Advances in Cognitive Systems (2020) Submitted 5/2020; published 2020

Towards Goal Inference for Human-Robot Collaboration

Shree Gotteti GOTTETI.2@WRIGHT.EDU
Wright State University, Dayton, OH 45435

Matthew Molineaux MATTHEW.MOLINEAUX@WRIGHT.EDU
Michael Cox MICHAEL.COX@WRIGHT.EDU
Wright State Research Institute, Beavercreek, OH 45431

Abstract

Spoken natural language instructions often leave a speaker's intent underspecified or unclear. We propose a goal inference procedure that extracts user intent using natural language processing techniques. This procedure uses semantic role labeling and synonym generation to extract utterance semantics, then analyzes a task domain to infer the user's underlying goal. This procedure is designed as an extension to the MIDCA cognitive architecture that enables human-robot collaboration. In this work, we describe a conceptual model of this procedure, lay out the steps a robot follows to make a goal inference, give an example use case, and describe the procedure's implementation in a simulated environment. We close with a discussion of the benefits and limitations of this approach.We expect this procedure to improve user satisfaction with agent behavior when compared to plan-based dialogue systems.

1. Introduction

Human-robot collaborations require both agents to communicate with each other in order to complete an action or a goal. Several different mediums of communication were developed such as those based on visual control, voice control, and imitation learning. However, voice guided interactions are the most natural way of communicating with a robot (Bugmann & Pires, 2005; Holada & Pelc, 2008; Batu 2012). Self-regulated autonomous agents should have the capability of processing natural language and also be able to understand the user intent. However a lot of these systems tend to be rigid in terms of how the dialogues are interchanged. We intend to develop a system using goal inference to understand the intent. This helps the system be more flexible, natural, and intuitive as it doesn't assume that the user is communicating a plan.

We describe our goal inference procedure that utilizes natural language processing techniques including synonym generation and semantic role labeling. The procedure understands the intent of the user by performing condition checks to infer an action or a goal. Our procedure has been integrated with an existing cognitive architecture.

In this case study, we discuss the related work in Section 2. We review background information about our cognitive architecture and natural language processing methods in Section 3. In Section 4, we describe our method and implementation. We layout the steps performed by our model in

Section 5. We discuss limitations of our procedure in Section 6. Section 7 is discussion and future work.

2. Related Work

In our procedure, the robot will receive voice guided interactions from a human. For each instruction, we use goal inference to draw conclusions about the user intent. In this section, we present the most closely related research on voice guided interaction systems and goal inference.

2.1 Voice Guided Interactions in Human-Robot Collaborations

Voice guided interactions are the most intuitive form of communication in a human-robot collaboration environment (Batu 2012; Bugmann & Pires 2005; Holada & Pelc 2008). Spiliotopoulos (2013) describes spoken dialogue systems that take the user utterance and generate an appropriate response using dialogue management techniques and a logical/linguistic mapping of the utterances. Dialogue management techniques, like state-based dialogue (Aust & Oerder, 1995; McTear, 1997; McTear, 1998) depend on asking a series of questions to achieve the task. However, depending on the task, the state-based dialogue graphs can get extremely large and often leave an unpleasant experience for the user as the number of questions increases. In contrast, frame-based dialogues are not as restricted as state-based dialogue; user utterances are constrained by the task domain but not by answering the questions the system considers important. However, the predictability of what the user wants decreases as the user utterances become less restricted. Plan-based dialogue (Allen, Dzikovska, Manshadi, & Swift, 2007; Ferguson & Allen, 1993; Allen et al., 1996) techniques try to understand the user's plan and work towards achieving it. However, this increases the overall complexity of the system. This will fail if the user gives a goal instead of giving a plan. The problem with these systems is that there is a trade off between interpretability vs. human-experience.

2.2 Goal Inference

Goal Inference is a non-trivial task, especially in dynamic environments where an agent must update its plans in response to an anomaly/exception or user instruction. Baker, Tenenbaum, and Saxe (2007) demonstrated goal inference using an *Inverse Planning Framework (IPF)*. IPF infers the agent's intent based on a set of observations of the agent's behaviors. In their multi-agent model, agents are assumed to be finding an optimal solution to a Markov Decision Problem (MDP). *MDP* is a standard formalism for planning in dynamic environments. Their model is an inversion of probabilistic planning in MDP through which the user derives the agent's inferred goal based on the series of actions committed by the agents. For their experiment, human subjects were asked to watch 2D intelligent agents and predict the most likely action/goal that the agent would choose based on the current stimulus. Although there was no collaboration between human and robotic agents, their work lays out the first quantitative tests on the aspect of goal inference.

