

# A Multimodal Dialogue System for Learning Structural Concepts in Blocks World

Ian Perera<sup>1</sup>, James F. Allen<sup>1,2</sup>, Choh Man Teng<sup>1</sup>, Lucian Galescu<sup>1</sup>

<sup>1</sup>Institute for Human and Machine Cognition, Pensacola, FL 32502 USA

<sup>2</sup>University of Rochester, Department of Computer Science, Rochester, NY 14627 USA  
iperera@ihmc.us, jallen@ihmc.us, cmteng@ihmc.us, lgalescu@ihmc.us

## Abstract

We present a modular, end-to-end dialogue system for a situated agent to address a multimodal, natural language dialogue task in which the agent learns complex representations of block structure classes through assertions, demonstrations, and questioning. The concept to learn is provided to the user through a set of positive and negative visual examples, from which the user determines the underlying constraints to be provided to the system in natural language. The system in turn asks questions about demonstrated examples and simulates new examples to check its knowledge and verify the user’s description is complete. We find that this task is non-trivial for users and generates natural language that is varied yet understandable by our deep language understanding architecture.

## 1 Introduction

Current artificial intelligence systems, even dialogue agents, tend to play the role of a tool in real-world or even simulated tasks. Often the human user must be given an artificial handicap to create a situation where the system can play a role as a collaborator rather than a tool with interface commands simply replaced by natural language equivalents (Brooks et al., 2012). We work towards a natural language dialogue agent that we hope will eventually become a collaborator rather than a tool by focusing on knowledge transfer through natural language and determining areas where a dialogue agent’s proactive nature is a benefit to learning. To this end, we apply deep language understanding techniques in the situated Blocks World environment, where a user can teach the system physical,

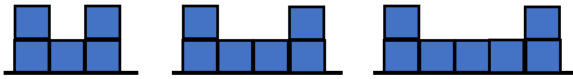
possibly compositional concepts to aid in development of natural language understanding without significant existing domain knowledge needed.

## 2 The Structure Learning Task

Many natural language dialogue tasks in a Blocks World environment focus on querying the environment (Winograd, 1971), block placement (Bisk et al., 2016; She et al., 2014), or training visual classifiers and grounding perception (Matuszek et al., 2012; Mast et al., 2016; Perera and Allen, 2015). Reference resolution has also been extensively studied in this environment and statistical methods show strong performance in quickly learning referring expressions (Kennington et al., 2015). However, our focus is exploring collaborative concept transfer with the goal of having situated agents learn from natural language dialogue and physical interaction to become better collaborators. With the goal of the system as a collaborator, we find it is important that the task carried out be non-trivial for the user. However, more difficult tasks can have drawbacks – they involve larger amounts of background knowledge and reasoning, progress can be difficult to evaluate, and often the language and concepts learned do not extend easily to other real world applications.

With these constraints in mind, we use a physical Bongard problem (Bongard et al., 1970; Weitnauer and Ritter, 2012) task for evaluating our system in a situated Blocks World environment. The user is provided with a set of visual examples, some positive and some negative, and must determine the constraints on the class of structure that allows the positive examples (and perhaps others) while avoiding any negative examples (Figure 1). By providing only visual clues, we leave the generation of the constraints entirely up to the user. The user then begins interacting with the system

### Positive Examples:



### Negative Examples:

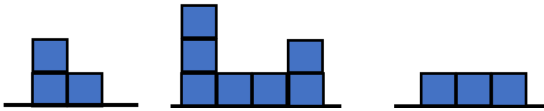


Figure 1: An example of the set of images provided to a participant for teaching the system a “U” shape. The user is tasked with explaining the underlying concept to the system such that the system correctly identifies the positive examples while rejecting the negative examples.

to describe the structure. During this time, the system is able to ask questions, check its model with demonstrations, and ask for the user to present examples. While this problem has been explored in the context of cognitive architectures (Foundalis, 2006) and reinforcement learning (Ramon et al., 2007), we are unaware of any prior work in the context of dialogue systems.

