Language Assisted Co-Planning
Research Comps Proposal

Stephen Brawner

Dec 13, 2013

1 Introduction

Recent developments in household robotics, and the emergence of new personal robotic platforms show encouraging progress toward automating less desired household tasks, like organization, cleaning, kitchen preparation, and household maintenance. Effective communication with robots will help reduce the burden required to manage these tasks. Natural language commands as a communication methodology are often ambiguous and insufficiently clear. Misunderstanding of language often requires the human user to repeat failed commands or if the capability is present, disambiguate through dialogue. This proposal seeks to make a robot capable of reasoning about goals and infer helpful actions without requiring overly specific and explicit instructions.

We are interested in the problem of creating a robotic agent that can helpfully interact with a human being. We conceptualize the overall problem of consisting of two subproblems, a novel co-planning problem and a language understanding problem. We present "co-planning" as an extension of the classical planning problem by defining a set of agents, each with its own operator set. The planning method will find an optimal strategy that minimizes the collective number of operators performed by all agents.

For large state-spaces, end-to-end planning may be impossible to solve in a reasonable amount of time, but by dividing the task into subgoals we could potentially expand the scale of plannable problems. Our approach to determining subgoals involves parsing corpora of natural language instructions into a logical form from which we can directly create plannable subgoals. Because natural language understanding is still an open problem, we do not rely on perfect annotations and instead assume that planning will be able to resolve understanding errors.

This document describes a proposed project for developing methods of increasing robots' assistive behavior in the context of a cooperative cooking task. The proposed method will leverage knowledge gained by parsing large corpora of recipes to infer helpful actions when working alongside a human user. This project will evaluate the proposed method in the context of an interactive culinary game and compare the performance of our robot cooperating with a human against a human working alone and a human cooperating with another human. We expect the time needed to accomplish the task to be significantly less than the human working by itself and approach the speed by which two people work together.

2 Related Work

Communication among two human individuals cooperating in a task strives to establish a common ground of understanding with the least amount effort as possible [1]. Grice [2] introduced, among his many maxims, the idea that people communicate by giving as much information as is necessary, and no more. To interact effectively with people, robots will need to infer details in communication that were not given by a person but instead thought to be obvious or implied. Vogel et al. [3, 4] have demonstrated the emergence of language implicature and Gricean Maxims by incorporating POMDPs and Decentralized POMDPs into their language

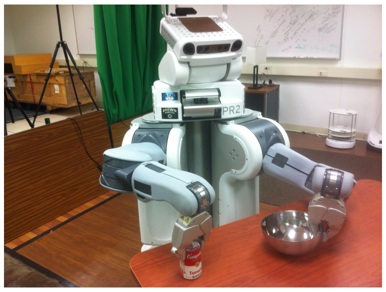

Figure 1: Willow Garage PR2 manipulating a tomato soup can and bowl

model. Language understanding and grounding is heavily dependent on the domain in which it occurs [1], therefore our method will seek to incorporate understanding of cooking from annotated collections of recipes.

The application of robotic technology to household tasks like baking [5], towel folding [6], and sock sorting [7] illustrate the growing suitability of robots to reducing burdensome chores at home. As impressive as these different demonstrations are, they require strict organization of the environment to simplify the perception and manipulation challenges presented in their different domains. Failures without the possibility of graceful recovery are inevitable in less structured environments. As shown by Tellex et al. [8], human-dependent recovery modes are a solution to improving the utility of robots in the face of failures in autonomy and uncertain environments. In our approach, we will make use of user modeling to aid planning through a series of difficult tasks.

Natural language interaction with robots remains a challenging problem, but there exists several promising approaches [9, 10, 11]. These methods have so far shown only direct grounding between language and actions. In settings where a robot serves as an assistant to be instructed, for example as an assistant in a kitchen setting, highly specific commands required by these methods can easily become overly burdensome for human partners. In this research, we will incorporate observation to improve goal inference and extend the utility of natural language approaches.