Another approach to goal inference was demonstrated by Bordallo, Previtali, Nardelli, and Ramamoorthy (2016). Their model achieves goal inference through counterfactual reasoning in a dynamic environment with multiple agents moving at different velocities. They incorporated sim-

ulated prior probabilities of agents traveling at a certain velocity with the classical goal planning (solving MDP) using Bayesian inference. In other words, their model uses Bayesian recursive estimation to create a goal space containing all possible goals/actions based on a set of observations. However, for this procedure we would require prior knowledge of likely goals which may not be a trivial calculation depending on the domain. Their results indicate their efficiency in predicting the trajectory of an agent. However, their experiments included human subjects walking to avoid clashing with agents; furthermore, the agents don't interact with humans through spoken natural language or text.

A similar Bayesian inference based goal inference model to improve both objective and perceived performance of agents in human-robot collaborations was put forth by Liu, et al., (2018). Tasks can be given to an individual or collaborative and are spread across the environment where the agents move. This model has three kinds of artificial agents: fixed, reactive and Bayesian-predictive. The fixed agent doesn't have any idea of the human agent's goal. The reactive agent quickly moves to a (pre-arranged) collaborative task point if the human agent reaches it before the artificial agent. The Bayesian-predictive agent is able to predict the future position of the human agent using Bayesian inference and thereby heading to the position of a collaborative task before the human reaches it. Similar to the above models, this also does not collaborate with human agents on a natural language level.

3. Background

Our procedure has been integrated with a cognitive architecture (described below). Once an instruction is received by a robot, it goes through its cognitive phases to infer a goal. This procedure heavily relies on natural language processing techniques and checking preconditions on operators. We use semantic role labeling to extract the semantic meaning of the instructions and Wordnet for synonym generation. In this section, we present background information on the MIDCA, operators in a domain, Wordnet and SRL.

3.1 The Metacognitive Integrated Dual Cycle Architecture

The *Metacognitive Integrated Dual Cycle Architecture (MIDCA)* is a cognitive architecture incorporating both cognition and metacognition cycles (Cox et al., 2016). One function of MIDCA's cycles is to allow an autonomous agent to respond robustly to unforeseen events in a dynamic environment. To do so, MIDCA agents use a note-assess-guide procedure (Anderson & Perlis, 2005; Cox, Oates, Paisner, & Perlis, 2012) that monitors for conflicts and assess the noted conflicts to generate responses that are carried out by the agent. The metacognition cycle's monitor-control loop notes decisions made at the cognitive level and responds to them; the cognitive loop notes conflicts between environment expectations and observations and responds to them. We will not be discussing the metacognition cycle any further, as it is beyond the scope of this paper. At the Cognitive Level or the Object level, the action-perception loop observes the world in the *"perceive"* phase and comprehends it in the *"interpret"* and *"evaluate"* phases. During evaluation, a MIDCA agent formulates new goals and updates existing goals. In the problem-solving phases *"intend"* and *"plan"*, a MIDCA agent selects one or more goals to achieve immediately, then generates a feasible

plan of actions or tasks to accomplish those goals. These actions are carried out in the *"act"*phase of the action-perception loop.

3.2 Domain Operators

In automated planning domain models (Ghallab, Nau, and Traverso 2004), the state (s) of the world is represented by literals that describe relationships between objects in the world. Given a goal***(g)***, a planner generates a plan *(π)*, which is a series of actions to be carried out in order to achieve the goal. A planning operator (**o**) describes legal actions (α) in that domain. Every operator has a head *(head(**o**))* consisting of its name and arguments, a set of preconditions *(pre(**o**))* and a set of *postconditions (post(**o**))*. If the preconditions of the associated operator are met in the current state ($s \models pre(\alpha)$), an action is executable in the state (s).

For example, consider stacking two blocks in the construction domain. Operator *"stack"* has two arguments: top-block and bottom-block. The preconditions of *"stack"* are that the agent must be holding the top block; the bottom block must not have anything on it already. The postconditions are that the top block is on the bottom block. Once we define these conditions, an operator is formalized as follows:

Operator: $stack(top\text{-}block, bottom\text{-}block)$
Preconditions: $holding(top\text{-}block) \wedge clear(bottom\text{-}block)$
Postconditions: $on(top\text{-}block, bottom - block)$

3.3 Wordnet

Wordnet (Miller, 1995) is an Online Lexical Database for English. It represents each word using syntactic contexts and semantic relations. Syntactic contexts capture the word in one of four forms: noun; verb; adjective; and adverb. Semantic relations link words together by using the *sense* or meaning of the word and depend on the syntactic context. Types of semantic relations include Synonymy (similar), Antonymy (opposite), Hyponymy (subordinate), Meronymy (part), Troponymy (manner), and Entailment (consequential). Synsets are sets of synonymous words, as determined by Wordnet's *Synonymy* relation.

