B)` is effectively polynomial-bound runtime over the number of pieces and thus polynomial-bound runtime with respect to n .

Therefore, Algorithm 3 also has polynomial-bounded runtime with respect to n in the worst case. \square

⁶This does mean that a computer-controlled piece will “commit suicide” and let itself be captured if that will be the consequence of the only combat it can initiate. The program does not allow the piece to “run away” unless it does not have a weapon equipped, which prevents it from attempting to capture any pieces in the first place.

A.4.2 Subset Sum is Polynomial-Time Reducible to FEH

For those who are unfamiliar with the Subset Sum problem, we briefly define it here.

Definition 58. *The Subset Sum problem is a decision problem to determine whether, for input set of nonnegative integers S and nonnegative integer z , there exists a subset $T \subseteq S$ such that $\sum_{t \in T} t = z$. The Subset Sum problem is NP-complete, where the factor of scaling is along the cardinality of the input set $|S|$.*

We now show that Subset Sum \leq_m FEH, which is done in three parts. First, we show that there exists a polynomial-time algorithm M that converts any problem in Subset Sum to another problem in FEH. Second, we prove that the FEH problems to which we convert each Subset Sum problems share solutions. That is, for all problems $(S, z) \in$ Subset Sum, (S, z) evaluates to \top if and only if $M(S, z)$ evaluates to \top . As an if-and-only-if-statement, we must prove each conditional statement/direction.

A.4.2.1 Existence of Polynomial-Time Algorithm

We begin with the construction of our FEH *gadget*, which defines the structure of the generalized FEH boards that represent corresponding Subset Sum problems. To conform with the combinations of features available in the original release of FEH (to show that it was always NP-complete), we will use pieces with infantry movement types in our gadget because infantry was the only movement type that was allowed to equip bow weapon types in the original set of released pieces. It consists of three types of layers, each shown in Figure A.1:

Enemy Layer The computer-controlled player is given two pieces that are *unable to take any actions* because they are surrounded by tall object obstacle tiles on all sides. They are also given melee weapon types so that they cannot act by attempting to capture any of the human-controlled player's pieces when they are

on the opposite side of the obstacle tiles. Both pieces equip the sword⁷ weapon ‘iron sword’ (6 might and no special abilities), special skill ‘miracle’ that is already fully charged before the round begins, and A-slot skill ‘distant counter’. For the five statistics, both pieces have 0 attack, 0 speed, 5 defense (which is the might of the ‘brave bow’ weapon in FEH), and 4 resistance (which is the might of the tome⁷ weapon type with the weakest might and no special abilities in FEH). The pieces do differ in the HP statistic value, though. Without loss of generality, we assign the piece on the left $(z + 2)$ HP and the piece on the right $((\sum_{s \in S} s) - z + 2)$ HP.

Set Element Layer The human-controlled player is given one piece in a constrained corridor that has tall object obstacle tiles on both sides. This constrains the movement of this piece along the corridor, and it is given a ranged weapon type so that it can attempt to capture the computer-controlled player’s pieces when it is on the opposite side of the obstacle tiles in the Enemy Layer. The piece in this layer only equips the red-tome⁷ weapon ‘flux’ (4 might and no special abilities) with no additional skills. For the five statistics, the piece has s attack for some $s \in S$, 0 speed, 0 defense, 0 resistance, and 1 HP.

Player Layer The human-controlled player is given one piece in a constrained corridor that has tall object obstacle tiles on both sides. This piece is also given a ranged weapon type so that it can attempt to capture the computer-controlled player’s pieces when it is on the opposite side of the obstacle tiles in the Enemy Layer. The piece in this layer only equips the colorless-bow weapon type ‘brave bow’ (5 might and the ‘brave’ special ability that reduces the piece’s speed

⁷We choose the sword weapon type without loss of generality from lances and axes, which only differ with respect to weapon color in this gadget. If one of the other weapon types are chosen, then the tome weapon type of the respective color type should also be chosen in order to avoid the attack statistic multiplier that results from advantages within the rock-paper-scissors mechanic.

statistic value by 5 and allows it to perform two consecutive capture attempts per standard capture attempt during combats that it initiates) with no additional skills. For the five statistics, the piece has 1 attack, 0 speed, 0 defense, 0 resistance, and 1 HP. We note that the bow weapon type was exclusively colorless when FEH first released, but still clarify the weapon color to avoid any confusion.

To construct the gadget, we *sandwich Set Element Layers, one per element of S , between the Enemy Layer and the Player Layer*. This ultimately creates a board of size 4 tiles horizontally by $2|S| + 4$ tiles vertically, and then we pad the board with $2|S|$ columns of tall object obstacle tiles in order to create a square-shaped board. The scalar value 2 is the vertical size of the Set Element Layer in order to evenly space out the pieces with respect to their infantry movement type. As the pieces are placed on the board to complete the gadget's setup, the piece in the Player Layer does not depend on S or z , the piece in each Set Element Layer depends on S such that each layer's piece corresponds to a unique element in S to set its attack statistic value, and the pieces in the Enemy Layer depend on both S and z in order to compute their HP statistic values. We present an algorithm with quadratic runtime $M \in O(|S|^2)$ in Algorithm 4 that constructs this gadget, which confirms that we can encode a Subset Sum problem as a FEH problem in polynomial time. \square

A.4.2.2 Solution to (S, z) Implies Solution to $M(S, z)$

We now show that for all problems $(S, z) \in \text{Subset Sum}$, if (S, z) evaluates to \top , then $M(S, z) \in \text{FEH}$ evaluates to \top .

Proof: Suppose that some arbitrary $(S, z) \in \text{Subset Sum}$ returns \top . Then this means there exists some $T \subseteq S$ such that $\sum_{t \in T} t = z$. Due to the construction of the gadget $M(S, z)$, the human-controlled player has some subset of their $(|S| + 1)$ pieces whose attack statistic values correspond to each element of T . Therefore, we

(a)

C_0	TOO	C_1	TOO...
TOO	TOO	TOO	TOO...
			TOO...

C_0	Sword (Red, Melee, Physical)	Infantry
We: Iron Sword (Mt: 6)	As: -	Sp: Miracle
A: Distant Counter	B: -	C: -
SS: -	$z + 2 \text{ HP}$	0 Atk
0 Spd	5 Def	4 Res

C_1	Sword (Red, Melee, Physical)	Infantry
We: Iron Sword (Mt: 6)	As: -	Sp: Miracle
A: Distant Counter	B: -	C: -
SS: -	$(\sum_{s \in S} s) - z + 2 \text{ HP}$	0 Atk
0 Spd	5 Def	4 Res

(b)

TOO	H_i	TOO	TOO...
TOO		TOO	TOO...

H_i	Red-Tome (Red, Ranged, Magical)	Infantry
We: Flux (Mt: 4)	As: -	Sp: -
A: -	B: -	C: -
SS: -	1 HP	$s_i \in S$ Atk
0 Spd	0 Def	0 Res

(c)

TOO	H	TOO	TOO...
TOO		TOO	TOO...

H	Colorless-Bow (Colorless, Ranged, Physical)	Infantry
We: Brave Bow (Mt: 5)	As: -	Sp: -
A: -	B: -	C: -
SS: -	1 HP	1 Atk
0 Spd	0 Def	0 Res

Figure A.1. The three types of layers used to construct a FEH gadget given some S and z . The board joins these layers along the horizontal edges, with one (a) Enemy Layer on the top, one (b) Set Element Layer per $s \in S$ in the middle, and one (c) Player Layer on the bottom. This sets the vertical tile length of the board, and the horizontal tile length of the board pads columns of tall object obstacle (TOO) tiles to make a square board shape.

Input: Set of Nonnegative Integers S and Integer z
Output: Generalized FEH Board B and Integer k

```

1 set size of array surface to  $(2|S| + 4)$  rows and  $(2|S| + 4)$  columns;
2 fill surface with tall object obstacle tiles;
3 initialize list pieces;
   //First, place the Player Layer in the back row
4 set element of surface at row  $(2|S| + 3)$  and column 1 to empty tile;
5 insert new Piece( $2|S| + 3, 1$ , human, colorless bow, infantry, brave bow,  

   NIL, NIL, NIL, NIL, NIL, NIL, 1, 1, 0, 0, 0) into pieces;
   //Next, append each Set Element Layer and keep track of the sum of  $S$ 
6 set sum to 0;
7 set set_index to  $2|S| + 1$ ;
8 foreach  $s$  in  $S$  do
9   increment sum by  $s$ ;
10  set element of surface at row  $(set\_index + 1)$  and column 1 to empty tile;
11  set element of surface at row set_index and column 1 to empty tile;
12  insert new Piece(set_index, 1, human, red tome, infantry, flux,  

   NIL, NIL, NIL, NIL, NIL, NIL, 1,  $s$ , 0, 0, 0) into pieces;
13  decrement set_index by 2;
end
   //Last, place the Enemy Layer in the front rows
14 set element of surface at row 2 and column 0 to empty tile;
15 set element of surface at row 2 and column 1 to empty tile;
16 set element of surface at row 2 and column 2 to empty tile;
17 set element of surface at row 0 and column 0 to empty tile;
18 set element of surface at row 0 and column 2 to empty tile;
19 insert new
   Piece(0, 0, computer, sword, infantry, iron sword, NIL, Miracle-Charged,  

   Distant Counter, NIL, NIL, NIL,  $z + 2, 0, 0, 5, 4$ ) into pieces;
20 insert new
   Piece(0, 2, computer, sword, infantry, iron sword, NIL, Miracle-Charged,  

   Distant Counter, NIL, NIL, NIL,  $sum - z + 2, 0, 0, 5, 4$ ) into pieces;
21 return (new Board(surface, pieces),  $|S| + 2$ )

```

Algorithm 4: Gadget Constructor M

construct the following sequence of actions for the human-controlled player to take each turn:

1. If the piece H_i in the Set Element Layer nearest the Enemy Layer has an attack statistic value that is in T , then H_i attempts to capture the computer-controlled player's piece C_0 on the left in the Enemy Layer. Otherwise, H_i attempts to capture the computer-controlled player's piece C_1 on the right in the Enemy Layer.
2. All the remaining human-controlled player's pieces are moved two spaces towards the Enemy Layer. This uses all the human-controlled player's pieces and thus ends their turn.
3. If the human-controlled player has more than one piece remaining at the start of the turn, then return to step 1.. Otherwise, only the piece H from the Player Layer should remain on the board for the human-controlled player to use—proceed to step 4..
4. Piece H attempts to capture either C_0 or C_1 . This uses all the human-controlled player's pieces and thus ends their turn.
5. Piece H attempts to capture whichever of C_0 or C_1 remains on the board.

Whenever some H_i (equipped with the ‘flux’ weapon) attempts to capture either C_0 or C_1 , H_i initiates the combat so that it plays out as follows:

1. C .'s HP is decreased by the amount of damage it receives, which is

$$H_i.\text{Mt} + H_i.\text{Atk} - C.\text{Res} = 4 + s_i - 4 = s_i. \quad (\text{A.1})$$

That is, C receives damage equal to the value of H_i 's corresponding element in S . If C .'s HP becomes 0 or less and it previously had 1 HP, then it is

captured—the piece is removed from the board and combat ends. Otherwise, if C .’s HP becomes 0 or less and it previously had more than 1 HP, then its ‘miracle’ special skill activates so that C .’s HP is set to 1 and it is not captured.

2. If C . is not captured, then it counters and attempts to capture H_i . Although they do not have the same weapon type range, C .’s A-slot skill ‘distant counter’ allows it to counter anyways. H_i .’s HP is decreased by the amount of damage it receives, which is

$$C.\text{Mt} + C.\text{Atk} - H_i.\text{Def} = 6 + 0 - 0 = 6. \quad (\text{A.2})$$

H_i .’s HP of 1 clearly becomes 0 or less so that it is captured—the piece is removed from the board and combat ends.

Thus combat with some piece H_i effectively *allocates its corresponding value in S to one of two partitions and then removes that value from S* because the piece is no longer able to participate in combat. The computer-controlled player’s two pieces represent these two subsets based on their HP statistic values: C_0 is the subset of elements in S whose sum should be z and C_1 is the complement with the remaining elements of S that do not contribute to the sum. The damage decreases the HP (target sum) by the element in S to portray the allocation. The spacing between the pieces ensures that only one element of S is able to initiate combat at a time, which is why the pieces are then moved forward like a conveyor belt. As mentioned during the gadget’s construction, the *computer-controlled player is unable to take any actions during the game* at all⁸ because its pieces are surrounded by obstacles so that

⁸These games might sound boring because one of the players cannot do anything for the entire game, but they are legitimate instances of the generalized FEH game. It is unlikely that such boards will ever appear in the actual game because the game designers are aware that such boards are not as much fun to play. However, the Aether Raids mode allows players to create their own maps, and some players use this as a strategy to force the human-controlled player to approach their pieces and

(1) they cannot move anywhere on the board and (2) they cannot attack with their weapon type range.