We believe this task addresses a number of issues with previous tasks in evaluating a system that can learn new concepts and structures. Participants typically spent two to three minutes going over the examples, showing that there was some thought required to correctly understand the structure on their part. Furthermore, often the constraints users provided were underspecified – they described what structures would be allowed, but sometimes failed to provide sufficient restrictions to avoid the negative examples. The system is then able to find gaps in the user’s description by presenting examples following the current description so far – bringing the system’s state in the dialogue from being solely a student to contributing to the task in a meaningful way.

In addition, this task can be scaled in difficulty or extended to other domains. Difficulty scaling can be achieved by using compositional constraints that build on existing knowledge (e.g., “Build a U shape, but make one column taller than the other”) or by creating more difficult structures to learn. The task could be adapted to other domains by augmenting the ontology and designing a new reasoning agent that could integrate asser-

tions into its model, while retaining similar interactions and the domain-generic modules.

## 2.1 Challenges

One of the primary challenges in this task is the wide variety of ways in which a user might describe or teach a class of structures. For example, they might describe necessary features or prohibited features. They may view possibilities as movement, saying “The columns have a row between them wherever they move.” They may present negative examples for the system to avoid. Some users describe a particular arrangement of blocks that should or should not appear (e.g., “There is never two blocks on top of each other”), while others describe a more holistic conception (e.g. “The maximum height is one block”). While we do not succeed in interpreting all such description modalities, we believe our current methods handle a large range of possible explanations and are amenable to advancements to understand a greater number of explanation types in the future.

## 3 Environment and Apparatus

Our system operates in a Blocks World environment consisting of 6-inch cubes placed on a table. Although the cubes have distinct images for identification and colored sides, we do not use this information in our current version – blocks can only be referred to using descriptions of their location in the environment. We use two Kinect 2.0’s to detect the blocks, with the separate perspective aiding in avoiding issues with occlusion. On the opposite side of the user is a monitor with a 3D avatar that speaks the system’s generated text and also has non-verbal communication capabilities such as nodding, pointing, and other more complex gestures. The environment is calibrated such that these gestures can point to the location of a block for communicating about it. The apparatus with the avatar is shown in Figure 2.

The apparatus has no physical means for the system to move blocks. However, during interaction the system we find it important for the system to build structures to test its knowledge. To do this, we generate a 3D image in a virtual representation of the current environment showing the blocks that the system wants to place as an example for the user. This can be sent to a separate tablet such that an assistant can place the blocks, or displayed on the screen for the user to evaluate themselves.



Figure 2: The apparatus and the environment containing the blocks used, with the screen displaying the avatar and 3D visualization of the environment.

User input is currently carried out by keyboard entry by the user or dictation by an assistant. We are currently implementing speech recognition to enable more natural communication. Towards this end we focus on finite state machine language models given the nature of assertions our system understands, but we may have to consider more flexible corpus-based models in the future, aided by transcripts of previous trials.

## 4 System Architecture

The heart of the dialogue management is the TRIPS architecture (Allen et al., 2001), which connects a number of components through KQML message passing (Finin et al., 1994), with each component augmented with domain-specific knowledge to varying extents. This dialogue management component, including parser, a generic ontology, and an API for interacting with a domain-specific module is open-source and available for download <sup>1</sup>. As opposed to other dialogue management systems like OpenDial (Lison and Kennington, 2016) or POMDP dialogue systems (Williams and Young, 2007), this dialogue management system is primarily suited for collaborative tasks where there is little to no knowledge of what dialogue state typically follows from the previous one – the user can move from statements about goals to assertions to questions in any order, determined primarily by the speech act detected in their utterance. For semantic language understanding and speech act interpretation of the user’s utterances, the domain-generic

<sup>1</sup><https://github.com/wdebeaum/cogent>

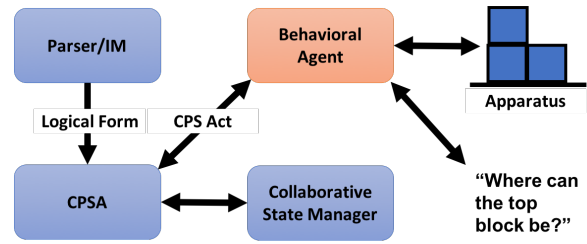


Figure 3: The TRIPS collaborative problem solving architecture adapted to this task. Only the Behavioral Agent and apparatus are specifically designed for this task – the other components are adapted only through additions to the ontology.