Model-based approaches that attempt to infer user desires and intentions are useful in a cooperative setting [12, 13]. These works demonstrate model-based interaction on small scale, closed-domain tasks. To demonstrate these methods in a cooperative cooking cooking task, our method will not only utilize a large amount of culinary knowledge parsed from annotated recipes, it will also seek to infer the goals of the task through observation.

3 Method

This work addresses the problem of creating an agent, ultimately a robotic agent, to helpfully interact with a human being. We conceptualize the overall problem of consisting of two subproblems, a novel co-planning problem and a language understanding problem.

We present "co-planning" as an extension of the classical planning problem. A standard planning problem consists of a state space, often defined implicitly via a set of objects and propositions, and an operator space. Operators transform states to states at some cost. The designer provides an initial state and set of one or more states that constitutes the goal to the planner. The planner finds a low-cost sequence of operators that transform the initial state into a goal state.

The co-planning problem extends the planning problem by defining a set of agents, each with its own operator set. In the centralized version, the co-planner generates a sequence of operators, selected round-robin from the set of agents, that transform the initial state into a goal state. In the decentralized version, the agents select, in round-robin fashion an operator that must factor in the choices made by the other agents. An optimal strategy minimizes the collective number of operators performed by all agents. Suboptimal strategies, where agents' actions are not completely cooperative, will result in costlier plans.

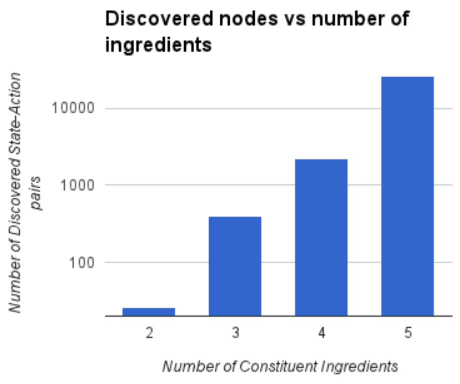

Figure 2: The factorial expansion of the required states to explore as the number of constituent ingredients increases

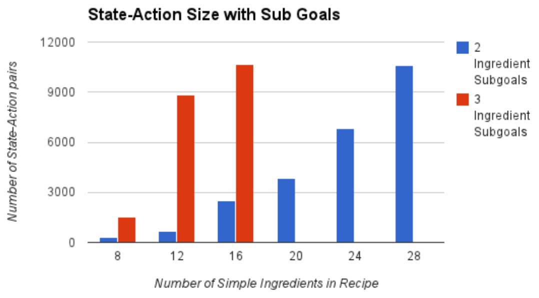

Figure 3: The expansion of the required states using subgoals with two ingredients and three ingredients per subgoal

In the asynchronous version, operators have durations in addition to costs and an agent selects an action when its previous action has been completed instead of a strict round-robin order. The optimal asynchronous strategy will minimize the time to reach the goal state.

In this work, we consider algorithms for the centralized version, but we evaluate them in a decentralized setting with live human beings playing the roles of the other agents.

We assume access to a task interpreter that maps natural language sentences to subgoal specifications. We train the task interpreter using labeled training data—a list of natural language sentences and their correct interpretation as subgoals. In this work, we interpret a sequence of subgoals as a sequence of separate (co-)planning problems. The initial states for each subgoal are the subgoal end states for the previous subgoal plan. In reality, no natural language interpreter is perfectly accurate, therefore the planning problem we solve must account for natural language failures.

We formulate the problem of mapping natural language task specifications to co-plans as the composition of the two prior problems. Mapping natural language instructions to subgoals will simplify planning through a complex problem, and planning itself allows for circumnavigation of issues in language understanding.

3.1 Co-planning in the Culinary Domain

We have chosen to illustrate this co-planning problem as two agents, a robot and a human, cooperating to cook a recipe in a household kitchen. The set of operators or actions available to the robot are a subset of those available to a human. At each turn, both agents select an action, with the goal of arriving at the completed task minimizing the number of turns taken together.