Based on the constructed sequence of actions above, we conclude that C_0 received a total damage of z from the H_i pieces because its HP was reduced by each element of T . Consequently, C_1 received a total damage of $\sum_{s \in S} -z$ from the H_i pieces because its HP was reduced by the elements in the subset $S - T$. However, due to our gadget's design, this means that neither C_0 nor C_1 is captured yet with 2 HP remaining each. At this point, the human-controlled player took $|S|$ turns (that is, 2 turns remain according to $k = |S| + 2$) and only has piece H remaining. Whenever H (equipped with the ‘brave bow’ weapon) attempts to capture either C_0 or C_1 , H initiates the combat so that it plays out as follows:

1. C .’s HP is decreased by the amount of damage it receives, which is

$$H.\text{Mt} + H.\text{Atk} - C.\text{Def} = 5 + 1 - 5 = 1. \quad (\text{A.3})$$

C .’s HP thus decreases from 2 to 1.

2. Due to the special ability of ‘brave bow’, H is allowed to attempt to capture C . again because H initiated the combat. So C .’s HP is again decreased by 1, which means it is now 0. Because it previously had 1 HP, C .’s ‘miracle’ special ability does not activate and it is captured—the piece is removed from the board and combat ends.

Thus H ’s purpose is simply to “clean up” the computer-controlled player’s pieces and remove them from the board after taking advantage of the solution to the respective

fall into a variety of traps. Ironically, some traps involve equipping ‘distant counter’ and similar skills to their pieces that will likely survive when the human-controlled player initiates combat. Perhaps they are unintentionally creating these gadgets?

Subset Sum problem. The placement of the computer-controlled player's pieces allows H to capture them in this way on the next two turns in either order.

Therefore, we have a guaranteed solution to the FEH problem $M(S, z)$ that takes exactly $k = |S| + 2$ turns to execute. \square

A.4.2.3 Solution to $M(S, z)$ Implies Solution to (S, z)

We now show that for all problems $(S, z) \in \text{Subset Sum}$, if $M(S, z) \in \text{FEH}$ evaluates to \top , then (S, z) evaluates to \top .

Proof: Unfortunately, reversing the reasoning of the proof in Section A.4.2.2 is not sufficient to prove this claim. It is possible that an alternative solution exists to the FEH problem with $M(S, z)$, and it is further possible that this alternative solution does not translate to the allocation technique applied above so that it fails to solve (S, z) . Therefore, we provide a *proof by contradiction*.

Suppose there exists some (S, z) with no solution to the Subset Sum problem, yet there is a solution to $M(S, z)$ satisfying the FEH problem. Then there is a sequence of actions π that removes both the computer-controlled player's pieces C_0 and C_1 from the board B within $k = |S| + 2$ turns.

If π takes fewer than k turns to execute, then we will quickly realize that this is impossible because the pieces are spaced far enough apart that only one piece can initiate a combat per turn as the other pieces progress through the corridor towards the Enemy Layer. Specifically, every piece belonging to the human-controlled player can attempt to capture C_0 or C_1 at most one time. Based on Equations A.1 and A.3, we can conclude that the total possible damage these pieces can deal within $|S| + 1$ turns is $(\sum_{s \in S} s) + 2$. This involves each H_i and H attempting to capture once. However, the sum of C_0 and C_1 's HP statistic values is $(\sum_{s \in S} s) + 4$; thus it is not possible to deal enough damage to capture both pieces without using all k turns. We

have a contradiction because *there is no solution to $M(S, z)$ that satisfies the FEH problem.*

If π takes exactly k turns to execute, then we at least know that a solution can exist for $M(S, z)$ based on Section A.4.2.2. However, it cannot be of the same form as that solution because it inherently implies that there is a solution to the Subset Sum problem with inputs (S, z) . Due to the gadget’s design, it is possible with enough turns to permute the pieces in the corridor in a manner similar to bubblesort. However, because sets are order-agnostic, S technically yields any permutation of Set Element Layers within the gadget when running M depending on the machine’s implementation of sets. Thus, we extract elements from S as needed for each H_i without loss of generality—we must still use H last because it is always in the Player Layer of the gadget. Let us call this sequence $R = [r_1, r_2, \dots, r_{|S|}]$ such that each $r_i \in S$ and $r_i = r_j$ if and only if $i = j$.

We must identify some R corresponding to the order of H_i such that, when followed by H , both C_0 and C_1 are successfully captured. Let us first target C_0 without loss of generality until it is captured. As the human-controlled player’s pieces initiate combat at the start of the round, they perform the respective damage to their target piece and are then removed from the board due to the ‘distant counter’ skill. Eventually, C_0 will be reduced to a low enough HP statistic value that some piece H_x can perform its corresponding $s_x = r_x$ damage where $s_x \geq C_0.\text{HP}$.

In the case where this holds *and* $C_0.\text{HP} = 1$, the combat will play out as follows:

1. C_0 ’s HP is decreased by the amount of damage it receives, which is s_x by Equation A.1. Because C_0 ’s HP is no greater than s_x , C_0 ’s HP becomes 0 or less. Because it previously had exactly 1 HP, its ‘miracle’ special skill does not activate and it is captured. The piece is removed from the board and combat ends.

That is, the attempted capture succeeds and the H_x piece remains on the board. Thus C_0 actually required at least $(\sum_{i=1}^{x-1} r_i) + 1 = z + 2$ damage in order to be captured. Rearranging the terms of this equality, we get the following equality that we will need in a moment:

$$\left(\sum_{i=1}^{x-1} r_i \right) - z = 1. \quad (\text{A.4})$$

So the remaining pieces can target C_1 and perform up to $\left(\sum_{i=x}^{|S|} r_i \right) + 2$ damage (the additional two damage comes from piece H). However, subtracting this from C_1 's total HP yields

$$\left[\left(\sum_{s \in S} s \right) - z + 2 \right] - \left[\left(\sum_{i=x}^{|S|} r_i \right) + 2 \right] = \left(\sum_{i=1}^{x-1} r_i \right) - z = 1$$

using Equation A.4. That is, C_1 is not successfully captured because it still has positive HP after capturing all the remaining pieces via counter during combat. Even with the ‘brave’ effect associated with H ’s equipped weapon, it will not capture C_1 and instead become captured. Thus this case yields a contradiction because $M(S, z)$ does not have such a solution. We note that this is equivalent to the case where C_0 is left alone with its 1 HP and all the remaining pieces (H_x through $H_{|S|}$ and H) first attempt to capture C_1 because the same pieces take the same actions in some other order (except for which piece might capture C_0 before attempting to capture C_1 , but these all have identical outcomes).

In the case where $s_x \geq C_0.\text{HP}$ holds and $C_0.\text{HP} > 1$, the combat will play out as follows:

1. C_0 ’s HP is decreased by the amount of damage it receives, which is s_x by Equation A.1. Because C_0 ’s HP is no greater than s_x , C_0 ’s HP becomes 0 or less. However, it previously had more than 1 HP so that its ‘miracle’ special skill activates. C_0 ’s HP is set to 1 and it is not captured.

- Because C_0 is not captured, it counters and attempts to capture H_x due to its ‘distant counter’ A-slot skill. H_x ’s HP is decreased by the amount of damage it receives, which is 6 by Equation A.2. H_x ’s HP of 1 clearly becomes 0 or less so that it is captured—the piece is removed from the board and combat ends.

That is, the attempted capture will fail and the H_x piece is still removed from the board. However, the next piece H_{x+1} will successfully capture C_0 and not be removed from the board. Thus C_0 actually required $(\sum_{i=1}^x r_i) + 1 > z + 2$ damage in order to be captured. Rearranging the terms of this inequality, we get the following inequality that we will need in a moment:

$$\left(\sum_{i=1}^x r_i \right) - z > 1. \quad (\text{A.5})$$

So the remaining pieces can target C_1 and perform up to $\left(\sum_{i=x+1}^{|S|} r_i \right) + 2$ damage (the additional two damage comes from piece H). However, subtracting this from C_1 ’s total HP yields

$$\left[\left(\sum_{s \in S} s \right) - z + 2 \right] - \left[\left(\sum_{i=x+1}^{|S|} r_i \right) + 2 \right] = \left(\sum_{i=1}^x r_i \right) - z > 1$$

using Equation A.5. That is, C_1 is not successfully captured because it still has positive HP after capturing all the remaining pieces via counter during combat. Even with the ‘brave’ effect associated with H ’s equipped weapon, it will not capture C_1 and instead become captured. Thus this case yields a contradiction because $M(S, z)$ does not have such a solution.

The one possible exception to this most recent case is where H_x through $H_{|S|}$ instead attempt to capture C_1 and allow H to complete the capture instead. This would imply that C_0 ’s remaining HP is at most 2 so that H can successfully capture it. However, the case where C_0 ’s HP is 1 has already been dismissed, which means

C_0 's remaining HP is 2 and the damage dealt so far is $\sum_{i=1}^{x-1} r_i = z$. Then the damage that can be applied to C_1 is

$$\left(\sum_{i=x}^{|S|} r_i \right) + 2 = \left(\sum_{i=1}^{|S|} r_i \right) - \left(\sum_{i=1}^{x-1} r_i \right) + 2 = \left(\sum_{s \in S} s \right) - z + 2.$$

This is exactly enough damage so that both C_0 and C_1 are captured by this case's sequence of actions. Although this confirms that there is a solution to $M(S, z)$, we point out that the pieces H_1 through H_{x-1} that attempted to capture C_0 inflicted exactly z damage. Due to the human-controlled player's pieces' correspondence with elements in S , this means that there exists elements $s_1, s_2, \dots, s_{x-1} \in S$ whose sum is z . Thus this case also yields a contradiction because (S, z) has a solution.

Therefore, no matter which case holds, we derive a contradiction where either $M(S, z)$ does not have a solution to the FEH problem or (S, z) does have a solution to the Subset Sum problem. Hence it must be the case that if $M(S, z) \in \text{FEH}$ evaluates to \top , then (S, z) evaluates to \top . \square

A.5 Concluding Remarks: Significance of FEH's Complexity

This appendix introduces a new NP-complete problem, the generalized form of Fire Emblem Heroes™ (FEH). In addition to the typical contributions that new NP-complete problems provide, namely providing another option for polynomial-time reductions when evaluating other algorithms' computational complexity, FEH is the first two-player game in this complexity class to our knowledge. In general, the computational complexity of determining whether someone can win a two-player game is PSPACE-hard or greater.

Section A.4.2 only proves that FEH is NP-hard, which is a lower bound on computational complexity. So this alone meant that FEH could still be in the PSPACE complexity class if verification of solutions required contemplating every possible move

that the opponent could make. However, Section A.4.1 takes advantage of one unique property of FEH that is not present in many two-player games: one player uses a fixed policy so that the player's *behavior is replicable and consequently predictable*. These features enable the player without a fixed policy to perfectly simulate their opponent and realize the outcomes of their own moves more quickly than considering every possible way that someone could respond. That is, the *uncertainty is eliminated for simpler verification*. Hence this predictability alone yields the upper bound of NP on the computational complexity, which is the key to proving FEH's simpler complexity.

In this dissertation, we propose the use of recognition algorithms in order to make other agents as predictable during interaction. Although such algorithms cannot guarantee finding their interactive partners' exact policies, they do provide insights into motivations and upcoming actions that allow some degree of prediction during the decision making process for interaction. As the extremes of such insights reduced the complexity of FEH compared to other two-player games, we believe that frameworks such as PRETCIL can similarly reduce the complexity of deciding how to interact around others.

