Building and Learning Structures in a Situated Blocks World Through Deep Language Understanding

Ian Perera[1], **James F. Allen**[1,2], **Choh Man Teng**[1], **Lucian Galescu**[1]
[1]Institute for Human and Machine Cognition, Pensacola, FL 32502 USA
[2]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 demonstrate a system for understanding natural language utterances for structure description and placement in a situated blocks world context. By relying on a rich, domain-specific adaptation of a generic ontology and a logical form structure produced by a semantic parser, we obviate the need for an intermediate, domain-specific representation and can produce a reasoner that grounds and reasons over concepts and constraints with real-valued data. This linguistic base enables more flexibility in interpreting natural language expressions invoking intrinsic concepts and features of structures and space. We demonstrate some of the capabilities of a system grounded in deep language understanding and present initial results in a structure learning task.

1 Introduction

Even as early as one of the first Blocks World natural language interaction systems, SHRDLU (Winograd, 1971), discussions about structures and space have been viewed as the foundation for future language understanding systems dealing with more abstract and higher-level concepts. Since then, the field has advanced in the task of learning how to understand utterances in Blocks World and other situated environments by using statistical methods grounding syntactic trees to entities and actions in the world to learn placement descriptions (Bisk et al., 2016), predicates (Kollar et al., 2013), actions (Kim and Mooney, 2012; She et al., 2014) or a combination of paths and actions (Tellex et al., 2011). However, rather than considering grounding solely as a mapping to actions and objects in the world, we use the deep language understanding capabilities of the TRIPS parser (Allen et al., 2008) to find deeper conceptual connections to primitive, composable, and often recursive aspects of structures, and use this knowledge to better understand conceptually-rich utterances without the need for training data. Inspired by the cognitive linguistic theory of conceptual mappings (Fauconnier, 1997), we focus on projection mappings between structure and set features and demonstrate instances of common situated language that makes use of such mappings. With these concepts grounded in a situated space, we believe we will be poised to extend the concepts in Blocks World into more abstract reasoning and language through grounded metaphor.

We also demonstrate the ability of our system to build up a model of a class of structures through natural language dialogue. Rather than constructing a new domain-specific representation for storing such knowledge, as in work by Hixon et al. (2015), we retain the semantic logical form structure as our base representation, using ontological concepts of comparison and semantic argument structures to ground concepts and predicates in the situated environment. We therefore aim to show that a linguistic structures from a semantic parser can serve as a strong base for reasoning and model-building in a situated context.

2 Capabilities and Tasks

We evaluate our system in a situated blocks world environment with 6-inch cubes placed on a table. Aside from unique identifiers for tracking, each cube is considered identical. Our physical apparatus consists of two Kinect 2.0's aimed at the table, with the multiple Kinects helping to avoid issues with block occlusion. The depth information is used to recognize and process position and rotation information that is then relayed to the system. Currently only block position is used, but orientation information is also recorded.

For user-system interaction, there is a screen at the end of the table across from the user which dis-

Proceedings of the First International Workshop on Spatial Language Understanding (SpLU-2018), pages 12–20
New Orleans, Louisiana, June 6, 2018. ©2018 Association for Computational Linguistics

plays an avatar that can speaks system-generated utterances and display it on-screen. The avatar can also point to blocks or locations on the table, although we do not use this functionality in our dialogues. When the system wants to place a block or provide an example structure, it generates a 3D image of the blocks overlaid with the existing scene that can be presented to the user or an assistant that will then place the blocks in the appropriate location. An image of the apparatus is shown in Figure 1.

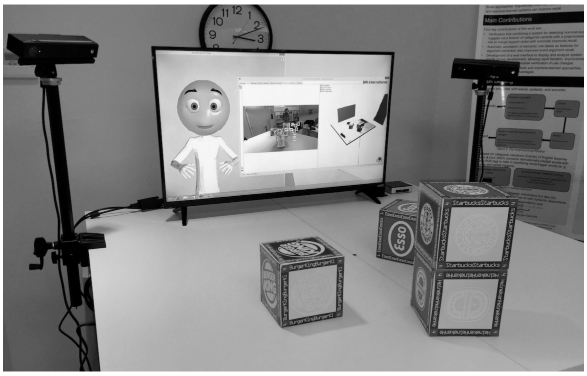

Figure 1: The apparatus used to interact with the system.

We focus on two tasks for evaluating our system within the context of a natural language dialogue system. The first is correctly understanding a variety of natural placement utterances and generating the expected placement of blocks that satisfies the command. There has been significant previous work on learning to interpret typical placement instructions (Bisk et al., 2016; Misra et al., 2017; Wang et al., 2016) or descriptions of block scenes (Bisk et al., 2018). While we have limited capabilities for understanding such instructions, this prior work is better suited for more robust and precise placement interaction that does not utilize conceptual composition. Therefore, rather than solely understanding simple phrases such as “Place a block on top of the leftmost block”, we focus our efforts towards understanding more complex phrases that utilize context, such as “Add another one,” and linguistic/semantic composition, as in, “Place three towers in a row with increasing height”.