3.4 Semantic Role Labeling

A *semantic role* (i.e., conceptual relation) describes a relationship between a non-verb word or phrase to an action verb in the same sentence. A few examples of common semantic roles are Agent, which describes the initiator of an action; Patient, which describes an object affected by the action; and Location, which describes where an action takes place. In *semantic role labeling (SRL)* (Kogan et al., 2009), a sentence is mapped to underlying *predicate argument structures (PAS)* that match each clause. A PAS contains two parts: a *predicate* which describes the action of a clause and *semantic arguments* which are mappings of the roles to words in the clause. Fundel (2019) developed the RelEx approach using dependency parse trees to extract relations from sentences. These are typically used to extract semantic roles. However, this approach can be difficult to understand for a non-linguist. In this work, we use a deep convolutional neural network-based approach for SRL named SENNA (Barnickel et al., 2009). In contrast to other lexical resources, SENNA doesn't

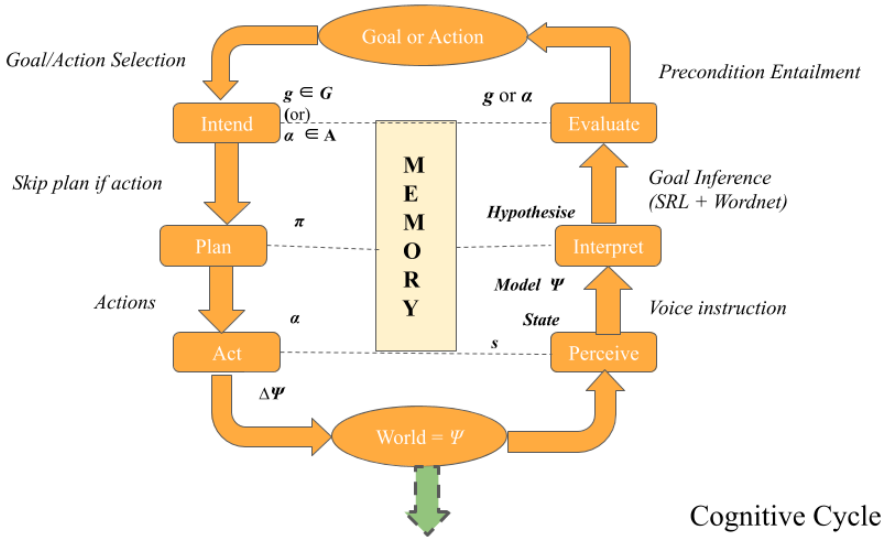

Figure 1. A conceptual model of the Goal Inference Model integrated with Cognitive Cycle of MIDCA. .

depend on syntax trees and is trained on PropBank (Palmer, Gildea& Kingsbury, 2005), a database of sentences from *The Wall Street Journal* that are manually annotated with their typical semantic roles.

4. The Goal Inference Procedure

We now describe our goal inference procedure which integrates natural language processing techniques including synonym generation and SRL into the comprehension phases of MIDCA's action-perception loop. To better explain the novelty of our model, we consider an environment where a human gives voice instructions and a self-regulated autonomous agent analyzes those instructions to infer a requested action or implied goal. Also For the purposes of this work, we assume that all voice instructions are rendered as text using an error-free process. To understand the intent of an instruction, the agent performs a simple test. In order for an instruction to be interpreted as an executable action, all preconditions on that action must be satisfied. If the test is passed, we infer that that action is the correct interpretation of the user's intent. Otherwise, the agent assumes that the user intent must have been an end-state they desire. The end-state, when achieved, satisfies the postconditions of the operator. In this case, the agent infers that the user intended to specify a goal and constructs a plan to achieve it.

4.1 Conceptual Model with MIDCA

We describe a conceptual model for goal inference in MIDCA in Figure 1. In *"perceive"* phase, the agent notes the initial state (s) of the world (Ψ) and listens for a voice instruction. This voice

instruction is rendered to text format. In the *"interpret"* phase, the SRL extracts the verb of the sentence and its semantic arguments. Using Wordnet and operators in the domain, the model identifies the operator associated with the extracted verb. The agent checks the preconditions on the operator and infers an action or a goal. An appropriate action or goal is added into the set of Goals (G) or Actions (A) during the *"evaluate"* phase. The *"intend"* phase selects a goal or action. The agent will skip the *"plan"* phase if the selected item is an action. Otherwise, the agent plans on achieving the goals by creating a series of sub-actions. In the *"act"* phase the actions are performed by the agent.