BIBLIOGRAPHY

- [1] Adamson, David, Dyke, Gregory, Jang, Hyeju, and Rosé, Carolyn Penstein. Towards an agile approach to adapting dynamic collaboration support to student needs. *International Journal of Artificial Intelligence in Education* 24, 1 (2014), 92–124.
- [2] Aggarwal, J.K., and Ryoo, M.S. Human activity analysis: A review. *ACM Computing Surveys* 43, 3 (Apr. 2011), 16:1–16:43.
- [3] Aha, David W., Darrell, Trevor, Pazzani, Michael, Reid, Darryn, Sammut, Claude, and Stone, Peter, Eds. *IJCAI-17 Workshop on Explainable AI (XAI) Proceedings*. Melbourne, Australia, 2017.
- [4] Akariss. KNOWLEDGE! AI Lessons - #1 Rally Traps | Fire Emblem Heroes. <https://youtu.be/NGLoh8vuorI>, Aug. 2019. [YouTube] Uploaded by Akariss.
- [5] Akgun, Baris, Cakmak, Maya, Yoo, Jae Wook, and Thomaz, Andrea Lockerd. Trajectories and keyframes for kinesthetic teaching: A human-robot interaction perspective. In *Proceedings of the Seventh Annual ACM/IEEE International Conference on Human-Robot Interaction* (Boston, Massachusetts, USA, 2012), pp. 391–398.
- [6] Aljazzar, Husain, and Leue, Stefan. K*: A heuristic search algorithm for finding the k shortest paths. *Artificial Intelligence* 175, 18 (2011), 2129–2154.
- [7] Amir, Ofra, and Gal, Ya’akov. Plan recognition and visualization in exploratory learning environments. *ACM Transactions on Interactive Intelligent Systems* 3, 3 (2013), 16:1–16:23.
- [8] Amir, Ofra, Kamar, Ece, Kolobov, Andrey, and Grosz, Barbara J. Interactive teaching strategies for agent training. In *International Joint Conferences on Artificial Intelligence* (2016), pp. 804–811.
- [9] Amodei, Dario, Olah, Chris, Steinhardt, Jacob, Christiano, Paul, Schulman, John, and Mané, Dan. Concrete problems in AI safety. *CoRR abs/1606.06565* (2016).
- [10] Anjum, Alvina, and Ilyas, Muhammad U. Activity recognition using smart-phone sensors. In *Proceedings of the IEEE Consumer Communications and Networking Conference* (Las Vegas, Nevada, USA, 2013), pp. 914–919.

- [11] Asai, Masataro, and Fukunaga, Alex. Fully automated cyclic planning for large-scale manufacturing domains. In *Proceedings of the Twenty-Fourth International Conference on Planning and Scheduling* (Portsmouth, New Hampshire, USA, 2014), pp. 20–28.
- [12] Asai, Masataro, and Fukunaga, Alex. Tie-breaking strategies for cost-optimal best first search. *Journal of Artificial Intelligence Research* 58 (2017), 67–121.
- [13] Asoh, Hideki, Vlassis, Nikos A., Motomura, Yoichi, Asano, Futoshi, Hara, Isao, Hayamizu, Satoru, Itou, Katsunobu, Kurita, Takio, Matsui, Toshihiro, Bunschoten, Roland, and Kröse, Ben J. A. Jijo-2: An office robot that communicates and learns. *IEEE Intelligent Systems* 16, 5 (2001), 46–55.
- [14] Aström, Karl J. Optimal control of Markov decision processes with incomplete state estimation. *Journal of Mathematical Analysis and Applications* 10 (1965), 174–205.
- [15] at Cornell University, Robot Learning Lab. Cornell activity datasets: CAD-60 & CAD-120. <http://pr.cs.cornell.edu/humanactivities/data.php>, 2009–2013. [Online].
- [16] Bäckström, Christer. Equivalence and tractability results for SAS+ planning. In *Proceedings of the Third International Conference on Principles of Knowledge Representation and Reasoning* (Cambridge, Massachusetts, USA, Oct. 1992), pp. 126–137.
- [17] Bai, Lu, Yeung, Chris, Efstratiou, Christos, and Chikomo, Moyra. Motion2Vector: Unsupervised learning in human activity recognition using wrist-sensing data. In *Adjunct Proceedings of the 2019 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2019 ACM International Symposium on Wearable Computers* (London, United Kingdom, 2019), UbiComp/ISWC ’19 Adjunct, Association for Computing Machinery, pp. 537–542.
- [18] Baker, Chris L., Saxe, Rebecca, and Tenenbaum, Joshua B. Action understanding as inverse planning. *Cognition* 113, 3 (2009), 329–349. Reinforcement learning and higher cognition.
- [19] Baker, Chris L., Tenenbaum, Joshua B., and Saxe, Rebecca. Goal inference as inverse planning. *Proceedings of the Annual Meeting of the Cognitive Science Society* 29, 29 (Oct. 2007), 779–784.
- [20] Belle, Vaishak, and Levesque, Hector. Foundations for generalized planning in unbounded stochastic domains. In *Proceedings of the Fifteenth International Conference on Principles of Knowledge Representation and Reasoning* (2016), pp. 380–389.

- [21] Bellman, Richard E. *Dynamic Programming*. Dover Books on Computer Science Series. Dover Publications, Mineola, New York, USA, 2003.
- [22] Benton, J., Coles, Amanda, and Coles, Andrew. Temporal planning with preferences and time-dependent continuous costs. In *Proceedings of the Twenty-Second International Conference on Automated Planning and Scheduling* (São Paulo, Brazil, 2013), ICAPS'12, AAAI Press, pp. 2–10.
- [23] Bernstein, Daniel S., Givan, Robert, Immerman, Neil, and Zilberstein, Shlomo. The complexity of decentralized control of Markov decision processes. *Mathematics of Operations Research* 27 (2002), 819–840.
- [24] Bertoli, Piergiorgio, Cimatti, Alessandro, Roveri, Marco, and Traverso, Paolo. Strong planning under partial observability. *Artificial Intelligence* 170, 4 (2006), 337–384.
- [25] Blaylock, Nate, and Allen, James. Fast hierarchical goal schema recognition. In *Proceedings of the Twenty-First National Conference on Artificial Intelligence* (Boston, Massachusetts, 2006), AAAI'06, AAAI Press, pp. 796–801.
- [26] Blei, David M., Ng, Andrew Y., and Jordan, Michael I. Latent Dirichlet allocation. *Journal of Machine Learning Research* 3 (2003), 993–1022.
- [27] Blum, Avrim L., and Furst, Merrick L. Fast planning through planning graph analysis. *Artificial Intelligence* 90, 1-2 (Feb. 1997), 281–300.
- [28] Bock, David L., Kettles, Doug, and Harrison, John. Automated, autonomous and connected vehicle technology assessment. Tech. rep., Electric Vehicle Transportation Center, Cocoa, Florida, 2016.
- [29] Böhmer, Matthias, Hecht, Brent, Schöning, Johannes, Krüger, Antonio, and Bauer, Gernot. Falling asleep with Angry Birds, Facebook and Kindle – A large scale study on mobile application usage. In *Proceedings of the Thirteenth International Conference on Human Computer Interaction with Mobile Devices and Services* (Stockholm, Sweden, 2011), MobileHCI '11, Association for Computing Machinery, pp. 47—56.
- [30] Bonet, Blai, and Geffner, Hector. Planning as heuristic search. *Artificial Intelligence* 129, 1-2 (2001), 5–33.
- [31] Borgo, Rita, Cashmore, Michael, and Magazzeni, Daniele. Towards providing justifications for planner decisions. In *Proceedings of the Second Workshop on Explainable Artificial Intelligence* (Stockholm, Sweden, July 2018), pp. 11–17.
- [32] Botea, Adi, Müller, Martin, and Schaeffer, Jonathan. Near optimal hierarchical path-finding. *Journal of Game Development* 1, 1 (2004), 7–28.

- [33] Bridle, James. The Nightmare Videos of Children’s YouTube—and What’s Wrong with the Internet Today — TED Talk. https://www.ted.com/talks/james_bridle_the_nightmare_videos_of_children_s_youtube_and_what_s_wrong_with_the_internet_today?language=en, Apr. 2018. [Online] Uploaded by TED.
- [34] Broekens, Joost, Heerink, Marcel, and Rosendal, Henk. Assistive social robots in elderly care: A review. *Gerontechnology* 8, 2 (2009), 94–103.
- [35] Bui, Hung H., Phung, Dinh Q., and Venkatesh, Svetha. Hierarchical hidden Markov models with general state hierarchy. In *Proceedings of the Nineteenth National Conference on Artificial Intelligence* (San Jose, California, USA, 2004), pp. 324–329.
- [36] Bui, Hung H., Venkatesh, Svetha, and West, Geoff. Policy recognition in the abstract hidden Markov model. *Journal of Artificial Intelligence Research* 17, 1 (2002), 451–499.
- [37] Bylander, Tom. The computational complexity of propositional STRIPS planning. *Artificial Intelligence* 69, 1-2 (1994), 165–204.
- [38] Bylander, Tom. The computational complexity of propositional STRIPS planning. *Artificial Intelligence* 69, 1 (1994), 165–204.
- [39] Cakmak, Maya, and Thomaz, Andrea L. Designing robot learners that ask good questions. In *Proceedings of the Seventh Annual ACM/IEEE International Conference on Human-Robot Interaction* (Boston, MA, USA, 2012), HRI ’12, ACM, pp. 17–24.
- [40] Callaway, Frederick, Lieder, Falk, Krueger, Paul M., and Griffiths, Thomas L. Mouselab-MDP: A new paradigm for tracing how people plan. In *Proceedings of the Third Multidisciplinary Conference on Reinforcement Learning and Decision Making* (Ann Arbor, Michigan, USA, June 2017), pp. 335–339.
- [41] Cambon, Stéphane, Alami, Rachid, and Gravot, Fabien. A hybrid approach to intricate motion, manipulation and task planning. *The International Journal of Robotics Research* 28, 1 (2009), 104–126.
- [42] Castro, Brenda, Roberts, Montana, Mena, Karla, and Boerkoel, Jim. Who takes the lead? automated scheduling for human-robot teams. In *Proceedings of AAAI 2017 Fall Symposium on AI for Human-Robot Interaction* (Arlington, Virginia, USA, 2017), pp. 85–89.
- [43] Castro, Brenda, Roberts, Montana, Mena, Karla, and Boerkoel, Jim. Who takes the lead? Automated scheduling for human-robot teams. In *Proceedings of the Third Symposium on Artificial Intelligence for Human-Robot Interaction at the AAAI Fall Symposium Series* (Arlington, Virginia, USA, Nov. 2017), pp. 85–89.

- [44] Chakraborti, Tathagata, Sreedharan, Sarath, and Kambhampati, Subbarao. Explicability versus explanations in human-aware planning. In *Proceedings of the Seventeenth International Conference on Autonomous Agents and Multiagent Systems* (Stockholm, Sweden, July 2018), pp. 2180–2182.
- [45] Chang, Chin-Liang, and Lee, Richard Char-Tung. *Symbolic Logic and Theorem Proving*. Academic Press, Inc., New York, New York, USA, 1973.
- [46] Changuel, Sahar, and Labroche, Nicolas. Content independent metadata production as a machine learning problem. In *Machine Learning and Data Mining in Pattern Recognition*, Petra Perner, Ed., vol. 7376 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2012, pp. 306–320.
- [47] Charniak, Eugene, and Goldman, Robert. Probabilistic abduction for plan recognition. Tech. Rep. CS-91-12, Brown University, Providence, RI, USA, 1991.
- [48] Charniak, Eugene, and Shimony, Solomon E. Probabilistic semantics for cost based abduction. In *Proceedings of the Eighth AAAI Conference on Artificial Intelligence* (Boston, Massachusetts, USA, 1990), pp. 106–111.
- [49] Chen, Li, Chen, Guanliang, and Wang, Feng. Recommender systems based on user reviews: The state of the art. *User Modeling and User-Adapted Interaction* 25, 2 (2015), 99–154.
- [50] Chen, Yu, Diethe, Tom, and Flach, Peter A. AdlTM: A topic model for discovery of activities of daily living in a smart home. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence* (New York, New York, USA, July 2016), pp. 1404–1410.
- [51] Chikhaoui, Belkacem, Wang, Shengrui, and Pigot, Hélène. ADR-SPLDA: Activity discovery and recognition by combining sequential patterns and latent dirichlet allocation. *Pervasive and Mobile Computing Special Issue on Pervasive Healthcare* 8, 6 (2012), 845–862.
- [52] Choi, Myung Geol, Yang, Kyung-Young, Igarashi, Takeo, Mitani, Jun, and Lee, Jehee. Retrieval and visualization of human motion data via stick figures. *Computer Graphics Forum* 31, 7 (2012), 2057–2065.
- [53] Chu, Yi, Song, Young Chol, Levinson, Richard, and Kautz, Henry A. Interactive activity recognition and prompting to assist people with cognitive disabilities. *Journal of Ambient Intelligence and Smart Environments* 4, 5 (2012), 443–459.
- [54] Clendaniel, Morgan. At volkswagen, robots are coming out of their cages. Fast Company, <http://www.fastcoexist.com/3016848/at-volkswagen-robots-are-coming-out-of-their-cages>, 2013. [Online; posted 9-September-2013].