To plan actions in the culinary domain, we created an Object-Oriented Markov Decision Process (OO-MDP) [14, 15] model using the Brown-UMBC Reinforcement Learning and Planning (BURLAP) framework [16]. We represent the state space of an OO-MDP as a collection of objects and their attribute values. Actions change values of these attributes and can remove or create instances of the objects. Table 1 lists the current objects and their associated attributes in our culinary domain and Table 2 lists the current actions.

In this OO-MDP model, the number of possible states is exceedingly large and difficult to plan through. Relational attributes like CONTAINS, which can have an arbitrary amount of ingredient parameters, have a combinatorial explosion as the number of containers and ingredients increases. Actions with multiple possible parameters like MOVE and POUR grow quadradically with the number of ingredients. Figure 2 shows the expansion of nodes in the planning tree visited as the number of constituent ingredients in a recipe increases, which grows with actions raised to the power of states, $|N| = |A|^{|S|}$. Any undirected search algorithm will fail to find a solution in a reasonable amount of time to a non-trivial recipe.

Objects	Discrete Attributes			Relational Attributes (Parameter)	
Ingredient	MIXED	MELTED	BAKED	CONTAINER(C)	
Container	MIXING	HEATING	BAKING	CONTAINS(I_1, ... , I_n)	SPACE(S)
Space	WORKING	HEATING	BAKING	CONTAINS(C_1, ... , C_n)	

Table 1: OO-MDP Objects and attributes in the Culinary State space

Actions	Parameters		Preconditions
Move	Container C	Space S	C not in S
Pour	Container C_1	Container C_2	C_1 not empty
Mix	Container C		C not empty, C is MIXING
Bake	Space S		S is BAKING, all Containers in S are BAKING

Table 2: OO-MDP Actions and Action parameters in the Culinary State space

To plan a complex recipe, we solve this problem by subdividing it into subgoals of simpler ingredients, which has a much more manageable growth, shown in figure 3. For example, to make brownies, a batter must be made. To make the batter, we might believe that first the dry ingredients are to be mixed in one bowl, and the wet ingredients are to be mixed in another. Planning over all possible actions on all possible objects results in a high dimensional, and intractable state space. By planning over only the ingredients required for a subgoal ingredient, planning to that subgoal will greatly simplify the computation required. Table 3 contains a list of actions generated from the planning framework to make a subgoal in a gnocchi recipe, salted water. The plan as generated, assumes all ingredients are *mise en place*. That is, all required ingredients have been pre-measured and placed in bowl near the working space.

Plans generated through the previously described methods will seek to minimize the amount of unnecessary actions as well as waiting time required by the human or robot. We plan to use a cost function associated with time steps. At each time step, any agents in the environment will take one action. Plans that require agents to wait unnecessarily or perform counter productive actions will require more time steps to complete and be more costly overall.

3.2 Parsing and Training on a Recipe Corpus

This project will make use of the University of Washington's Semantic Parsing Framework (SPF) [11] to convert a large corpus of instruction based recipes into a semantic logical form representation of subgoals. We train the framework on a collection annotated recipes, a list of recipe steps and their representative logical form. With SPF, this logical form will consist of predicates, noun phrases, and verb phrases generalizable to the cooking domain so that it can adequately capture the language provided in culinary recipes and general cooking dialogue. The logical form must also be possible to map to the planning framework, so that useful sets of actions can be generated.

Action	Agent	Parameter 1	Parameter 2
Move	robot	water_bowl	counter
Pour	human	water_bowl	mixing_bowl_1
Move	robot	salt_bowl	counter
Pour	human	salt_bowl	mixing_bowl_1
Mix	human	mixing_bowl_1	

Table 3: Generated Burlap plan to make salted water with a robot and human agent

English Phrase	SPF Logical Form
salted water	$\lambda x.is(x, mixed(salt, water))$
large pot of salted water	$\lambda x.\lambda y.container(x) \wedge is(x, mixed(salt, water))$

Table 4: Examples of annotations from recipe text to a logical form compatible with SPF

English Sentence	Bring a large pot of salted water to a boil
Lamda Calculus	$\lambda p.\lambda c.\lambda i.heat(p, c) \wedge is(i, mixed(`salt', `water')) \wedge in(i, c) \wedge postAction(boiling(i))$

SPF Form	`(lambda $0:p (lambda $1:c (lambda $2:in`
	`  (and:<t*,t> (heat:<<<a,p>,e>,t> $0 $1)`
	`    (is:<e,<e,t>> $2`
	`      (mixed:<in*,t> salt:in water:in))`
	`    (in:<in,<c,t>> $2 $1)`
	`    (post:<a,<e,t>>(boiling:<in,t> $2))))))`

Table 5: Conversion from english sentence to lambda calculus logical form and SPF recognized representation. The variables for the lambda calculus variables p, c, i have types person, container, ingredient respectively.