4.2 Generate and Filter Synonyms

We use the Natural Language ToolKit interface (Loper & Bird, 2002) to Wordnet to generate Synsets for a set of domain-specific keywords. For our current examples, we manually selected synonyms of operator and relation names given by WordNet. In the construction domain, some action verbs for operator *"stack"* and predicate *"on"* are stack, on, put, and place.

As the domain of discourse is limited, few synonyms must be filtered out. Since the operators describe actions, only synonyms in their noun and verb forms were used. Some manual work is required to filter out words specific to other domains. For example, some of the Wordnet synonyms of *"put"* are put, put_option, place, and invest. However, in the blocks world domain, invest and put_option are not viable candidates.

This refined domain-specific keyword list, which includes the synonyms, is included in the initial knowledge of the agent. They are stored as a list of dictionaries with operators as keys and the refined synonym lists as values. Given instruction, the agent searches the instruction for the keywords. If the search fails, the instruction is not within the agent's capabilities. If the search yields a match, the agent infers the instruction as an action or a goal after an associated operator is picked.

4.3 Semantic Arguments to Domain Objects Mapping

We used Practical Natural Language Processing Tools (practNLPTools) (Barnickel, 2009) library over SENNA. SENNA helps to extract SRL. As SENNA is trained on PropBank, it may not be able to extract semantic roles from domain-specific instruction (if any). Therefore, sentences that couldn't be annotated were considered as a failed response by an agent in inferring the user's intent. If a sentence contains more than one verb, the SRL annotator will generate two PAS's, one for each verb. However, all PAS's whose predicates do not fit in the domain are filtered out. Once the PAS is generated, the semantic arguments are mapped to objects in a simulation by looking at descriptors in the arguments. Some examples of descriptors are the location of the objects like *on-table, in hand,* and the color of the objects like *red block, yellow triangle*. The agent looks for descriptors and maps them based on state representation. If the descriptors are missing, there is ambiguity about what object the user is referring to. Our model doesn't solve this ambiguity yet. In the Example, "put" is one of the domain-specific keywords and the associated operator is stack. The SRL annotator generates PAS, with a verb (V) of *"put"* and two semantic arguments A1 and A2 refer to the roles of the phrases with respect to the verb.

Example: *put the red block on the green block*
PAS: V: "put" A1: *"the **red** block"* A2: *"on the **green** block"*
The descriptors in A1 and A2 are ***"red"*** and ***"green"*** colors. If the state representation contains any objects with these attributes, they are mapped to domain objects. Currently, the procedure cannot disambiguate between two same color blocks.
Domain-Specific Mapping: operator *"stack"* objects: $[(block, \textbf{\textit{red}}), (block, \textbf{\textit{green}})]$

4.4 Semantic Arguments to Operator Arguments Mapping

We used a dictionary to map the colors to the operator arguments. Their domain-specific mapping is converted to operator argument mapping to create an instantiated action which will be used to infer the user intent. In the example, the *red block* is mapped to the operator argument A and the *green block* is mapped to the operator argument B. We assume that the domain doesn't have blocks of the same color and the order of the arguments given is the order of the operator arguments. So, the top block is A and the bottom block is B
instantiated-action: $stack(A, B)$

4.5 Inference through Precondition Checks

The user instruction either describes an action to be performed or describes an end state of the world which is interpreted as a goal. Making the distinction on whether the instruction is an action or a goal is difficult. So, we rely upon checking the conditions of the action associated with the operator in domain-specific mapping

4.5.1 Action Creation

Our approach assumes that if a user intends to specify an action to perform, the current state of the world should already satisfy the preconditions of that action. An executable action doesn't require planning. Based on this fact, we run a simple test to see if the preconditions of the extracted operator are met. If they are met, the agents infer that the user intends to request an action. Therefore, MIDCA does not insert any new goals during the *"intend"* phase of the cognitive cycle and in the *"plan"* phase, inserts the requested action as a trivial plan. In the Example from section 4.3, the domain-specific mapping extracted from the instruction shows the operator *"stack"*, its domain object arguments, and operator arguments.. Once this is obtained, we check for all the preconditions associated with a instantiated action over the operator *"stack"* is $stack(A, B)$, that is we check whether the statement $holding(red\ block/A) \wedge clear(green\ block/B)$ is true. So, if it is true, the agent infers the instruction as an executable action.