- [55] Cohn, Robert, Durfee, Edmund, and Singh, Satinder. Comparing action-query strategies in semi-autonomous agents. In *Proceedings of the Tenth International Conference on Autonomous Agents and Multiagent Systems* (2011), pp. 1287–1288.
- [56] Coltin, Brian, Biswas, Joydeep, Pomerleau, Dean, and Veloso, Manuela. Effective semi-autonomous telepresence. In *RoboCup 2011: Robot Soccer World Cup XV*, Thomas Röfer, N. Michael Mayer, Jesus Savage, and Uluç Saranlı, Eds., vol. 7416 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2012, pp. 365–376.
- [57] Culberson, Joseph C., and Schaeffer, Jonathan. Searching with pattern databases. In *Proceedings of the Eleventh Biennial Conference of the Canadian Society for Computational Studies of Intelligence on Advances in Artificial Intelligence* (Berlin, Heidelberg, 1996), AI ’96, Springer-Verlag, pp. 402–416.
- [58] Daskalakis, Constantinos, Goldberg, Paul W., and Papadimitriou, Christos H. The complexity of computing a nash equilibrium. In *Proceedings of the Thirty-Eighth Annual ACM Symposium on Theory of Computing* (Seattle, WA, USA, 2006), STOC ’06, ACM, pp. 71–78.
- [59] Davidov, Dmitry, and Markovitch, Shaul. Multiple-goal heuristic search. *Journal of Artificial Intelligence Research* 26 (2006), 417–451.
- [60] De la Torre, Fernando, Hodgins, Jessica, Bargtail, Adam, Martin, Xavier, Macey, Justin, Collado, Alex, and Beltran, Pep. Guide to the Carnegie Mellon University multimodal activity (CMU-MMAC) database. Tech. rep., Robotics Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA, April 2008.
- [61] Dean, Thomas, Kaelbling, Leslie P., Kirman, Jak, and Nicholson, Ann. Planning under time constraints in stochastic domains. *Artificial Intelligence* 76 (1995), 35–74.
- [62] Di, Pei, Huang, Jian, Nakagawa, Shotaro, Sekiyama, Kosuke, and Fukuda, Toshio. Fall detection and prevention in the elderly based on the ZMP stability control. In *IEEE Workshop on Advanced Robotics and its Social Impacts (ARSO)* (Tokyo, Japan, 2013), pp. 82–87.
- [63] Dixit, Avinash K., Skeath, Susan, and Reiley Jr., David H. *Games of Strategy*, 4 ed. W. W. Norton & Company, 2014.
- [64] Domshlak, Carmel, Hoffmann, Jörg, and Katz, Michael. Red-black planning. *Artificial Intelligence* 221, C (Apr. 2015), 73–114.
- [65] Dragan, Anca, and Srinivasa, Siddhartha. Generating legible motion. In *Proceedings of Robotics: Science and Systems* (Berlin, Germany, June 2013).

- [66] Dragan, Anca, and Srinivasa, Siddhartha. Integrating human observer inferences into robot motion planning. *Autonomous Robots* 37, 4 (Dec. 2014), 351–368.
- [67] Droeschen, David, Stückler, Jörg, and Behnke, Sven. Learning to interpret pointing gestures with a time-of-flight camera. In *Proceedings of the Sixth ACM/IEEE International Conference on Human-Robot Interaction* (Lausanne, Switzerland, 2011), pp. 481–488.
- [68] E-Martín, Yolanda, R-Moreno, María D., and Smith, David E. A fast goal recognition technique based on interaction estimates. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence* (Buenos Aires, Argentina, July 2015), pp. 761–768.
- [69] E-Martín, Yolanda, and Smith, David E. Goal recognition with noisy observations. In *Proceedings of the AAAI Workshop on Plan, Activity, and Intent Recognition* (2017), pp. 804–811.
- [70] Erol, Kutluhan, Hendler, James, and Nau, Dana S. HTN planning: Complexity and expressivity. In *Proceedings of the Twelfth National Conference on Artificial Intelligence* (Seattle, Washington, 1994), pp. 1123–1128.
- [71] Faria, Diego R., Vieira, Mario, Premebida, Cristiano, and Nunes, Urbano. Probabilistic human daily activity recognition towards robot-assisted living. In *Proceedings of the Twenty-Fourth IEEE International Symposium on Robot and Human Interactive Communication* (Kobe, Japan, 2015), pp. 582–587.
- [72] Fasola, Juan, and Matarić, Maja J. Socially assistive robot exercise coach: Motivating older adults to engage in physical exercise. In *Experimental Robotics*, Jaydev P. Desai, Gregory Dudek, Oussama Khatib, and Vijay Kumar, Eds., vol. 88 of *Springer Tracts in Advanced Robotics*. Springer International Publishing, 2013, pp. 463–479.
- [73] Fikes, Richard E., and Nilsson, Nils J. STRIPS: A new approach to the application of theorem proving to problem solving. In *Proceedings of the Second International Joint Conference on Artificial Intelligence* (London, England, 1971), pp. 608–620.
- [74] Fine, Shai, Singer, Yoram, and Tishby, Naftali. The hierarchical hidden Markov model: Analysis and applications. *Machine Learning* 32, 1 (1998), 41–62.
- [75] Fox, Maria, and Long, Derek. PDDL2.1: An extension to PDDL for expressing temporal planning domains. *Journal of Artificial Intelligence Research* 20, 1 (Dec. 2003), 61–124.
- [76] Fox, Maria, Long, Derek, and Magazzeni, Daniele. Explainable planning. In *IJCAI-17 Workshop on Explainable AI Proceedings* (Melbourne, Australia, 2017), pp. 24–30.

- [77] Freedman, Richard G., Chakraborti, Tathagata, Talamadupula, Kartik, Magazzeni, Daniele, and Frank, Jeremy D. User interfaces and scheduling and planning: Workshop summary and proposed challenges. In *The 2018 AAAI Spring Symposium Series: The Design of the User Experience for Artificial Intelligence* (Stanford, California, USA, 2018), p. In Press.
- [78] Freedman, Richard G., Guo, Jingyi, Turkett, William H., and Pauca, V. Paúl. Hierarchical modeling to facilitate personalized word prediction for dialogue. In *Proceedings of the AAAI Workshops: Plan, Activity, Intent Recognition (PAIR)* (Bellevue, Washington, USA, 2013), pp. 2–9.
- [79] Freedman, Richard G., Jung, Hee-Tae, and Zilberstein, Shlomo. Plan and activity recognition from a topic modeling perspective. In *Proceedings of the Twenty-Fourth International Conference on Automated Planning and Scheduling* (Portsmouth, New Hampshire, USA, 2014), pp. 360–364.
- [80] Freedman, Richard G., Jung, Hee-Tae, and Zilberstein, Shlomo. Temporal and object relations in unsupervised plan and activity recognition. In *Proceedings of AAAI 2015 Fall Symposium on AI for Human-Robot Interaction* (Arlington, Virginia, USA, 2015), pp. 51–59.
- [81] Freedman, Richard G., and Zilberstein, Shlomo. Using metadata to automate interpretations of unsupervised learning-derived clusters. In *First IJCAI Workshop on Human is More Than a Labeler (BeyondLabeler)* (New York, New York, USA, 2016), pp. 1–7.
- [82] Freedman, Richard G., and Zilberstein, Shlomo. Integration of planning with recognition for responsive interaction using classical planners. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence* (San Francisco, CA, USA, 2017), pp. 4581–4588.
- [83] Freedman, Richard G., and Zilberstein, Shlomo. A unifying perspective of plan, activity, and intent recognition. In *Proceedings of the AAAI Workshops: Plan, Activity, Intent Recognition* (Honolulu, Hawaii, USA, 2019), pp. 1–8.
- [84] Frigyik, Bela A., Kapila, Amol, and Gupta, Maya R. Introduction to the dirichlet distribution and related processes. Tech. Rep. UWEETR-2010-0006, University of Washington, Seattle, WA, USA, 2010.
- [85] Fu, Jicheng, Calderon Jaramillo, Andres, Ng, Vincent, Bastani, Farokh B., and Yen, I-Ling. Fast strong planning for fully observable nondeterministic planning problems. *Annals of Mathematics and Artificial Intelligence* 78, 2 (Oct. 2016), 131–155.
- [86] Gabrilovich, Evgeniy, and Markovitch, Shaul. Wikipedia-based semantic interpretation for natural language processing. *Journal of Artificial Intelligence Research* 34, 1 (Mar. 2009), 443–498.

- [87] Gales, Mark, and Young, Steve. The application of hidden Markov models in speech recognition. *Foundations and Trends in Signal Processing* 1, 3 (Jan. 2007), 195–304.
- [88] Garey, Michael R., and Johnson, David S. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, New York, USA, 1990.
- [89] Geib, Christopher, and Goldman, Robert. Recognizing plans with loops represented in a lexicalized grammar. In *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence* (San Francisco, California, USA, 2011), pp. 958–963.
- [90] Geib, Christopher, Weerasinghe, Janith, Matskevich, Sergey, Kantharaju, Pavani, Craensen, Bart, and Petrick, Ronald P. A. Building helpful virtual agents using plan recognition and planning. In *Twelfth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment* (San Francisco, California, USA, 2016), pp. 162–168.
- [91] Geib, Christopher W. Delaying commitment in plan recognition using combinatorial categorial grammars. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence* (Pasadena, California, USA, 2009), IJCAI’09, Morgan Kaufmann Publishers Inc., pp. 1702–1707.
- [92] Geib, Christopher W., and Steedman, Mark. On natural language processing and plan recognition. In *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence* (Hyderabad, India, 2007), pp. 1612–1617.
- [93] Gerevini, Alfonso E., Haslum, Patrik, Long, Derek, Saetti, Alessandro, and Dimopoulos, Yannis. Deterministic planning in the fifth international planning competition: {PDDL3} and experimental evaluation of the planners. *Artificial Intelligence* 173, 5–6 (2009), 619–668. Advances in Automated Plan Generation.
- [94] Giampapa, Joseph A., Sycara, Katia P., and Sukthankar, Gita. Toward identifying process models in ad hoc and distributed teams. In *Proceedings of the First International Working Conference on Human Factors and Computational Models in Negotiation* (Delft, The Netherlands, 2008), HuCom ’08, Association for Computing Machinery, pp. 55–62.
- [95] Gibson, Eleanor J. *Perceiving the Affordances: A Portrait of Two Psychologists*. Taylor & Francis, Milton Park Abingdon, Oxfordshire, UK, 2001.
- [96] Goldman, Robert P., Geib, Christopher W., Kautz, Henry, and Asfour, Tamim. Plan Recognition – Dagstuhl Seminar 11141. *Dagstuhl Reports* 1, 4 (2011), 1–22.

- [97] Goldman, Robert P., and Kuter, Ugur. Hierarchical task network planning in common lisp: The case of SHOP3. In *Proceedings of the Twelfth European LISP Symposium* (Genova, Italy, Apr. 2019), Zenodo, pp. 73–80. This material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA) and the Air Force Research Laboratory under Contract Number FA8750-17-C-0184. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of DARPA, the Department of Defense, or the United States Government.
- [98] Goldsmith, Judy, and Mundhenk, Martin. Competition adds complexity. In *Proceedings of the Twenty-First Annual Conference on Neural Information Processing Systems* (Vancouver, British Columbia, Canada, Dec. 2007), pp. 561–568.
- [99] Gombolay, Matthew, Gutierrez, Reymundo, Sturla, Giancarlo, and Shah, Julie. Decision-making authority, team efficiency and human worker satisfaction in mixed human-robot teams. In *Proceedings of Robotics: Science and Systems* (Berkeley, California, USA, 2014).
- [100] Gombolay, Matthew, Jensen, Reed, Stigile, Jessica, Son, Sung-Hyun, and Shah, Julie. Learning to tutor from expert demonstration via apprenticeship scheduling. In *Proceedings of the Association for the Advancement of Artificial Intelligence Workshop on Human-Machine Collaborative Learning* (San Francisco, California, USA, February 4 2017).
- [101] Gombolay, Matthew, Yang, Xi Jessie, Hayes, Bradley, Seo, Nicole, Liu, Zixi, Wadhwania, Samir, Yu, Tania, Shah, Neel, Golen, Toni, and Shah, Julie. Robotic assistance in the coordination of patient care. *The International Journal of Robotics Research* 37, 10 (2018), 1300–1316.
- [102] Gordon, Goren, Spaulding, Samuel, Westlund, Jacqueline Kory, Lee, Jin Joo, Plummer, Luke, Martinez, Marayna, Das, Madhurima, and Breazeal, Cynthia. Affective personalization of a social robot tutor for children’s second language skills. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence* (Phoenix, Arizona, USA, Feb. 2016), Dale Schuurmans and Michael P. Wellman, Eds., AAAI Press, pp. 3951–3957.
- [103] Gori, Ilaria, Sinapov, Jivko, Khante, Priyanka, Stone, Peter, and Aggarwal, J.K. Robot-centric activity recognition ‘in the wild’. In *Proceedings of the International Conference on Social Robotics (ICSR)* (Paris, France, October 2015), pp. 224–234.
- [104] Graepel, Thore, Herbrich, Ralf, and Gold, Julian. Learning to fight. In *Proceedings of the International Conference on Computer Games: Artificial Intelligence, Design and Education* (2004), pp. 193–200.