TRIPS parser (Allen et al., 2008) generates logical forms and speech act possibilities backed by a domain-augmented ontology. The relevant speech act is then determined by the Interpretation Manager (IM), which also fills in remaining context-dependent references before sending this information to the Collaborative Problem Solving Agent (CPSA). The CPSA facilitates communication between the parser/IM, Collaborative State Manager (CSM) and the Behavioral Agent (BA) as Collaborative Problem Solving (CPS) Acts. These acts include adopting, selecting, proposing, and rejecting goals, queries to the user or the system, and reporting the current status of a given module. The overall architecture is shown in Figure 3.

### 4.1 Collaborative State Manager

The Collaborative State Manager stores and responds to queries regarding the systems goal state and facilitates decisions based on goal context. As opposed to the CPSA, the CSM does not have a notion of dialogue context, but does respond to speech acts that require the system to generate a response based on the systems state and knowledge. It also is responsible for generating the necessary clarification messages to continue with dialogue, and for managing initiative in mixed-initiative tasks based on a changing domain-specific environment.

To make these decisions, the CSM is designed with a combination of domain-independent behavior and domain-specific knowledge supplied at a broad level. Such knowledge takes the form of a specification of which types of goals might be considered goals in their own right (e.g., teaching the system a concept, building a structure), and which are considered subgoals (e.g., showing an example, adding a constraint to the system’s struc-

ture model). The goal hierarchy consists of one or more top-level goals, with sub-goals, queries, and assertions added as child nodes to create a tree structure. With this structure, the user and the system can resolve sub-goals and blocking actions such as goal failures and rejected goals without losing the overall goal context. The system works with the user to ensure that there is a top-level goal when beginning the dialogue to ensure the proper context is available for the system, offering possible top-level goals based on the action or assertion the user provides.

The CSM uses a light, domain-specific knowledgebase of top-level goals, subgoals, and related speech acts to infer the users intentions and goals based on the incoming speech acts. For example, the statement “The top block must be on the left-most column” would yield a proposed subgoal in a structure building task, but should be resolved as an assertion to be added to the BA’s model during the structure learning task. If there is no top-level goal, the system would ask the user about the top-level goal (e.g. “Are you trying to teach me something?” in response to an assertion) to establish one. When the CSM is unable to resolve ambiguity given the information it has, it will generate a response that indicates the system needs more information from the user, and will provide possible solutions such that other modules can generate responses to try to provide efficient communication.

## 4.2 Behavioral Agent

The BA is the domain-specific aspect of the system dealing with interaction with and reasoning over the environment. In this system, we also relegate language generation to this component. To design a BA in this architecture, one creates a module that accepts a set of incoming messages, dealing with goal proposals, requests for execution status (i.e. finished a task, waiting for the user, or currently acting), queries about the environment or model, in addition to a “what-next” message that serves to provide dialogue initiative to the system. As the BA receives goals, it determines whether they are achievable and accepts them, and then proposes the next goal – for example, a teaching goal will be responded with a subgoal to describe an aspect of the structure. As assertions are processed, they will be added to the model, rejected if not understood in context, or clarified with a query in the case of ambiguous statements.

## 4.3 Goal Management

The base TRIPS architecture provides some means for the user to respond to errors through dialogue, and provides flexibility in goal management. For example, if the user wants to continue the dialogue in a different way, they can reject it by responding “No” to the BA’s goal proposal, or they can continue to provide assertions or ask questions even when the system has proposed the goal. This flexibility is essential to reduce user frustration when coming up against obstacles and ensure that the user feels a sense of control even when the BA is proactive in dialogue.

## 5 Constraint Processing

Constraints are processed as assertions that are interpreted as holding generally during the structure learning process, rather than relying on the identity of any one particular block or group of blocks. Therefore, the utterance, “The top block has to be on the left” may or may not currently be true about a particular example, but nevertheless should hold in all positive instances of a structure class.