We have annotated a small number of example recipes so far. The logical form consists of a long list of predicates, types and constants. Table 4 illustrates a few examples of annotations from a simple gnocchi recipe.

Regardless of the capabilities of our chosen task interpreter, we still encounter phrases that will be unlikely to be parsed correctly. One step in the gnocchi example we are using asks to "shape small portions of the dough into long 'snakes.'" Unless there are many such examples in our annotated corpus, the correct predicates to specify 'snakes' of dough will not exist.

3.3 Connecting Subgoals and Recipes to Language Propositions

To compose the problems of mapping natural language task specifications to co-plans, we will match predicates that specify subgoal-like ingredients with propositional functions in the planning framework. Matching predicates and propositional functions in this manner will allow for an expandable planning framework to incorporate a large number of recipes and cooking plans.

In table 5 shows the annotated form of the sentence "bring a large pot of salted water to a boil." After stripping away less descriptive predicates like 'and', 'is', and 'in', what remains is a simplified form that can be converted into generalized code as shown in table 6.

To generate plans on tasks for which annotations do not exist, we will use the annotated recipes as a training set to infer subgoal plans with the SPF framework. For phrases found in recipes that SPF fails to generate reasonable predicates, the subgoal plan that is generated will instead contain actions that require the human to accomplish the steps. By allowing for human dependence in generating these plans, we can incorporate a larger domain of recipes that our robot would be incapable of resolving by itself.

For example in our shaping dough into 'snakes' example, it is clear that an agent must manipulate the dough into some shape even if it is impossible to know exactly how. The robotic agent would infer from this that the human agent require use of the dough even if it does not know the required tools or method of shaping.

4 Experimental Verification

The initial demonstration of this system will be a simple screen-based game that seeks to answer the question: Can a robot knowledgeable in planning culinary recipes provide helpful actions in novel tasks or order-invariant tasks? The user will have a set of tasks to accomplish and will require assistance from the robot

Simplified SPF Form	`(heat:<<<a,p>,e>,t> $0 $1) ...`
	`    (mixed:<in*,t> salt:in water:in) ...`
	`    (post:<a,<e,t>>(boiling:<in,t> $2))`

Burlap code	`for (String subIngredient : recipe.contents) {`
	`     // "salt", "water"`
	`    if (!ingredient.contains(subIngredient)) { return false;}`
	`}`
	`return ingredient.isBoiling() = recipe.isBoiling();`

Table 6: A conversion between a simplified SPF logical form description to representative Burlap code. The provided Java code returns a boolean indicating whether the examined ingredient matches the parsed recipe.

to accomplish many of them. Through a GUI interface, the human will attempt to accomplish the tasks. A robot assistant will be capable of accomplishing a subset of the human's required actions.

This experiment will require three different modes which compare the number of time steps required to accomplish a given task. To compare the performance of our methods, we will compare a human cooperating with a robot agent against a human working alone and a human working with another human. We expect that our method will perform significantly better than a human working alone, and approach the capability of the human cooperating with another human.

5 Conclusion

By enabling robots to infer helpful actions without strict, overly specific communication, we can simplify the management of household robots. To avoid scenarios where explaining the tasks to be accomplished takes just as much time as accomplishing them by oneself, this method of inferring helpful actions will actually reduce the work load on the human users to allowing them to spend time on the activities they love.