- [105] Greenwald, Amy, Campbell, Murray, Bowling, Michael, Kitano, Hiroaki, Kasparov, Garry, and Silver, David. AI History Panel: Advancing AI by Playing Games. <https://vimeo.com/389556398>, Feb. 2020. [Online] Uploaded by AAAI Livestreaming.
- [106] Griffiths, Thomas L. Gibbs sampling in the generative model of latent Dirichlet allocation. Tech. rep., Stanford University, 2002.
- [107] Griffiths, Thomas L., Steyvers, Mark, Blei, David M., and Tenenbaum, Joshua B. Integrating topics and syntax. In *Proceedings of the Eighteenth Annual Conference on Neural Information Processing Systems* (Vancouver, BC, Canada, 2004), pp. 537–544.
- [108] Grosz, Barbara J. Beyond mice and menus. *Proceedings of the American Philosophical Society* 149, 4 (2005), 529–543.
- [109] Guizzo, Eric. Cynthia Breazeal Unveils Jibo, a Social Robot for the Home. IEEE Spectrum, <http://spectrum.ieee.org/automaton/robotics/home-robots/cynthia-breazeal-unveils-jibo-a-social-robot-for-the-home>, July 2014. [Online; posted 16-July-2014].
- [110] Hajibagheri, Alireza, Lakkaraju, Kiran, Sukthankar, Gita, Wigand, Rolf T., and Agarwal, Nitin. Conflict and communication in massively-multiplayer online games. In *Proceedings of the Eighth International Conference on Social Computing, Behavioral-Cultural Modeling, and Prediction* (Washington DC, USA, Mar. 2015), pp. 65–74.
- [111] Hansen, Eric A., and Zhou, Rong. Anytime heuristic search. *Journal of Artificial Intelligence Research* 28, 1 (2007), 267–297.
- [112] Harmon, Leon D. Automated tactile sensing. *The International Journal of Robotics Research* 1, 2 (1982), 3–32.
- [113] Hart, Peter E., Nilsson, Nils J., and Raphael, Bertram. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics* 4, 2 (July 1968), 100–107.
- [114] Hayes, Bradley, Cakmak, Maya, and Rosenthal, Stephanie. Introduction to the special issue on artificial intelligence and human-robot interaction. *Journal of Human-Robot Interaction* 7, 2 (Oct. 2018).
- [115] Hayes, Bradley, and Shah, Julie A. Interpretable models for fast activity recognition and anomaly explanation during collaborative robotics tasks. In *IEEE International Conference on Robotics and Automation* (2017), pp. 6586–6593.
- [116] Helmert, Malte. The fast downward planning system. *Journal of Artificial Intelligence Research* 26, 1 (2006), 191–246.

- [117] Herbst, Evan, Ren, Xiaofeng, and Fox, Dieter. RGB-D flow: Dense 3-D motion estimation using color and depth. In *Proceedings of the IEEE International Conference on Robotics and Automation* (Karlesruhe, Germany, 2013), pp. 2276–2282.
- [118] Higginbotham, D. Jeffery, Lesher, Gregory W., Moulton, Bryan J., and Roark, Brian. The application of natural language processing to augmentative and alternative communication. *Assistive Technology* 24, 1 (2012), 14–24.
- [119] Hobbs, Jerry R., Stickel, Mark E., Appelt, Douglas E., and Martin, Paul. Interpretation as abduction. *Artificial Intelligence* 63, 1 (1993), 69–142.
- [120] Hochreiter, Sepp, and Schmidhuber, Jürgen. Long short-term memory. *Neural Comput.* 9, 8 (Nov. 1997), 1735–1780.
- [121] Hodgins, Jessica K. Carnegie Mellon Graphics Lab Motion Capture Database. <http://mocap.cs.cmu.edu>, 2002. [Online, last viewed August 2017].
- [122] Hoffmann, Jörg. Analyzing search topology without running any search: On the connection between causal graphs and h+. *Journal of Artificial Intelligence Research* 41 (2011), 155–229.
- [123] Hoffmann, Jörg, and Nebel, Bernhard. The FF planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research* 14 (2001), 253–302.
- [124] Houvaras, Anghus. Slaves to an Algorithm: The YouTube Creator Conundrum. <https://medium.com/@theanghus/slaves-to-an-algorithm-the-youtube-creator-conundrum-e1ad75aa8fe6>, Dec. 2018. [Online] Last accessed 2020 March 6.
- [125] Hu, Yue, Harabor, Daniel, Qin, Long, Yin, Quanjun, and Hu, Cong. Improving the combination of JPS and geometric containers. In *Proceedings of the Twenty-Ninth International Conference on Automated Planning and Scheduling* (Berkeley, California, USA, 2019), pp. 209–213.
- [126] Hu, Yuxiao, and De Giacomo, Giuseppe. Generalized planning: Synthesizing plans that work for multiple environments. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence* (Barcelona, Catalonia, Spain, 2011), pp. 918–923.
- [127] Hu, Yuxiao, and Levesque, Hector. A correctness result for reasoning about one-dimensional planning problems. In *Proceedings of the Twelfth International Conference on Principles of Knowledge Representation and Reasoning* (2010), pp. 362–371.
- [128] Huỳnh, Tâm, Fritz, Mario, and Schiele, Bernt. Discovery of activity patterns using topic models. In *Proceedings of the Tenth International Conference on Ubiquitous Computing* (Seoul, South Korea, 2008), pp. 10–19.

- [129] Inoue, Naoya, and Inui, Kentaro. Ilp-based reasoning for weighted abduction. In *Proceedings of the AAAI Workshops: Plan, Activity, Intent Recognition (PAIR)* (San Francisco, California, USA, 2011), pp. 25–32.
- [130] Isbell, Charles, Shelton, Christian R., Kearns, Michael, Singh, Satinder, and Stone, Peter. A social reinforcement learning agent. In *Proceedings of the Fifth International Conference on Autonomous Agents* (Montreal, Quebec, Canada, 2001), AGENTS '01, ACM, pp. 377–384.
- [131] Jackson, Peter. *Introduction to Expert Systems*, 3rd ed. Addison-Wesley Longman Publishing Co., Inc., USA, 1998.
- [132] Jacoff, Adam, Messina, Elena, Weiss, Brian A., Tadokoro, Satoshi, and Nakagawa, Yuki. Test arenas and performance metrics for urban search and rescue robots. In *Proceedings of the 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems* (Las Vega, Nevada, USA, Oct. 2003), vol. 4 of *IROS*, pp. 3396–3403.
- [133] Jain, Raghvendra, and Inamura, Tetsunari. Bayesian learning of tool affordances based on generalization of functional feature to estimate effects of unseen tools. *Artificial Life and Robotics* 18, 1-2 (2013), 95–103.
- [134] James, Mike. Microsoft research shows how to turn any camera into a depth camera. I Programmer, <http://www.i-programmer.info/news/194-kinect/7641-microsoft-research-shows-how-to-turn-any-camera-into-a-depth-camera.html>, Aug. 2014. [Online; posted 13-August-2014].
- [135] Jiang, Yun, Koppula, Hema S., and Saxena, Ashutosh. Hallucinated humans as the hidden context for labeling 3D scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (Portland, Oregon, USA, 2013), pp. 2993–3000.
- [136] Johns, Mishel, Sibi, Srinath, and Ju, Wendy. Effect of cognitive load in autonomous vehicles on driver performance during transfer of control. In *Adjunct Proceedings of the Sixth International Conference on Automotive User Interfaces and Interactive Vehicular Applications* (Seattle, WA, USA, 2014), AutomotiveUI '14, ACM, pp. 1–4.
- [137] Joo, Hanbyul, Liu, Hao, Tan, Lei, Gui, Lin, Nabbe, Bart, Matthews, Iain, Kanade, Takeo, Nobuhara, Shohei, and Sheikh, Yaser. Panoptic studio: A massively multiview system for social motion capture. In *Proceedings of the 2015 IEEE International Conference on Computer Vision* (Santiago, Chile, 2015), ICCV '15, IEEE Computer Society, pp. 3334–3342.
- [138] Jugovac, Michael, and Jannach, Dietmar. Interacting with recommenders: Overview and research directions. *ACM Transactions on Interactive Intelligent Systems* 7, 3 (2017), 10.

- [139] Jung, Hee-Tae, Freedman, Richard G., Foster, Tammie, Choe, Yu-Kyong, Zilbermanstein, Shlomo, and Grupen, Roderic A. Learning therapy strategies from demonstration using latent Dirichlet allocation. In *Proceedings of the Twentieth International Conference on Intelligent User Interfaces* (Atlanta, Georgia, USA, 2015), pp. 432–436.
- [140] Jung, Hee-Tae, Freedman, Richard G., Takahashi, Takeshi, Wong, Jay M., Zilbermanstein, Shlomo, Grupen, Roderic A., and Choe, Yu-Kyong. Adaptive therapy strategies: Efficacy and learning framework. In *Proceedings of the IEEE/RAS-EMBS International Conference on Rehabilitation Robotics* (Singapore, 2015), pp. 950–955.
- [141] Kaelbling, Leslie P., Littman, Michael L., and Cassandra, Anthony R. Planning and acting in partially observable stochastic domains. *Artificial Intelligence* 101 (1998), 99–134.
- [142] Kamar, Ece, Hacker, Severin, and Horvitz, Eric. Combining human and machine intelligence in large-scale crowdsourcing. In *Proceedings of the Eleventh International Conference on Autonomous Agents and Multiagent Systems* (2012), pp. 467–474.
- [143] Kambhampati, Subbarao. Challenges of Human-Aware AI Systems. <http://rakaposhi.eas.asu.edu/haai-aaai/>, Feb. 2018. [Online] Last accessed 2020 March 6.
- [144] Kaminka, Gal A., Vered, Mor, and Agmon, Noa. Plan recognition in continuous domains. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence* (New Orleans, Louisiana, USA, 2018), pp. 6202–6210.
- [145] Kang, Cecilia. New robots in the workplace: Job creators or job terminators? Washington Post, March 6, 2013.
- [146] Kantharaju, Pavan, Ontañón, Santiago, and Geib, Christopher W. Scaling up CCG-based plan recognition via Monte-Carlo tree search. In *2019 IEEE Conference on Games* (London, UK, Aug 2019), pp. 1–8.
- [147] Karnaugh, Maurice. A new toolkit for non-admissible heuristic search. In *Proceedings of the Sixth Conference on Artificial Intelligence Applications* (Santa Barbara, California, USA, 1990), IEEE Press, pp. 162–168.
- [148] Katz, Michael, Sohrabi, Shirin, and Udrea, Octavian. Top-Quality planning: Finding practically useful sets of best plans. In *Proceedings of the Thirty-fourth Conference on Artificial Intelligence* (2020).
- [149] Kautz, H. A. *A formal theory of plan recognition*. PhD thesis, University of Rochester, 1987.