Constraints can be general properties about the structures, such as the maximum/minimum height or width, or they can refer to particular blocks or groups of blocks. All non-general constraints must contain a referring expression, which consists of a referred object or arrangement (i.e., blocks, rows, columns, or spaces) and optionally a location description to pick out a particular object. The assertion can assert that such a referent exists in the structure, constrain a particular feature of the referent, (e.g., width, height, the number of blocks it contains), or dictate its location relative to the rest of the blocks or a particular set of blocks denoted by another referring expression. We also have limited support for compositional referents in referring expressions, picking out certain aspects of a structure (e.g., “the ends of the row”).

A constraint can be designated as exclusive, which means that only one instance of a particular object can have that property (e.g. “Only the left-most column has more than 2 blocks”). Currently we take an object-type scoping for this restriction. In addition, we handle negations at certain scopes, such as disallowing a particular arrangement (e.g. “There are no columns of height greater than 2”) or location (e.g. “There is no block next to a column”). An example of some utterances understood by the system is shown in Figure 4.

“The leftmost column’s height is 3 blocks.”  
“The height is 3.”  
“The height of the leftmost column is less than the height of the rightmost column.”  
“There is a column with at least 3 blocks.”  
“There is a space between the top 2 blocks.”  
“The bottom row is connected.”  
“The top block is always on the left.”  
“The top block can be anywhere.”

Figure 4: Examples of understood constraints.

### 5.1 Constraint Extraction

To extract constraints, we primarily depend upon the logical form structure of the TRIPS parser, which allows direct extraction of the types of constraints we are interested in due to its argument structure of concepts. We first determine all referring expressions by finding mentions of blocks or arrangements. Then we add any modifiers to their location. Once the referring expressions are found, we construct a constraint, which consists of the subject (the :figure argument in the TRIPS logical form), the reference object or property (the :ground argument), and the feature of comparison (e.g., height, width, count) or a predicate constraining the location of the subject relative to some reference set. Figure 5 shows an example of structures extracted from a logical form.

### 5.2 Constraint Evaluation

When the system is asked to evaluate an example or create its own, it evaluates all current constraints by finding referents for each referring expression according to the object type and predicate. Predicates are calculated using predefined rules, either specifying constraints that apply to individual blocks or using axis-aligned bounding boxes. These rules have built-in tolerances of a half-block width to account for noise or imprecise placement. We then calculate the features and predicates in the constraint for the resolved reference and return a value for each constraint.

Initial versions of the system primarily built up constraints from a sequence of user utterances. When performing Wizard of Oz studies, we found that an issue with this method is that it can sometimes be difficult for users to formulate and describe a concise, consistent model without any direction. This can lead to run-on sentences which are difficult to parse, or, if parsed, difficult to interpret as constraints. We believe one reason for this

is that, while the system provides affirmation at the end of an utterance entered by keyboard, it does not give non-verbal or verbal cues of understanding during speech recognition. Therefore, the user sometimes continues explaining in various ways looking for a signal that the system understands.

To address this issue, we designed our system to take a more proactive role in conversation. While the system still takes a free-form description or constraint at the beginning of the conversation, it then begins to ask questions about the structure class, ask for examples, present examples, and respond to the user’s questions about examples. The system can choose a feature of the structure that has not been described (to ensure the user feels that the system has understood the structure so far) and generate a query to send to the user. An unfilled version of the constraint is sent to the CPSA to aid in resolving the query, and the TRIPS parser is able to handle user responses fragments to fill in the constraint. The strategies to generate these utterances are described in Section 5.3.

The system generates its own examples of structures given its current knowledge as well. The BA will generate random arrangements of blocks in a grid structure until the current constraints have been satisfied, and then return a new structure as an example. This allows the user to see the result of the constraints and refine their constraints, while also providing more evidence that the system is understanding what is being said.

At any time, the user can ask whether a particular example is correct given the constraints provided so far. The system enumerates the constraints and can then state whether each constraint is satisfied and why or why not. If there are many constraints, the system will summarize the positive constraints (e.g., “The width is right” rather than “The width is greater than 2”) while focusing on the negative constraints to avoid long-winded explanations and provide a more natural response.