4.5.2 Goal Creation

Consequently, if the preconditions are not met, the instruction cannot be an action. The agent assumes that the human must have been describing a desired end state of the world instead of an action. To achieve this end state, the agent first infers its description based on the postconditions of the operator. These postconditions are treated as a goal, which the agent creates and adopts. Then, during the plan phase the agent generates a feasible series of actions to fulfill the goal.

For the example from section 4.3, the preconditions on the instantiated action $stack(A, B)$ are $holding(red\ block/A) \wedge clear(green\ block/B)$. Let's say the agent is not holding the red block. So, the agent knows that it is not an immediate/executable action. Then the agent assumes that *"put the red block on the green block"* describes an end state which is obtained by looking at postconditions. In this case, the desired end state is that the red block must be on the green block. So, the agent infers that the human wants to create a goal of stacking two blocks which is the operator identified from the instruction. The MIDCA planner will generate a viable plan for that goal.

Note that the goal inferred from the postconditions may not always match the user's intent. For example, the goal behind *"pickup the yellow triangle"*can mean any of the following: the desired end state is that you want the yellow triangle in hand (this is to be the postcondition of pickup operator), or you didn't want the yellow triangle to be on the table anymore, or it is an action in achieving the goal of clearing some other block (assuming yellow triangle is on top of it). Our procedure doesn't compare alternative goals at this point, but we think we are headed in the right direction for inferring a goal or an action from a given instruction.

4.6 Pseudocode

The goal-inference process shown in Algorithm 1 gets the initial state (s) as the input. The robot has the prior information of synsets of domain keywords (K) through Wordnet. The algorithm checks if the operator (o) is in the keywords (K). If it is not, then it is not a valid instruction. If not, the algorithm runs a simple test on the preconditions of the instantiated action associated with the operator. The arguments are mapped to domain objects and operators arguments. If the current state of the world entails the preconditions then an executable action is inferred. Otherwise, the algorithm assumes that the user is describing a desired state and infers a goal from the postconditions of the instantiated action. The pseudocode for the algorithm is described below.

Algorithm 1: Goal Inference through condition check

input: State of the world, ***s***
Data: Synsets of domain-specific keywords ***K***
Result: *ACTION* or *GOAL* g or *INVALID* instruction

Get Instruction ***I***← Speech in text form
Initialize semantic role labeler ***SRL***
Get predicate argument structures ***PAS*** ← ***SRL(I)***
Initialize operator ***o*** ←***pAS.get_verb()***
Get arguments ***args*** ← get_arguments(***PAS***)
Get Color-Object Argument Map ***map*** // from memory // for args → operator args/variables
Initialize Mapping ***map*** θ **= *map_syntactic_arguments(o, args)*** // operator variables → domain objs
if o $\in$ **K then**
    `// Test for goal inference (SRL + Wordnet)`
    if s $\models$ *preconditions(***instantiated-action(o, map, args)***) then**
        `/* since preconditions are met, add the action to be performed based on operator. */`
        return $\alpha \leftarrow$ *subst* (***head(o)***, θ)
    end
    else
        `// assumes that the instruction is describing the end state`
        `// create goal based on postconditions (instantiated-action`
        return $g \leftarrow$ *subst* (*postconditions(*instantiated-action(**o, map, args**), θ)
    end
    `// In the "plan" phase, MIDCA Agent will plan for this goal.`
end
else
    return *INVALID*
end

5. Usage Examples

The instructions used while developing our procedure are tabulated in Table 1. These examples are for the purpose of demonstrating that our procedure allows for flexibility in the sentences, thus increasing human satisfiability. To illustrate the usage examples, we consider the construction domain which supports the following operators: *"stack"*; *"unstack"*; *"pickup"*; and *"putdown"*. The domain consists of several blocks. Each block is described by its color, and identified by a symbol, by convention a capital letter. The same initial environment state (s) is used in all examples below and is depicted in Figure 2 (a).

1. V: refers to the verb of the sentence
2. A: refers to the semantic arguments related to the verb

Table 1. Instructions used while developing the Goal-Inference Model.