- [150] Kautz, Henry A. A formal theory of plan recognition and its implementation. In *Reasoning About Plans*, James F. Allen, Henry A. Kautz, Richard N. Pelavin, and Josh D. Tenenberg, Eds. Morgan Kaufmann, San Francisco, CA, USA, 1991, pp. 69–125.
- [151] Kelley, Richard, Nicolescu, Monica, Tavakkoli, Alireza, King, Christopher, and Bebis, George. Understanding human intentions via hidden Markov models in autonomous mobile robots. In *Proceedings of the Third ACM/IEEE International Conference on Human-Robot Interaction* (Amsterdam, Netherlands, 2008), pp. 367–374.
- [152] Kelley, Richard, Tavakkoli, Alireza, King, Christopher, Ambardekar, Amol, Nicolescu, Monica, and Nicolescu, Mircea. Context-based bayesian intent recognition. *IEEE Transactions on Autonomous Mental Development* 4, 3 (2012), 215–225.
- [153] Kelly, John Paul, Botea, Adi, and Koenig, Sven. Offline planning with hierarchical task networks in video games. In *Proceedings of the Fourth Artificial Intelligence and Interactive Digital Entertainment Conference* (2008), pp. 60–65.
- [154] Kelly, Norene. All the world’s a stage: What makes a wearable socially acceptable. *Interactions* 24, 6 (Oct. 2017), 56–60.
- [155] Kennedy, James, Baxter, Paul, and Belpaeme, Tony. The robot who tried too hard: Social behaviour of a robot tutor can negatively affect child learning. In *Proceedings of the Tenth Annual ACM/IEEE International Conference on Human-Robot Interaction* (Portland, Oregon, USA, 2015), HRI ’15, ACM, pp. 67–74.
- [156] Keren, Sarah, Gal, Avigdor, and Karpas, Erez. Goal recognition design. In *Proceedings of the Twenty-Fourth International Conference on Automated Planning and Scheduling* (Portsmouth, New Hampshire, USA, 2014), pp. 154–162.
- [157] Keren, Sarah, Gal, Avigdor, and Karpas, Erez. Goal recognition design for non-optimal agents. In *Proceedings of the Twenty-Ninth International Conference on Artificial Intelligence* (Austin, Texas, USA, 2015), pp. 3298–3304.
- [158] Kim, Been, Rudin, Cynthia, and Shah, Julie A. The Bayesian case model: A generative approach for case-based reasoning and prototype classification. In *Advances in Neural Information Processing Systems Twenty-Seven*, Zoubin Ghahramani, Max Welling, Corrina Cortes, Neil D. Lawrence, and Kilian Q. Weinberger, Eds. Curran Associates, Inc., 2014, pp. 1952–1960.
- [159] Kim, Phil, Williams, Brian C., and Abramson, Mark. Executing reactive, model-based programs through graph-based temporal planning. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence* (Seattle, Washington, USA, 2001), IJCAI’01, Morgan Kaufmann Publishers Inc., pp. 487–493.

- [160] Kingma, Diederik P., and Welling, Max. Auto-encoding variational bayes. In *Second International Conference on Learning Representations* (Banff, Alberta, Canada, Apr. 2014).
- [161] Kitano, Hiroaki, and Tadokoro, Satoshi. Robocup rescue: A grand challenge for multiagent and intelligent systems. *AI Magazine* 22, 1 (Mar. 2001), 39.
- [162] Kobren, Ari, Tan, Chun How, Ipeirotis, Panagiotis, and Gabrilovich, Evgeniy. Getting more for less: Optimized crowdsourcing with dynamic tasks and goals. In *Proceedings of the Twenty-Fourth International Conference on World Wide Web* (Florence, Italy, 2015), International World Wide Web Conferences Steering Committee, pp. 592–602.
- [163] Koedinger, Kenneth R., and Corbett, Albert T. Cognitive tutors: Technology bringing learning science to the classroom. In *The Cambridge Handbook of the Learning Sciences*, R. Keith Sawyer, Ed. Cambridge University Press, 2006, pp. 61–77.
- [164] Koppula, Hema S., and Saxena, Ashutosh. Learning spatio-temporal structure from RGB-D videos for human activity detection and anticipation. In *Proceedings of the International Conference on Machine Learning* (Atlanta, Georgia, USA, 2013), pp. 792–800.
- [165] Kory, Jacqueline, and Breazeal, Cynthia. Storytelling with robots: Learning companions for preschool children’s language development. In *The 23rd IEEE International Symposium on Robot and Human Interactive Communication* (Aug 2014), pp. 643–648.
- [166] Kring, Alex, Champandard, Alex J., and Samarin, Nick. DHPA* and SHPA*: Efficient hierarchical pathfinding in dynamic and static game worlds. In *Proceedings of the Sixth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment* (Stanford, California, USA, 2010), AIIDE’10, AAAI Press, pp. 39–44.
- [167] Kulkarni, Anagha, Klenk, Matthew, Rane, Shantanu, and Soroush, Hamed. Resource bounded secure goal obfuscation. In *Proceedings of the AAAI Workshops: Plan, Activity, Intent Recognition* (Honolulu, Hawaii, USA, 2019), pp. 1–8.
- [168] Kulkarni, Anagha, Srivastava, Siddharth, and Kambhampati, Subbarao. A unified framework for planning in adversarial and cooperative environments. In *Proceedings of the The Thirty-Third AAAI Conference on Artificial Intelligence* (Honolulu, Hawaii, USA, 2019), pp. 2479–2487.
- [169] Kunapuli, Gautam, Odom, Phillip, Shavlik, Jude W., and Natarajan, Sriraam. Guiding autonomous agents to better behaviors through human advice. In *Proceedings of the IEEE Thirteenth International Conference on Data Mining* (Dallas, Texas, USA, 2013), pp. 409–418.

- [170] Kuter, Ugur, Nau, Dana, Reisner, Elnatan, and Goldman, Robert P. Using classical planners to solve nondeterministic planning problems. In *Proceedings of the International Conference on Automated Planning and Scheduling* (Sydney, Australia, 2008), pp. 190–197.
- [171] LaValle, Steven M. *Planning Algorithms*. Cambridge University Press, New York, NY, USA, 2006.
- [172] Leddy, Michael T., and Dollar, Aaron M. Preliminary design and evaluation of a single-actuator anthropomorphic prosthetic hand with multiple distinct grasp types. In *2018 Seventh IEEE International Conference on Biomedical Robotics and Biomechatronics* (Enschede, The Netherlands, Aug. 2018), Biorob, pp. 1062–1069.
- [173] Lesh, Neal. Adaptive goal recognition. In *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence - Volume 2* (Nagoya, Japan, 1997), Morgan Kaufmann Publishers Inc., pp. 1208–1214.
- [174] Levine, Steven, and Williams, Brian. Concurrent plan recognition and execution for human-robot teams. In *Proceedings of the Twenty-Fourth International Conference on Automated Planning and Scheduling* (Portsmouth, New Hampshire, USA, 2014), pp. 490–498.
- [175] Levine, Steven J. *Risk-bounded coordination of human-robot teams through concurrent intent recognition and adaptation*. PhD thesis, Massachusetts Institute of Technology, 2019.
- [176] Levine, Steven J., and Williams, Brian C. Watching and acting together: Concurrent plan recognition and adaptation for human-robot teams. *Journal of Artificial Intelligence Research* 63 (2018), 281–359.
- [177] Li, Yongmou, Shi, Dianxi, Ding, Bo, and Liu, Dongbo. Unsupervised feature learning for human activity recognition using smartphone sensors. In *Proceedings of the Second International Conference on Mining Intelligence and Knowledge Exploration* (Cork, Ireland, 2014), Rajendra Prasath, Philip O'Reilly, and T. Kathirvalavakumar, Eds., Springer International Publishing, pp. 99–107.
- [178] Lichtenstein, David, and Sipser, Michael. GO is polynomial-space hard. *Journal of the ACM* 27, 2 (Apr. 1980), 393–401.
- [179] Littman, Michael L., Goldsmith, Judy, and Mundhenk, Martin. The computational complexity of probabilistic planning. *Journal of Artificial Intelligence Research* 9 (1998), 1–36.
- [180] Lotinac, Damir, Segovia-Aguas, Javier, Jiménez, Sergio, and Jonsson, Anders. Automatic generation of high-level state features for generalized planning. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence* (New York, New York, USA, 2016), pp. 3199–3205.

- [181] Louie, Wing-Yue Geoffrey, McColl, Derek, and Nejat, Goldie. Acceptance and attitudes toward a human-like socially assistive robot by older adults. *Assistive Technology* 26, 3 (2014), 140–150.
- [182] Lusena, Christopher, Goldsmith, Judy, and Mundhenk, Martin. Nonapproximability results for partially observable markov decision processes. *Journal of Artificial Intelligence Research* 14 (2001), 83–113.
- [183] MacGlashan, James, Ho, Mark K., Loftin, Robert, Peng, Bei, Wang, Guan, Roberts, David L., Taylor, Matthew E., and Littman, Michael L. Interactive learning from policy-dependent human feedback. In *Proceedings of the Thirty-Fourth International Conference on Machine Learning* (International Convention Centre, Sydney, Australia, Aug. 2017), Doina Precup and Yee Whye Teh, Eds., vol. 70 of *Proceedings of Machine Learning Research*, PMLR, pp. 2285–2294.
- [184] Madani, Omid, Bui, Hung Hai, and Yeh, Eric. Efficient online learning and prediction of users' desktop actions. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence* (Pasadena, California, USA, July 2009), pp. 1457–1462.
- [185] Maeda, Guilherme, Ewerton, Marco, Lioutikov, Rudolf, Ben Amor, Heni, Peters, Jan, and Neumann, Gerhard. Learning interaction for collaborative tasks with probabilistic movement primitives. In *Proceedings of the Fourteenth IEEE-RAS International Conference on Humanoid Robots* (Madrid, Spain, Nov. 2014), pp. 527–534.
- [186] Mainprice, Jim, Hayne, Rafi, and Berenson, Dmitry. Goal set inverse optimal control and iterative replanning for predicting human reaching motions in shared workspaces. *IEEE Transactions on Robotics* 32, 4 (Aug 2016), 897–908.
- [187] Mandow, L., and De la Cruz, J. L. Pérez. A new approach to multiobjective a* search. In *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence* (Edinburgh, Scotland, 2005), IJCAI'05, Morgan Kaufmann Publishers Inc., pp. 218–223.
- [188] Marr, David. *Vision: A Computational Investigation into the Human Representation and Processing of Visual Information*. Henry Holt and Co., Inc., USA, 1982.
- [189] Masters, Peta, and Sardina, Sebastian. Cost-based goal recognition for path-planning. In *Proceedings of the Sixteenth Conference on Autonomous Agents and Multiagent Systems* (São Paulo, Brazil, 2017), International Foundation for Autonomous Agents and Multiagent Systems, pp. 750–758.
- [190] McCallum, Andrew Kachites. Mallet: A machine learning for language toolkit. <http://mallet.cs.umass.edu>, 2002.

- [191] McDermott, Drew, Ghallab, Malik, Howe, Adele, Knoblock, Craig, Ram, Ashwin, Veloso, Manuela, Weld, Daniel, and Wilkins, David. PDDL – The planning domain definition language – Version 1.2. Tech. Rep. CVC TR-98-003, Yale Center for Computational Vision and Control, New Haven, CT, USA, 1998.
- [192] Min, Wookhee, Ha, Eun Young, Rowe, Jonathan, Mott, Bradford, and Lester, James. Deep learning-based goal recognition in open-ended digital games. In *Proceedings of the Tenth Conference on Artificial Intelligence and Interactive Digital Entertainment* (2014), pp. 37–43.
- [193] Mirsky, Reuth, Gal, Ya’akov, and Shieber, Stuart M. CRADLE: An online plan recognition algorithm for exploratory domains. *ACM Transactions on Intelligent Systems and Technology* 8, 3 (2017), 45:1–45:22.
- [194] Mirsky, Reuth, and Gal, Ya’akov Kobi. SLIM: Semi-lazy inference mechanism for plan recognition. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence* (New York, NY, USA, 2016), pp. 394–400.
- [195] Mirsky, Reuth, Gal, Ya’akov (Kobi), and Shieber, Stuart M. CRADLE: an online plan recognition algorithm for exploratory domains. *ACM Transactions on Intelligent Systems and Technology* 8, 3 (2017), 45:1–45:22.
- [196] Miura, Shuwa, and Zilberstein, Shlomo. Maximizing plan legibility in stochastic environments. In *Proceedings of the AAAI Workshops: Plan, Activity, Intent Recognition (PAIR)* (New York, New York, USA, 2020), pp. 1–7.
- [197] Mohseni-Kabir, Anahita, Rich, Charles, Chernova, Sonia, Sidner, Candace L., and Miller, Daniel. Interactive hierarchical task learning from a single demonstration. In *Proceedings of the Tenth Annual ACM/IEEEInternational Conference on Human-Robot Interaction* (New York, NY, USA, 2015), HRI ’15, ACM, pp. 205–212.
- [198] Mok, Brian Ka-Jun, Yang, Stephen, Sirkin, David, and Ju, Wendy. A place for every tool and every tool in its place: Performing collaborative tasks with interactive robotic drawers. In *2015 24th IEEE International Symposium on Robot and Human Interactive Communication* (Aug 2015), pp. 700–706.
- [199] Mora-Davison, Benjamin. Fire Emblem - The Role of Random. <https://youtu.be/wyYRF9bXAH?t=70>, Oct. 2017. [YouTube] Uploaded by Pavise.
- [200] Mörtl, Alexander, Lawitzky, Martin, Kucukyilmaz, Ayse, Sezgin, Metin, Basdogan, Cagatay, and Hirche, Sandra. The role of roles: Physical cooperation between humans and robots. *International Journal of Robotics Research* 31, 13 (Nov. 2012), 1656–1674.
- [201] Mutlu, Bilge, Kanda, Takayuki, Forlizzi, Jodi, Hodgins, Jessica, and Ishiguro, Hiroshi. Conversational gaze mechanisms for humanlike robots. *ACM Transactions on Interactive Intelligent Systems* 1, 2 (Jan. 2012), 12:1–12:33.