### 5.3 Learning Strategies

We designed the system to follow, with some deviations, a general strategy for learning based on our initial Wizard of Oz tests (Dahlbäck et al., 1993), such that the system can build up knowledge in a way that supports the interactions between different types of knowledge learned throughout an interaction. The system first asks the user for an aspect of the structure, discouraging long-winded

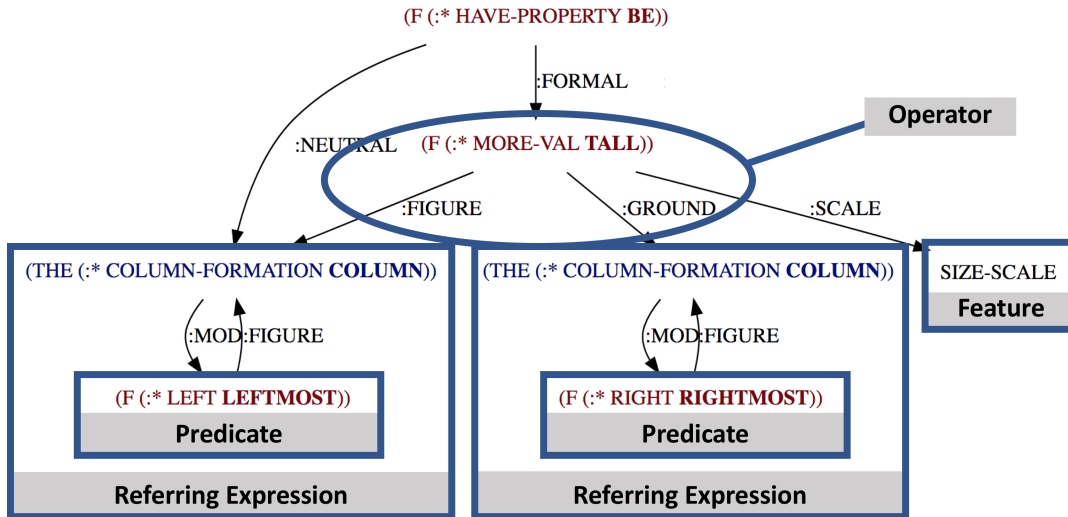


Figure 5: An example of constraint extraction from the TRIPS logical form for the utterance “The left-most column is taller than the rightmost column”

descriptions that can overwhelm the parser and tend to be less structured. With a constraint learned, the system then asks for an example. While the system cannot currently learn from the block positions themselves, it uses the example to generate a pertinent referring expression (e.g., “the top block”) that can then be used to ask about a constraint. This query guides the user to thinking about the structure in a concrete and specific way to aid in generating descriptions.

Because the system likely has two or more constraints at this point, it will then generate an example of its own based on its learned constraints and check for correctness. This allows the user to focus on aspects that are explicitly disallowed, since the system’s random generation process often reveals when constraints are underspecified. These learning strategies then continue until the user is satisfied that the system understands the structure. A sample dialogue showing the system’s abilities is given in Figure 6.

## 6 Wizard of Oz Experiments

Before developing our system, we carried out several Wizard of Oz experiments to aid in development of the TRIPS parser, ensure that our system had capabilities suited to the dialogue of a naïve user, and determine which interaction strategies of the system would be most effective for learning and most natural for the user. So far we have conducted 3 tasks each with 3 participants who were unaware of the details of the project, yielding 90

user utterances. Two of the participants were familiar with natural language processing research, while one was not.

**User:** I want to teach you a mountain shape.  
**System:** Okay. Can you describe an aspect of the structure?  
**U:** Yes. The rightmost column has 3 blocks.  
**S:** Ok. What is the greatest the height can be?  
**U:** 3.  
**S:** Ok, got it. Can you show me an example?  
**U:** Yes. This is an example. *⟨shows example⟩*  
**S:** Okay.  
**U:** Is this right? *⟨shows example⟩*  
**S:** I think so, because the height is less than or equal to 3 and the right column’s number is equal to 3.  
**U:** The left column has 2 blocks.  
**S:** Ok.  
**U:** Is this right? *⟨shows example⟩*  
**S:** I don’t think so, because the height is less than or equal to 3 but the left column’s number is not equal to 2 even though the right column’s number is equal to 3.

Figure 6: An example dialogue showing interaction with the system.

To ensure the closest approximation to the actual system, we used a tablet for the researcher standing in for the system. Because of the design of the apparatus where the system generates 3D images (and presumably diagnostic data), the participants believed it was actually the system per-



forming the dialogue and interpreting the user’s utterances, and the researcher was simply an assistant. In fact, the tablet served as a method for the researcher to respond with predetermined phrases, including acceptances, requests for examples, and questions about the structure. These phrases were then sent through the speech generation component of the system.