SNo.	Sentence	SRL-PAS
1	"Let the blue block be placed on the green block"	V[1]: placed, A[2]: Blue and Green Blocks
2	"the red block is uncluttered from the yellow triangle"	V: uncluttered, A: Red Block and Yellow triangle.
3	"Grab the yellow triangle"	V: Grab, A: Yellow triangle
4	"Please stack the yellow triangle on the blue block"	V: please, A: stack the yellow triangle on the blue block V: stack, A: yellow triangle and blue block
5	"You need to grab the yellow triangle"	V: need, A: grab the yellow triangle
6	"Put down the red block"	N/A
7	"I want the red block in your hand"	V: want, A: red block and hand
8	"Hold the purple triangle"	N/A
9	"The yellow triangle is uncluttered from the red block"	V: uncluttered, A: yellow triangle

We elaborate the goal-inference procedure followed by the agent with a few of these sentences. When sentence 1: *"please put the blue block on the green block*" is received in the *"perceive"* phase, the SRL annotator identifies the main verb as *"put"* with *"blue block"* and *"green block"* as semantic arguments. The map is used to map the arguments to the operator arguments. The *blue block"* and *"green block"* are mapped as B and C and the assumed order as top and bottom block respectively. The domain specific operator for the verb is identified as *"stack"*. The preconditions for instantiated action $stack(B, C)$ are $clear(C) \wedge holding(B)$. Observing the initial state, we can conclude that C is clear; however, the MIDCA agent is not holding B. Since all the preconditions are not met, the agent concludes that the instruction is not an executable action. This process of reaching the instantiated action is carried out in the *"interpret"* phase. The precondition entailment leads to the inference which is added to the goals or actions in the *"evaluate"* phase. The agent then assumes that the user is describing a desired end state, so it infers an intended goal from the post conditions. In this case, the postconditions of action $stack(B, C)$ state that block B is on block C. So, the agent infers the goal as $on(B, C)$. Note *"on"* is the predicate associated with *"stack"* in the construction domain. This is added into the set of goals (G) in the *"evaluate"* phase. After this goal is selected in the *"intend"* phase, the agent creates a plan to achieve it in the *"plan"* phase. In this scenario, a satisfying plan is: $[pickup(C), stack(B, C)]$. These actions are performed in the two subsequent *"act"* phases. The final state achieved after the agent performs the inferred goal is shown in Figure 2(b).

Similarly, when sentence 3: *"Grab the yellow triangle"* is perceived, the operator extracted for the verb *"grab"* is *"pickup"*. The predicate for this verb is *"holding"* and the argument *"yellow triangle"* is mapped to D as operator argument and to the corresponding domain object. The current state of the world is described in Figure 2(b). The current state doesn't satisfy the preconditions of operator *"pickup"* because the D block is on the A block; hence, the action $pickup(D)$ is not executable. Therefore, the agent will infer a goal from the postconditions which is that the agent is holding the block D. So, the agent infers a goal of $holding(D)$. Observe that the predicate for the verb itself is holding which is a postcondition. After this goal is selected, the agent creates the plan

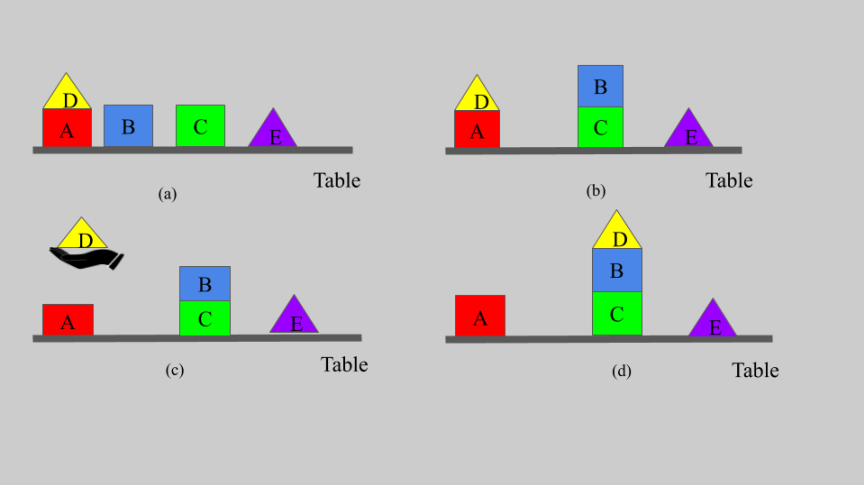

Figure 2. (a) Initial state of the objects (blocks, triangles) in the construction domain, (b) State after executing the stacking goal on B (blue) block and C (green) block, (c) State after executing a pickup goal on D (yellow) triangle, and (d) State after executing the action of the putting the D (yellow) triangle on B (blue) block.

$[unstack(D, A), pickup(D)]$ to achieve it. This leads to the final state of D being in the agent's hand. The final state is shown in Figure 2(c). Note that there are other alternative goals that might satisfy this instruction, such as *"unstack"* the yellow triangle from the *"red block"*. We are not exploring these possibilities in this paper.