- [202] Natarajan, Sriraam, Bui, Hung H., Tadepalli, Prasad, Kersting, Kristian, and Wong, Weng-Keen. Logical hierarchical hidden Markov models for modeling user activities. In *Proceedings of the Eighteenth International Conference on Inductive Logic Programming* (Prague, Czech Republic, 2008), pp. 192–209.
- [203] Natarajan, Sriraam, Odom, Phillip, Joshi, Saket, Khot, Tushar, Kersting, Kristian, and Tadepalli, Prasad. Accelerating imitation learning in relational domains via transfer by initialization. In *Inductive Logic Programming*, Gerson Zaverucha, Vítor Santos Costa, and Aline Paes, Eds., Lecture Notes in Computer Science. Springer Berlin Heidelberg, Nancy, France, 2014, pp. 64–75.
- [204] Natarajan, Sriraam, Tadepalli, Prasad, and Fern, Alan. A relational hierarchical model for decision-theoretic assistance. *Knowledge and Information Systems* 32, 2 (2012), 329–349.
- [205] Nau, Dana, Au, Tsz-Chiu, Ilghami, Okhtay, Kuter, Ugur, Murdock, J. William, Wu, Dan, and Yaman, Fusun. SHOP2: An HTN planning system. *Journal of Artificial Intelligence Research* 20 (2003), 379–404.
- [206] Nau, Dana, Cao, Yue, Lotem, Amnon, and Munoz-Avila, Hector. SHOP: Simple hierarchical ordered planner. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence - Volume 2* (Stockholm, Sweden, 1999), IJCAI’99, Morgan Kaufmann Publishers Inc., pp. 968–973.
- [207] Nguyen, Tuan A., Kambhampati, Subbarao, and Do, Minh Binh. Synthesizing robust plans under incomplete domain models. *Proceedings of Neural Information Processing Systems* (2013).
- [208] Nguyen, Tuan Anh, Do, Minh, Gerevini, Alfonso Emilio, Serina, Ivan, Srivastava, Biplav, and Kambhampati, Subbarao. Generating diverse plans to handle unknown and partially known user preferences. *Artificial Intelligence* 190 (2012), 1–31.
- [209] Nikolaidis, Stefanos, Lasota, Przemyslaw A., Rossano, Gregory F., Martinez, Carlos, Fuhlbrigge, Thomas A., and Shah, Julie A. Human-robot collaboration in manufacturing: Quantitative evaluation of predictable, convergent joint action. In *Proceedings of the Forty-Fourth Internationel Symposium on Robotics* (Seoul, South Korea, Oct. 2013), pp. 1–6.
- [210] Nikolaidis, Stefanos, and Shah, Julie A. Human-robot interactive planning using cross-training: A human team training approach. In *Proceedings of the Infotech@Aerospace Conference* (2012), pp. 1–11.
- [211] Nintendo. Fire Emblem Heroes. <https://fire-emblem-heroes.com/en/>, 2017. [Online] Last accessed 2020 February 16.
- [212] Oard, Douglas W., and Kim, Jinmook. Implicit feedback for recommender systems. In *Proceedings of the AAAI Workshop on Recommender Systems* (1998), pp. 81–83.

- [213] Ordóñez, Francisco, and Roggen, Daniel. Deep convolutional and LSTM recurrent neural networks for multimodal wearable activity recognition. *Sensors* 16, 1 (Jan 2016), 115.
- [214] Orkin, Jeff. Three states and a plan: The A.I. of F.E.A.R. In *Proceedings of the Game Developers Conference* (San Jose, CA, USA, 2006), pp. 1–18.
- [215] Pednault, Edwin P. D. ADL: Exploring the middle ground between STRIPS and the situation calculus. In *Proceedings of the First International Conference on Principles of Knowledge Representation and Reasoning* (Toronto, Canada, 1989), Morgan Kaufmann Publishers Inc., pp. 324–332.
- [216] Penberthy, J. Scott, and Weld, Daniel S. UCPOP: A sound, complete, partial order planner for ADL. In *Proceedings of the Third International Conference on Knowledge Representation and Reasoning* (Cambridge, Massachusetts, USA, Oct. 1992), pp. 103–114.
- [217] Pereira, Ramon Fraga, Oren, Nir, and Meneguzzi, Felipe. Landmark-based heuristics for goal recognition. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence* (San Francisco, CA, USA, 2017), pp. 3622–3628.
- [218] Perovšek, Matic, Vavpetič, Anže, Cestnik, Bojan, and Lavrač, Nada. A wordification approach to relational data mining. In *Discovery Science*, Johannes Fürnkranz, Eyke Hüllermeier, and Tomoyuki Higuchi, Eds., vol. 8140 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2013, pp. 141–154.
- [219] Perovšek, Matic, Anže Vavpetič, Anze, Kranjc, Janez, Cestnik, Bojan, and Lavrač, Nada. Wordification: Propositionalization by unfolding relational data into bags of words. *Expert Systems with Applications* 42, 17-18 (2015), 6442–6456.
- [220] Petrick, Ronald P. A., and Bacchus, Fahiem. Extending the knowledge-based approach to planning with incomplete information and sensing. In *Proceedings of the The Fourteenth International Conference on Automated Planning and Scheduling* (Whistler, British Columbia, Canada, 2004), pp. 2–11.
- [221] Petrick, Ronald P. A., and Foster, Mary Ellen. Planning for social interaction in a robot bartender domain. In *Proceedings of the International Conference on Automated Planning and Scheduling Special Track on Novel Applications* (Rome, Italy, June 2013), pp. 389–397.
- [222] Pheonixmaster1. Fire Emblem Heroes F2P Guides! https://www.youtube.com/playlist?list=PL-UHDqcwcC4j_3RB1MqiMN-AtU-QGR16e, Feb. 2017. [YouTube] All videos in playlist uploaded by Pheonixmaster1.

- [223] Pnueli, Amir, and Rosner, Roni. On the synthesis of a reactive module. In *Proceedings of the Sixteenth ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages* (Austin, Texas, USA, 1989), POPL '89, Association for Computing Machinery, pp. 179–190.
- [224] Poo Hernandez, Sergio, Bulitko, Vadim, and Spetch, Marcia. Keeping the player on an emotional trajectory in interactive storytelling. In *Proceedings of the Eleventh Artificial Intelligence and Interactive Digital Entertainment Conference* (2015), pp. 65–71.
- [225] Rabin, Steve, and Sturtevant, Nathan R. Combining bounding boxes and jps to prune grid pathfinding. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence* (2016), pp. 746–752.
- [226] Ramírez, Miquel, and Geffner, Hector. Plan recognition as planning. In *Proceedings of the Twenty-first International Joint Conference on Artificial Intelligence* (Pasadena, California, USA, 2009), pp. 1778–1783.
- [227] Ramírez, Miquel, and Geffner, Hector. Probabilistic plan recognition using off-the-shelf classical planners. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence* (Atlanta, Georgia, USA, 2010), pp. 1121–1126.
- [228] Ramírez, Miquel, and Geffner, Hector. Goal recognition over POMDPs: Inferring the intention of a POMDP agent. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence* (Barcelona, Catalonia, Spain, 2011), pp. 2009–2014.
- [229] Renz, Jochen, Ge, XiaoYu (Gary), Zhang, Peng, and Stephenson, Matthew. AI Birds.org - Angry Birds AI Competition. <https://aibirds.org/>, 2012–2017. [Online; last accessed 14-December-2017].
- [230] Ricci, Francesco, Rokach, Lior, Shapira, Bracha, and Kantor, Paul B. *Recommender Systems Handbook*, 1st ed. Springer-Verlag, Berlin, Heidelberg, 2010.
- [231] Riedl, Mark O., and Young, R. Michael. Narrative planning: Balancing plot and character. *Journal of Artificial Intelligence Research* 39, 1 (Sept. 2010), 217–268.
- [232] Riek, Laurel D. Wizard of Oz studies in HRI: A systematic review and new reporting guidelines. *Journal of Human-Robot Interaction* 1, 1 (July 2012), 119–136.
- [233] Rieping, Kristin, Englebienne, Gwenn, and Kröse, Ben. Behavior analysis of elderly using topic models. *Pervasive and Mobile Computing* 15, 0 (2014), 181–199.
- [234] Roll, Ido, Aleven, Vincent, McLaren, Bruce M., and Koedinger, Kenneth R. Improving students' help-seeking skills using metacognitive feedback in an intelligent tutoring system. *Learning and Instruction* 21, 2 (2011), 267–280.

- [235] Rosenthal, Stephanie, and Veloso, Manuela. Mobile robot planning to seek help with spatially-situated tasks. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence* (Toronto, Ontario, Canada, 2012), AAAI'12, AAAI Press, pp. 2067–2073.
- [236] Rudin, Cynthia. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence* 1 (May 2019), 206–215.
- [237] Russell, Stuart, and Norvig, Peter. *Artificial Intelligence: A Modern Approach*, 3 ed. Prentice Hall Press, 2009.
- [238] Sanner, Scott. Relational dynamic influence diagram language (RDDL): Language description. *Technical Report, Australian National University* (2010).
- [239] Schaal, Stefan. Learning from demonstration. In *Proceedings of the Ninth International Conference on Neural Information Processing Systems* (Denver, Colorado, USA, 1996), NIPS'96, MIT Press, pp. 1040–1046.
- [240] Scheutz, Matthias. The case for explicit ethical agents. *AI Magazine* 38, 4 (Dec. 2017), 57–64.
- [241] Schwenk, Markus, Vaquero, Tiago, and Nejat, Goldie. Schedule-based robotic search for multiple residents in a retirement home environment. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence* (Quebec City, Quebec, Canada, 2014), pp. 2571–2577.
- [242] Segovia-Aguas, Javier, Jiménez, Sergio, and Jonsson, Anders. Hierarchical finite state controllers for generalized planning. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence* (New York, New York, USA, 2016), pp. 3235–3241.
- [243] Shah, Julie, and Breazeal, Cynthia. An empirical analysis of team coordination behaviors and action planning with application to human-robot teaming. *Human Factors* 52, 2 (2010), 234–245.
- [244] Shoham, Yoav, and Tennenholtz, Moshe. On the emergence of social conventions: modeling, analysis, and simulations. *Artificial Intelligence* 94, 1 (1997), 139–166. Economic Principles of Multi-Agent Systems.
- [245] Shotton, Jamie, Fitzgibbon, Andrew, Cook, Mat, Sharp, Toby, Finocchio, Mark, Moore, Richard, Kipman, Alex, and Blake, Andrew. Real-time human pose recognition in parts from a single depth image. In *Proceedings of the Twenty-Fourth IEEE Conference on Computer Vision and Pattern Recognition* (Colorado Springs, Colorado, USA, 2011), pp. 1297–1304.
- [246] Shvo, Maayan, and McIlraith, Sheila A. Active goal recognition. In *Proceedings of the Thirty-fourth Conference on Artificial Intelligence* (2020).