We found several advantages of users believing that the system was actually engaging in dialogue. First, the users used simpler language than they might have with a person, while still providing sufficient variation for exploring possible utterances. Second, users sometimes provided their own thoughts aside to the researcher, allowing a specific glimpse into users’ responses to certain utterances or tasks. For example, once when the system asked, “How tall can the structure be?”, the participant said as an aside, “It can be any height,” and then responded to the system “At least two blocks.” Finally, we could evaluate how the dialogue might progress without being interrupted by failures of the system at the parsing or interpretation level. We processed the Wizard of Oz dialogues with the TRIPS parser and correctly parsed 89% of the utterances. 90% of these correct parses also yielded the correct constraint when interpreted by the current state of the system.

## 6.1 Description Modalities

We recognized several different description modalities participants used when describing the structure without responding to a particular feature query. When the system asked questions, typically the user responded directly to the question, reducing the utterance complexity. However, these variations on the expected descriptions reveal interesting insight into how users generate representations of the concepts they are provided.

**Basic Constraints** – “The height of the leftmost column is greater than 2” – These descriptions are the simplest to interpret and make up the majority of user utterances, especially when the system is proactive in dialogue.

**Arrangement Constraint** – “The column can be either second in the row or third in the row.” Here the definite article conveys that there should be a single instance of a column, and the ordinal reference to the row constrains its position – even though in certain cases a row might be considered to be a line of blocks only a single block high.

We handle such cases by inferring a sequence of columns left to right, and then processing ordinal references to enforce constraints.

**Movement Modality** – “This top block can move wherever.” – These descriptions, using movement as a surrogate for possibility, are slightly more difficult to interpret, but can often be handled by our loose interpretation of logical forms that focuses on referring expressions (“the top block”) and predicates (“wherever”) without focusing on the event term of “moving”.

**Transformation Modality** – “The left block is a column.” – The difficulty in parsing a constraint described in this way is that a particular type of object (“the left block”) is picked from the structure with the goal of changing it to another arrangement (“a column”, presumably with more than one block). This can also be viewed as an explicit bridge between the current and desired state to fulfill the constraints. Currently we encourage users to describe the structure in a general way – however, this type of utterance demonstrates an example where a specific error or instance applies generally enough to serve as a constraint.

**Procedural/Construction Modality** – “No, you’re missing a row. Subtract by one every time you go up.” – Here the user is describing the structure by providing the process for creating it. While our current system does not understand these types of utterances, we are working on a similar task in parallel with this environment that can be leveraged to understand this modality (Perera et al., 2018). The key aspect of understanding this type of utterance is the compositionality of primitive concepts, namely a row, its length, and the sequence of rows that will presumably be placed by the agent building the structure.

A common property of these more complex modalities is the awareness of and reliance on the composition of primitive concepts, such as elements, order, and placement, when referring to more complex concepts such as rows and columns, which can have multiple attributes and purposes when describing a structure. We believe that a system to interpret these types of expressions and extend to other domains must therefore have a compositional model that allows complex concepts to be built from primitive concepts that are simpler to detect or identify. This compositional aspect of concepts used in communication has both informed the design of the system and inspired this

task which makes use of such modalities.

## 6.2 Preliminary Evaluations

Alongside adapting the system to handle our existing Wizard of Oz dialogues, we have begun preliminary user evaluations, primarily for testing for robustness and unforeseen methods of communicating constraints to the system. An example dialogue with a user is shown in Figure 7.

**User:** I want to teach you a shape.  
**System:** Okay. Can you tell me something about the structure?  
**U:** The bottom row must be at least 2.  
**S:** Hmm, I didn't quite get that.  
**U:** The bottom row has at least 2 blocks.  
**S:** Ok. What is the greatest the width can be?  
**U:** 5.  
**S:** Ok, got it. Can you show me an example?  
**U:** Yes. This is an example. *(shows example)*  
**S:** Okay. Where can the top block be?  
**U:** The top block is always on the left.  
**S:** Okay.  
**U:** *(user builds example)* Is this right?  
**S:** I think so, because the bottom row's number is greater than or equal to 2 and the width is less than or equal to 5 and the top block is on the left.