After demonstrating our goals with an interactive cooperative game, we plan to demonstrate this work on physical robots helping in culinary tasks in a mock kitchen. Physically interacting with a robot will present a set of challenges that cannot be as well controlled as a virtual game. For example, the time required for a robot to accomplish a task could easily entice the human to accomplish more tasks alone.

To effectively offload tasks to robot assistants, these assistants need a level of autonomy that significantly improves the livelihood of their users. Only through effective communication, and an inherent capability to model user intent and desire will robots be able to serve a helpful function in personal robotics.

6 Schedule

- **12/13 Proposal presentation**
- **1/15 Dynamic re-planning with novel recipes**
- **2/15 First draft of defense paper, preliminary results**
- **3/1 Final results**
- **3/8 Final draft of defense paper submitted to faculty**
- **3/15 Defense proposal**

References

[1] Herbert H Clark and Susan E Brennan. Grounding in communication. *Perspectives on socially shared cognition*, 13(1991):127–149, 1991.

[2] H Paul Grice. Logic and conversation. *1975*, pages 41–58, 1975.

[3] Adam Vogel, Christopher Potts, and Dan Jurafsky. Implicatures and nested beliefs in approximate Decentralized-POMDPs. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, Sofia, Bulgaria, August 2013. Association for Computational Linguistics.

[4] Adam Vogel, Max Bodoia, Christopher Potts, and Dan Jurafsky. Emergence of gricean maxims from multi-agent decision theory. In *Proceedings of NAACL 2013*, 2013.

[5] Mario Bollini, Stefanie Tellex, Tyler Thompson, Nicholas Roy, and Daniela Rus. Interpreting and executing recipes with a cooking robot. In *13th International Symposium on Experimental Robotics*, 2012.

[6] Jeremy Maitin-Shepard, Marco Cusumano-Towner, Jinna Lei, and Pieter Abbeel. Cloth grasp point detection based on multiple-view geometric cues with application to robotic towel folding. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 2308–2315. IEEE, 2010.

[7] Ping Chuan Wang, Stephen Miller, Mario Fritz, Trevor Darrell, and Pieter Abbeel. Perception for the manipulation of socks. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 4877–4884. IEEE, 2011.

[8] Stefanie Tellex, Ross A. Knepper, Adrian Li, Thomas M. Howard, Daniela Rus, and Nicholas Roy. Assembling furniture by asking for help from a human partner. In *Human-Robot Interaction 2013 Workshop on Collaborative Manipulation*, 2013.

[9] Thomas Kollar, Stefanie Tellex, Matthew R. Walter, Albert Huang, Abraham Bachrach, Sachi Hemachandra, Emma Brunskill, Ashis Banerjee, Deb Roy, Seth Teller, and Nicholas Roy. Generalized grounding graphs: A probabilistic framework for understanding grounded language. *Journal of Artificial Intelligence Research*, 2013.

[10] David L Chen and Raymond J Mooney. Learning to interpret natural language navigation instructions from observations. In *AAAI*, volume 2, pages 1–2, 2011.

[11] Yoav Artzi and Luke Zettlemoyer. UW SPF: The University of Washington Semantic Parsing Framework, 2013.

[12] Jesse Gray, Cynthia Breazeal, Matt Berlin, Andrew Brooks, and Jeff Lieberman. Action parsing and goal inference using self as simulator. In *Robot and Human Interactive Communication, 2005. ROMAN 2005. IEEE International Workshop on*, pages 202–209. IEEE, 2005.

[13] Cynthia Breazeal, Jesse Gray, and Matt Berlin. An embodied cognition approach to mindreading skills for socially intelligent robots. *The International Journal of Robotics Research*, 28(5):656–680, 2009.

[14] Carlos Diuk, Andre Cohen, and Michael L Littman. An object-oriented representation for efficient reinforcement learning. In *Proceedings of the 25th international conference on Machine learning*, pages 240–247. ACM, 2008.

[15] James Macglashan and Marie desJardins. Multi-source option-based policy transfer. 2013.

[16] James Macglashan. *BURLAP: the Brown-UMBC Reinforcement Learning and Planning Java Code Library.* Brown University, Providence, RI, 2013. URL `http://burlap.cs.brown.edu`.