For sentence 4: *"Please stack the yellow triangle on the blue block"*, the SRL annotator extracts two verbs namely *"please"* and *"stack"*. However, the main verb of the sentence is *"stack"* as *"please"* is not a domain specific operator. The agent will identify the operator as *"stack"* and the predicate as *"on"*. The arguments are mapped to C and B objects in the domain.The preconditions of the operator are $holding(D) \wedge clear(B)$. Since the current state satisfies the preconditions, the agent will infer that this action is executable and perform it, putting block D on block B. The MIDCA agent's final state shown in Figure 2(d).

6. Limitations

We use some of the examples from Table 1, to explore the limitations of our goal inference procedure. First, our procedure doesn't consider all possible alternative goals associated with an instruction. Second, some verbs identified by the SRL Annotator do not lead to correct identification of an operator, because they do not describe executable operations. For example in sentence 5, the user intent is to pick up the red block but the verb *"need"* does not correspond to any domain operator. The annotator does not recognize the word *"grab"* as a verb in contrast to the behavior displayed on sentence 3. Similarly in sentence 7: *"I want the red block in your hand"*, the intent was that the agent will be holding the block. However, the verb *"want"* doesn't correspond to any operator in the domain. Third, the SRL annotator extracts verbs but will fail if there are not enough arguments for the operator. To demonstrate this issue, consider sentence 6: *"Put down the red block"*. This

fails as it couldn't identify down as the second argument. So, our procedure will fail to infer a goal or action. However, *"putdown"* combined is a domain operator. The means the sentence itself can be converted to a valid goal/task in the domain. In contrast, the SRL sometimes failed to identify the verb. For example the word *"hold"* in sentence 8: *"Hold the purple triangle"*, is not interpreted as a verb by the SRL annotator. So, the SRL fails to extract a PAS. However, *"hold"* can be associated with domain predicate *"holding"*. This means the sentence represents a valid goal but our procedure failed to recognize it.

Another limitation of the procedure is due to the rigidity of the annotator. We have to be very specific with the sentence structure or else the annotator either fails to extract PAS or doesn't extract all the arguments. A sample case is if we change the order of the arguments in the sentence. Consider Sentences 2 and 9, they differ in the order of their arguments: yellow triangle and blue block. Although the SRL is able to detect both the arguments in sentence 2, it fails to detect the second argument in sentence 9. So, the order of the arguments matters while giving the instruction.

7. Discussion and Future Work

In this paper, we describe a simple procedure for inferring the intent behind a user-provided instruction. We demonstrated the effectiveness of this procedure for inferring user intent in some initial examples. We believe that our procedure is headed towards achieving flexibility and improving human robot collaborative experience. We plan to develop a concrete experimental design with human subjects in order to get evidence about whether this procedure increases user satisfaction.

We identified several limitations of the procedure caused by the design assumptions and the performance of SRL. The MIDCA agent only considers a single possible goal that the user might have meant. In future work, we plan on considering multiple goals consistent with the same instruction. We also plan to explore training the SRL annotator with domain specific instructions and developing new approaches to interpretation of instructions with no verb. Some of these limitations are caused by our commitment to accepting free-form instructions rather than engaging in rigid pre-scripted conversations. So, despite the limitations, we believe that with additional work we can increase user satisfaction through goal inference in Human Robot collaborations.

In future work, we plan to provide experimental evidence for increased user satisfaction due to the goal inference procedure, extend the procedure to additional domains to demonstrate generality, and add the capability to disambiguate similar objects through conversation. We will look into alternative techniques to decrease the manual work in the filtering of the synsets. We will further investigate what type of verbs affects the performance of the SRL and look into improving it. We also plan to investigate inference of the most likely alternative goal using Bayesian inference and/or an agent's history of interactions with the environment. We will also explore adding domain-specific knowledge graphs or ontologies to increase the accuracy of user intent inference.

Acknowledgements

Funding from a graduate fellowship at Wright State University for the first author is acknowledged.

References

Allen, J.F., Miller, B., Ringger, E., & Sikorski. T. (1996). Robust Understanding in a Dialogue System. *In Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics*. (pp. 62-70). Santa Cruz, CA: Morgan Kaufmann.

Allen, J. F., Dzikovska, M., Manshadi, M., & Swift, M. (2007). Deep linguistic processing for spoken dialogue systems. *In proceedings of EACL-07 Workshop on Deep Linguistic Processing*. Prague.

Anderson, M. L., & Perlis, D. (2005). Logic, self-awareness and self-improvement: The metacognitive loop and the problem of brittleness. *Journal of Logic and Computation 15*(1).

Aust, H. & Oerder, M. (1995). Dialogue control in auto-matic inquiry systems .*In proceedings of the ESCA Work-shop on Spoken Dialogue Systems*. (pp. 121-124). Vigsø.