- [247] Simonite, Tom. Why this might be the Model T of workplace robots. *MIT Technology Review*, <http://www.technologyreview.com/news/520456/why-this-might-be-the-model-t-of-workplace-robots/>, Oct. 2013. [Online; posted 21-October-2013].
- [248] Singh, Jagbir. A characterization of positive poisson distribution and its statistical application. *SIAM Journal on Applied Mathematics* 34, 3 (1978), 545–548.
- [249] Singhal, Amit. Modern information retrieval: A brief overview. *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering* 24, 4 (2001), 35–43.
- [250] Smith, David E., and Weld, Daniel S. Conformant graphplan. In *Proceedings of the Fifteenth AAAI Conference on Artificial Intelligence* (Madison, Wisconsin, USA, 1998), pp. 889–896.
- [251] Smith, Mat. Are you ready for your first home robot? Meet Pepper. Engadget, <http://www.engadget.com/2014/06/12/home-robot-pepper/>, June 2014. [Online; posted 12-June-2014].
- [252] Sohrabi, Shirin, Riabov, Anton V., and Udrea, Octavian. Plan recognition as planning revisited. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence* (New York, New York, USA, July 2016), pp. 3258–3264.
- [253] Sondik, E. J. *The Optimal Control of Partially Observable Markov Processes*. PhD thesis, Stanford University, 1971.
- [254] Song, Young Chol, Kautz, Henry, Allen, James, Swift, Mary, Li, Yuncheng, Luo, Jiebo, and Zhang, Ce. A Markov logic framework for recognizing complex events from multimodal data. In *Proceedings of the Fifteenth ACM on International Conference on Multimodal Interaction* (Sydney, Australia, 2013), pp. 141–148.
- [255] Srivastava, Siddharth, Fang, Eugene, Riano, Lorenzo, Chitnis, Rohan, Russell, Stuart, and Abbeel, Pieter. Combined task and motion planning through an extensible planner-independent interface layer. In *IEEE International Conference on Robotics and Automation* (May 2014), pp. 639–646.
- [256] Srivastava, Siddharth, Immerman, Neil, and Zilberstein, Shlomo. Learning generalized plans using abstract counting. In *Proceedings of the Twenty-Sixth Conference on Artificial Intelligence* (Chicago, Illinois, USA, 2008), pp. 991–997.
- [257] Srivastava, Siddharth, Immerman, Neil, and Zilberstein, Shlomo. Applicability conditions for plans with loops: Computability results and algorithms. *Artificial Intelligence* 191 (2012), 1–19.

- [258] Srivastava, Siddharth, Russell, Stuart J., and Pinto, Alessandro. Metaphysics of planning domain descriptions. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence* (Phoenix, Arizona, USA, 2016), pp. 1074–1080.
- [259] Steedman, Mark. *Surface Structure and Interpretation*. Linguistic Inquiry Series. MIT Press, 1996.
- [260] Steyvers, Mark, and Griffiths, Tom. Probabilistic topic models. In *Latent Semantic Analysis: A Road to Meaning*, T. Landauer, S. Dennis McNamara, and W. Kintsch, Eds. Laurence Erlbaum, 2007.
- [261] Stocky, Tom, Faaborg, Alexander, and Lieberman, Henry. A commonsense approach to predictive text entry. In *The ACM SIG International Conference on Computer-Human Interaction Extended Abstracts on Human Factors in Computing Systems* (Vienna, Austria, 2004), pp. 1163–1166.
- [262] Su, Xing, Tong, Hanghang, and Ji, Ping. Activity recognition with smartphone sensors. *Tsinghua Science and Technology* 19, 3 (June 2014), 235–249.
- [263] Sukthankar, Gita, Geib, Christopher, Bui, Hung H., Pynadath, David, and Goldman, Robert P. *Plan, Activity, and Intent Recognition: Theory and Practice*. Elsevier Science, Waltham, MA, USA, 2014.
- [264] Sung, Ja-Young, Christensen, Henrik I., and Grinter, Rebecca E. Robots in the wild: Understanding long-term use. In *Proceedings of the Fourth ACM/IEEE International Conference on Human Robot Interaction* (La Jolla, California, USA, 2009), HRI ’09, Association for Computing Machinery, pp. 45–52.
- [265] Sutton, Richard S., and Barto, Andrew G. *Introduction to Reinforcement Learning*. MIT Press, Cambridge, MA, USA, 1998.
- [266] Synnaeve, Gabriel, and Bessière, Pierre. A bayesian model for plan recognition in rts games applied to starcraft. In *Proceedings of the Seventh AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment* (Palo Alto, California, USA, 2011), AIIDE’11.
- [267] Tastan, Bulent, Chang, Yuan, and Sukthankar, Gita. Learning to intercept opponents in first person shooter games. In *Proceedings of the IEEE Conference on Computational Intelligence in Games* (Granada, Spain, sep 2012), pp. 100–107.
- [268] Tastan, Bulent, and Sukthankar, Gita. Learning policies for first person shooter games using inverse reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence for Interactive Digital Entertainment* (Palo Alto, California, USA, 2011), pp. 85–90.
- [269] Team, Doodle. Home — Doodle. <https://doodle.com/en/>, 2007. [Online] Last accessed 2020 March 6.

- [270] Teevan, Jaime, Iqbal, Shamsi T., and Von Veh, Curtis. Supporting collaborative writing with microtasks. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (2016), pp. 2657–2668.
- [271] Tellex, Stefanie, Knepper, Ross, Li, Adrian, Rus, Daniela, and Roy, Nicholas. Asking for help using inverse semantics. In *Proceedings of Robotics: Science and Systems* (Berkeley, USA, July 2014).
- [272] Thomaz, Andrea, Hoffman, Guy, and Cakmak, Maya. Computational human-robot interaction. *Foundations and Trends in Robotics* 4, 2-3 (Dec. 2016), 105–223.
- [273] Thrun, Sebastian, Bennewitz, Maren, Burgard, Wolfram, Cremers, Armin B., Dellaert, Frank, Fox, Dieter, Hähnel, Dirk, Rosenberg, Charles, Roy, Nicholas, Schulte, Jamieson, and Schulz, Dirk. Minerva: A second-generation museum tour-guide robot. In *Proceedings of the 1999 IEEE International Conference on Robotics and Automation* (Detroit, Michigan, USA, 1999), vol. 3, pp. 1999–2005.
- [274] Trinh, Ha, Waller, Annalu, Vertanen, Keith, Kristensson, Per Ola, and Hanson, Vicki L. Phoneme-based predictive text entry interface. In *Proceedings of the Sixteenth International ACM SIG Conference on Computers & Accessibility* (Rochester, New York, USA, 2014), pp. 351–352.
- [275] Trnka, Keith, McCaw, John, Yarrington, Debra, McCoy, Kathleen F., and Pennington, Christopher. User interaction with word prediction: The effects of prediction quality. *ACM Transactions Accessible Computing* 1, 3 (2009), 17:1–17:34.
- [276] Ulanoff, Lance. Google’s project Tango: Old ideas, new packaging. Mashable, <http://mashable.com/2014/02/21/google-project-tango-op-ed/>, Feb. 2014. [Online; posted 21-February-2014].
- [277] Unhelkar, Vaibhav V., Pérez-D’Arpino, Claudia, Stirling, Leia, and Shah, Julie A. Human-robot co-navigation using anticipatory indicators of human walking motion. In *IEEE International Conference on Robotics and Automation* (May 2015), pp. 6183–6190.
- [278] Urmson, Chris, and Whittaker, William L. Self-driving cars and the urban challenge. *IEEE Intelligent Systems* 23, 2 (2008).
- [279] van den Briel, Menkes, Benton, J., Kambhampati, Subbarao, and Vossen, Thomas. An LP-based heuristic for optimal planning. In *Principles and Practice of Constraint Programming* (Berlin, Heidelberg, 2007), Christian Bessière, Ed., Springer Berlin Heidelberg, pp. 651–665.

- [280] Vattam, Swaroop, Klenk, Matthew, Molineaux, Matthew, and Aha, David W. Breadth of approaches to goal reasoning: A research survey. In *Goal Reasoning: Papers from the ACS Workshop* (College Park, Maryland, USA, Dec. 2013), pp. 111–126.
- [281] Vered, Mor, and Kaminka, Gal A. Heuristic online goal recognition in continuous domains. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence* (Melbourne, Australia, 2017), pp. 4447–4454.
- [282] Verve. Mia’s AI Manipulation Instructional Academy. <https://vervefeh.github.io/FEH-AI/>, July 2018. [Online] Last accessed 2020 February 16.
- [283] Wang, Yang, and Mori, Greg. Human action recognition by semi-latent topic models. *IEEE Transactions on Pattern Analysis and Machine Intelligence Special Issue on Probabilistic Graphical Models in Computer Vision* 31, 10 (2009), 1762–1774.
- [284] Wayllace, Christabel, Hou, Ping, Yeoh, William, and Son, Tran Cao. Goal recognition design with stochastic agent action outcomes. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence* (New York, New York, USA, 2016), IJCAI’16, AAAI Press, pp. 3279–3285.
- [285] Weld, Daniel S. An introduction to least commitment planning. *AI Magazine* 15, 4 (1994), 27–61.
- [286] Westlund, Jacqueline Kory, Lee, Jin Joo, Plummer, Luke, Faridi, Fardad, Gray, Jesse, Berlin, Matt, Quintus-Bosz, Harald, Hartmann, Robert, Hess, Mike, Dyer, Stacy, dos Santos, Kristopher, Örn Adalgeirsson, Sigurdur, Gordon, Goren, Spaulding, Samuel, Martinez, Marayna, Das, Madhurima, Archie, Maryam, Jeong, Sooyeon, and Breazeal, Cynthia. Tega: A social robot. In *Proceedings of the Eleventh Annual ACM/IEEE International Conference on Human-Robot Interaction* (Christchurch, New Zealand, 2016), p. 561.
- [287] Wray, Kyle H., Witwicki, Stefan J., and Zilberstein, Shlomo. Online decision-making for scalable autonomous systems. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence* (2017), pp. 4768–4774.
- [288] Wray, Kyle Hollins, Pineda, Luis, and Zilberstein, Shlomo. Hierarchical approach to transfer of control in semi-autonomous systems. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence* (New York, New York, USA, 2016), pp. 517–523.
- [289] Wray, Kyle Hollins, and Zilberstein, Shlomo. Multi-objective POMDPs with lexicographic reward preferences. In *Proceedings of the Twenty-Fourth International Conference on Artificial Intelligence* (Buenos Aires, Argentina, 2015), IJCAI’15, AAAI Press, pp. 1719–1725.

- [290] Xue, Su, Wu, Meng, Kolen, John, Aghdaie, Navid, and Zaman, Kazi A. Dynamic difficulty adjustment for maximized engagement in digital games. In *Proceedings of the Twenty-Sixth International Conference on World Wide Web Companion* (Perth, Australia, 2017), WWW ’17 Companion, International World Wide Web Conferences Steering Committee, pp. 465–471.
- [291] Yang, Qiang. Activity recognition: Linking low-level sensors to high-level intelligence. In *Proceedings of the Twenty-First International Joint Conference on Artificial Intelligence* (Pasadena, California, USA, July 2009), pp. 20–25.
- [292] Yoon, Sung Wook, Fern, Alan, and Givan, Robert. FF-Replan: A baseline for probabilistic planning. In *Proceedings of the Seventeenth International Conference on Automated Planning and Scheduling* (Providence, Rhode Island, 2007), pp. 352–359.
- [293] Younes, Håkan L. S., and Littman, Michael L. PPDDL1.0: An extension to PDDL for expressing planning domains with probabilistic effects. Tech. Rep. CMU-CS-04-162, Carnegie Mellon University, Pittsburgh, PA, USA, 2004.
- [294] Young, James, Ishii, Kentaro, Igarashi, Takeo, and Sharlin, Ehud. Style by demonstration: Teaching interactive movement style to robots. In *Proceedings of the Seventeenth ACM International Conference on Intelligent User Interfaces* (Lisbon, Portugal, 2012), pp. 41–50.
- [295] Zhang, Bin, Nakamura, Tomoaki, and Kaneko, Masahide. Adaptive fusion of multi-information based human identification for autonomous mobile robot. In *Proceedings of the Twenty-fourth IEEE International Symposium on Robot and Human Interactive Communication* (Kobe, Japan, 2015).
- [296] Zhang, Hao, and Parker, Lynne E. 4-dimensional local spatio-temporal features for human activity recognition. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems* (San Francisco, California, USA, 2011), pp. 2044–2049.
- [297] Zhao, Shengdong, Nakamura, Koichi, Ishii, Kentaro, and Igarashi, Takeo. Magic cards: A paper tag interface for implicit robot control. In *Proceedings of the ACM SIG Conference on Human Factors in Computing Systems* (Boston, Massachusetts, USA, 2009), pp. 173–182.
- [298] Zilberstein, Shlomo. Using anytime algorithms in intelligent systems. *AI Magazine* 17, 3 (1996), 73–83.
- [299] Zilberstein, Shlomo. Building strong semi-autonomous systems. In *Proceedings of the Twenty-Ninth Conference on Artificial Intelligence* (Austin, Texas, 2015), pp. 4088–4092.

- [300] Zilberstein, Shlomo, and Russell, Stuart J. Anytime sensing, planning and action: A practical model for robot control. In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence* (Chambery, France, 1993), pp. 1402–1407.
- [301] Zook, Alexander, and Riedl, Mark O. Automatic game design via mechanic generation. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence* (2014), pp. 530–536.