Figure 7: A taken from initial runs of the evaluation with the autonomous system and a user unfamiliar with the system.

Failures of the system are handled by the dialogue manager, with unparseable user utterances causing the system to continue with a question where the response is easy for the system to handle, such as asking for the maximum height or width, or by asking the user to rephrase. In initial trials, users were often able to rephrase constraints in a way the system could understand. Furthermore, users reported that the difficulty of the task made dialogue setbacks seem like a complementary challenge of clearly expressing an idea rather than an obstacle to an otherwise simple task.

To track development of the system, we will evaluate according to several metrics along with user surveys. The first measure will be the number of positive examples successfully recognized by the trained system and the number of negative examples successfully rejected. Next, we plan to track robustness by determining the number of cancellations, undos, or restarts by user, as well

as the efficacy of extracting constraints from user assertions. In addition, a final task which ensures that communication is two-way will be to reverse roles and have the system explain the concept to the user based on what it has learned from prior interactions with a different user.

## 7 Conclusion

We believe our system shows promise in the task of teaching a system new concepts in Blocks World in a manner extendable to multiple types of descriptions and with applications to multiple domains. While our first priority is to handle the most common description modalities of users to ensure broader coverage, we also begin the process of using this system as a stepping stone for language understanding and dialogue in other domains by mapping our concepts and predicates into a database to be used by our collaborators in this and related projects. With multiple definitions of features and predicates, we plan to use these concrete physical representations as proxies for more abstract and metaphorical reasoning capabilities to be developed in other systems.

Because the rules and interpretation are hand-crafted, brittleness can be an issue but is partially mitigated through dialogue repair. Given the primarily symbolic nature of the system and the difficulty of specifying composition with statistical models or neural networks, we focus our efforts on building rules to understand conceptual composition rather than processing utterances using statistical techniques. However, development of a broader range of understood constraint modalities can extend this dialogue system to other domains that involve a direct or indirect spatial or temporal component – such as scheduling, building graphical models, or directing scenes of a movie. Finally, we believe the compositionality inherent in the type of communication captured requires background knowledge about the conceptual structures we inherently use in discussing complex ideas.

## 8 Acknowledgements

This work is supported by the DARPA CwC program and the DARPA Big Mechanism program under ARO contract W911NF-14-1-0391. Special thanks to SRI for their work in developing the physical apparatus, including block detection and avatar software.