Baker, C. L, Tenenbaum, J., & Saxe, R. R. (2007). Goal Inference as Inverse Planning. *Proceedings of the Annual Meeting of the Cognitive Science Society*, 29. Retrieved from https://escholarship.org/uc/item/5v06n97q

Barnickel, T., Weston, J., Collobert, R., Mewes, H. W., & Stümpflen, V. (2009). Large scale application of neural network based semantic role labeling for automated relation extraction from biomedical texts. *PloS one*, *4*(7), e6393. https://doi.org/10.1371/journal.pone.0006393

Akan, B. (2012). Human Robot Interaction Solutions for Intuitive Industrial Robot Programming (Licentiate dissertation). Mälardalen University, Västerås. Retrieved from http://urn.kb.se/resolve?urn=urn:nbn:se:mdh:diva-14315

Bordallo, A., Previtali, F., Nardelli, N., & Ramamoorthy, S. (2015). Counterfactual reasoning about intent for interactive navigation in dynamic environments. *2015 IEEE/RSJ International Conference on Intelligent Robots & Systems (IROS)*, 2943.

Bugmann Guido, & Norberto Pires J. (2005). Robot-by-Voice: Experiments on Commanding an Industrial Robot Using the Human Voice. *Industrial Robot: An International Journal*, 32(6), 505–511. https://doi-org.ezproxy.libraries.wright.edu/10.1108/01439910510629244

Cox, M., Alavi, Z., Dannenhauer, D., Eyorokon, V., Munoz-Avila, H., & Perlis, D. (2016). MIDCA: A Metacognitive, Integrated Dual-Cycle Architecture for Self-Regulated Autonomy. In *AAAI Conference on Artificial Intelligence*. https://www.aaai.org/ocs/index.php/AAAI/AAAI16/paper/view/12292/12151

Cox, M. T., Oates, T., Paisner, M., & Perlis, D. (2012). Noting anomalies in streams of symbolic predicates using A-distance. *Advances in Cognitive Systems, 2*, 167-184.

Ferguson G., & Allen, J.F. (1993). Generic Plan Recognition for Dialogue Systems. *proceeding of ARPA Workshop on Human Language Technology*.(pp. 21-23). Princeton, NJ.

Fundel, K., Küffner, R., & Zimmer, R. (2007). RelEx--relation extraction using dependency parse trees. *Bioinformatics (Oxford, England)*, *23*(3), 365–371. https://doi.org/10.1093/bioinformatics/btl616

Ghallab, M., Nau, D. S., & Traverso, P. (2016). *Automated planning and acting*. Cambridge University Press.

Holada, M., & M. Pelc. (2008). The Robot Voice-Control System with Interactive Learning. In Lazinica, A.(Ed.), *New Developments in Robotics Automation and Control*. Rijeka, *Czech Republic*: InTechOpen.

Kogan, Y., Collier, N., Pakhomov, S., & Krauthammer, M. (2005). Towards semantic role labeling & IE in the medical literature. *Proceedings of AMIA Annual Symposium.* (pp. 410–414).

Liu, C., Hamrick, J. B., Fisac, J. F., Dragan, A. D., Hedrick, J. K., Sastry, S. S., & Griffiths, T. L. (2018). *Goal Inference Improves Objective and Perceived Performance in Human-Robot Collaboration*.

Loper, E., & Bird, S. (2002). *NLTK: The Natural Language Toolkit*.

McTear, M. F. (2002). Spoken Dialogue Technology: Enabling the Conversational User Interface. *ACM Computing Surveys*, *34*(1), 90. https://doi-org.ezproxy.libraries.wright.edu/10.1145/505282.505285

McTear, M. (1998). Modelling spoken dialogues with state transition diagrams: experiences of the CSLU toolkit. *In Proceedings of the International Conference on Spoken Language Processing* (pp. 1223-1226). Sydney, Australia.

Miller, G. A. (1995). Word Net: A Lexical Database for English. *Communications of the ACM*, *38*(11), 39–41. https://doi-org.ezproxy.libraries.wright.edu/10.1145/219717.219748

Palmer, M., Gildea, D., & Kingsbury, P. (2005). The Proposition Bank: An Annotated Corpus of Semantic Roles. *Computational Linguistics*, *31*(1), 71–106. https://doi-org.ezproxy.libraries.wright.edu/10.1162/0891201053630264

Spiliotopoulos, D., Androutsopoulos, I., & Spyropoulos, D.C. (2001). Human-Robot Interaction Based on Spoken Natural Language Dialogue. *In Proceedings of the European Workshop on Service and Humanoid Robots.* (pp. 25-27).