## References

- J. Allen, Mary Swift, and Will de Beaumont. 2008. Deep Semantic Analysis of Text. In *Symposium on Semantics in Systems for Text Processing (STEP)*, pages 343–354, Morristown, NJ, USA. Association for Computational Linguistics.
- James Allen, George Ferguson, and Amanda Stent. 2001. An architecture for more realistic conversational systems. In *Proceedings of the 6th international conference on Intelligent user interfaces - IUI '01*, pages 1–8.
- Yonatan Bisk, Deniz Yuret, and Daniel Marcu. 2016. Natural Language Communication with Robots. *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 751–761.
- M. M. (Mikhail Moiseevich) Bongard, Joseph K. Hawkins, and Theodore Cheron. 1970. *Pattern recognition*. Spartan Books.
- Daniel J Brooks, Constantine Lignos, Cameron Finucane, Mikhail S Medvedev, Ian Perera, Vasumathi Raman, Hadas Kress-gazit, Mitch Marcus, and Holly a Yanco. 2012. Make It So : Continuous , Flexible Natural Language Interaction with an Autonomous Robot. In *AAAI Workshop on Grounding Language for Physical Systems*.
- N. Dahlbäck, A. Jönsson, and L. Ahrenberg. 1993. Wizard of Oz studies - why and how. *Knowledge-Based Systems*, 6(4):258–266.
- Tim Finin, Richard Fritson, Don McKay, and Robin Mcentire. 1994. KQML as an Agent Communication Language. In *Proceedings of the Third International Conference on Information and Knowledge Management*. ACM Press.
- Harry Foundalis. 2006. *Phaeaco: A Cognitive Architecture Inspired By Bongard's Problems*. Ph.D. thesis.
- Casey Kennington, Livia Dia, and David Schlangen. 2015. A Discriminative Model for Perceptually-Grounded Incremental Reference Resolution. In *Proceedings of the 11th International Conference on Computational Semantics*, 2002, pages 195–205, Beijing, China.
- Pierre Lison and Casey Kennington. 2016. OpenDial: A Toolkit for Developing Spoken Dialogue Systems with Probabilistic Rules. *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 67–72.
- V Mast, Z Falomir, and D Wolter. 2016. Probabilistic Reference and Grounding with PRAGR for Dialogues with Robots. *Journal of Experimental & Theoretical Artificial Intelligence*, 28(in press).
- Cynthia Matuszek, N FitzGerald, Luke Zettlemoyer, Liefeng Bo, and Dieter Fox. 2012. A Joint Model of Language and Perception for Grounded Attribute Learning. In *Proceedings of the International Conference on Machine Learning (ICML)*.
- Ian Perera and James Allen. 2015. Quantity, Contrast, and Convention in Cross-Situated Language Comprehension. *Proceedings of the Nineteenth Conference on Computational Natural Language Learning*, pages 226–236.
- Ian Perera, James F. Allen, Choh Man Teng, and Lucian Galescu. 2018. Building and Learning Structures in a Situated Blocks World Through Deep Language Understanding. In *SPLU 2018: Proceedings of the HLT-NAACL Workshop on Spatial Language Understanding*, New Orleans, Louisiana. Association for Computational Linguistics.
- Jan Ramon, Kurt Driessens, and Tom Croonenborghs. 2007. Transfer Learning in Reinforcement Learning Problems Through Partial Policy Recycling. *Lecture Notes in Computer Science*, 4701:699–707.
- Lanbo She, Shaohua Yang, Yu Cheng, Yunyi Jia, Joyce Y Chai, and Ning Xi. 2014. Back to the Blocks World: Learning New Actions through Situated Human-Robot Dialogue. In *Proceedings of the SIGDIAL 2014 Conference*, June, pages 89–97.
- Erik Weitnauer and Helge Ritter. 2012. Physical Bongard Problems. In *IFIP Advances in Information and Communication Technology*, volume 381 AICT, pages 157–163.
- Jason D. Williams and Steve Young. 2007. Partially observable Markov decision processes for spoken dialog systems. *Computer Speech and Language*, 21(2):393–422.
- Terry Winograd. 1971. Procedures as a Representation for Data in a Computer for Understanding Natural Language. Technical report, Massachusetts Institute of Technology Artificial Intelligence.

## A Supplemental Material

Concept	Lemmas
ABOVE	above
HIGHER	higher
BELOW	below, beneath, under, underneath
LOWER	lower
ADJACENT	adjacent (to), next to, beside, by, contiguous (with), flush
CONNECTED TOGETHER	abut, adjoin, connect, touch together
ON	on, on top of
LEVEL	level with
TOP-LOC...	top
MIDDLE-LOC	middle
BOTTOM-LOC	bottom
BETWEEN	(in) between
CENTER	center
LEFT-LOC	left, lefthand, leftmost
RIGHT-LOC	right, righthand, rightmost
ANYWHERE	anywhere

Table 1: The list of predicates understood by the system, with their concept in the ontology, and matching lemmas that can resolve to that concept.

Ontological Concept	Data Type
ONT::WIDTH-SCALE	real+, count
ONT::HEIGHT-SCALE	real+, count
ONT::LENGTH-SCALE	real+, count
ONT::CENTER	point
ONT::LOCATION	point
ONT::STARTPOINT	point
ONT::ENDPOINT	point
ONT::TOP-LOC...	point
ONT::BOTTOM-LOC...	point
ONT::NUMBER	count
ONT::COL-FORMATION	column
ONT::ROW-FORMATION	row
ONT::DIRECTION	vector
ONT::HORIZONTAL	(real+)
ONT::VERTICAL	(real+)
ONT::LINE	(real+)

Table 2: Some of the features generated by the system for blocks, sets of blocks, and sequences, listed by their concept in the TRIPS ontology and the resulting data type. A data type in parentheses indicates the value is not presented to the user but is used in comparisons against other sets of blocks.