The second task is teaching the system to learn a class of structures by providing it with a set of constraints. The user is provided with a number of positive and negative visual examples of a class of structures to learn (akin to resolving a Bongard problem (Bongard et al., 1970)), and once they have determined the underlying constraints of the structures, they must engage in a dialogue with the system to teach it the structure class so that it will be able to recognize structures belonging to that class. This task importantly differs from the first task and prior situated language understanding work in that the user is not communicating a specific goal structure to be achieved by the user through placement actions, but instead providing a set of general constraints and concepts that admit a number of possible structures.

3 System Architecture

We build upon the TRIPS architecture (Allen et al., 2001), which connects a number of components through message passing, with each component able to be tailored to a particular domain.

3.1 Semantic Extraction

The first component that sees user language input is the domain-general, rules-based TRIPS parser that is backed by a domain-generic ontology augmented with domain-specific concepts, such as blocks, rows, and columns. The output from the parser is a logical form semantic graph with a number of possible speech acts. This output is then passed to the Interpretation Manager (IM), which determines the appropriate speech act given dialogue context and further refines roles and interpretations according to domain-specific rules and ontological constraints.

3.2 Problem Solving Act Generation

Next, the output of the IM is processed by the Collaborative Problem Solving Agent, a central module that facilitates the acts that make up collaborative problem solving (i.e. the joint task actions carried out by the user and system). It passes the output to the Collaborative State Manager (CSM) to generate and store a new goal, query, or assertion. The appropriate act is then sent to the Behavioral Agent (BA), tasked with reasoning and acting in the environment. If the system has a query or goal proposal for the user, the message passing works in reverse, with the goal or query added to the goal hierarchy stored in the CSM. The current goal or query provides a context for resolving future utterances, providing additional information to aid in choosing the appropriate speech act.

3.3 Application to Tasks

In the structure building task, we primarily make use of the goal and sub-goal mechanisms to provide actions to the BA so that it can act in the environment towards the desired structure. The user can also provide assertions containing definitions of substructures to be used in the building process. In the structure learning task, we primarily process assertions that describe the general properties of the structure type, and queries to ask the user about properties or ask for an example. The task to be completed is determined by the user specifying the goal as the first utterance (e.g., "I want to build a staircase" versus "I want to teach you what a staircase is"). This top-level goal provides additional context for utterance resolution. For example, the utterance "The left-most block must be on top" would be processed as a proposed goal in the structure building task (as it describes a difference between the current and desired state), but as an assertion in the structure learning task (as it describes a property that should generally hold).

4 Semantic Logical Form Backing

Rather than convert semantic information from the semantic output of a domain-general parser, we directly use the semantic output of the TRIPS parser (backed by a combination of a domain-general and domain-specific ontology) as the underlying logical representation for assertions, constraints, and commands. This backing is enabled by a number of features specific to the output of the TRIPS parser. First, the ontology provides a method to generalize multiple related utterances or fragments to a single interpretation to be conveyed to the reasoning agent and handled similarly. Second, the semantic output of the TRIPS parser includes an ontology and tree-based representation of scales, which makes feature comparisons explicit and provides units for evaluating scales using the appropriate metric. Finally, the semantic roles (*figure* for head properties, and *ground* for reference properties) provide a more nuanced level of comparison among object and structure features than typical semantic parsers focused on events and higher-level interactions among people. For example, the sentence, "The left column is taller than the rightmost column," taller resolves to a concept ONT::MORE-VAL (enabling a simple operator extraction), with a *scale* of ONT::HEIGHT-SCALE, a *figure* of "the left column", and a *ground* of "the rightmost column".

Developing and relying on the semantic structure for reasoning provides a long-term advantage for extending the physical domain to handle reasoning in different domains or at a more abstract level. Currently, the structures used to tie the semantic structures to the domain could easily be extended to other domains simply by modifying the interpretation of predicates and generating new features, while referring expression and dialogue processing can remain largely unchanged. A metaphorical reasoning system, for example, could make use of the same semantic structures and simply modify the reasoning environment and generate inference from a physical simulation or concrete projection of abstract concepts, and could borrow predicates and features from Blocks World.

5 Predicates

Predicates describe binary positional aspects relating a block or structure to a particular context. All predicates have at least one argument, the subject, but typically also admit a context (e.g., other blocks, or the rest of the scene). For example, even though the *top* predicate may seem to take only one argument, we resolve it using a second argument that contains the complement of the scene (in the case of "the top block") or a contrast set (in the case of "the top block of the left column"). Predicates are used both for referring expressions to choose a particular group of blocks and for applying constraints to structure properties and placement instructions. For example, the command "the top block must be on the left" uses a predicate for both the referring expression (*the top block*) and a constraint on its location (*on the left*).

Rather than defining logical formulas for evaluating predicates, our predicates are designed programmatically using real-valued coordinates in 3D space with an emphasis on relations dealing with a vertical 2D plane between the user and the system's viewpoint. They are evaluated either by comparing positions and dimensions over the quantification of the blocks in the structures or over axis-aligned bounding boxes encapsulating the blocks. For example, the predicate *above(a,b)* requires that the x- and y-coordinate extents of the bounding boxes of the a and b intersect and that the minimum z-coordinate value of a is greater than the maximum z-value of b (with z being the vertical dimension). Each predicate is mapped to one or more TRIPS ontological concepts for evaluating when such a predicate appears in the logical form. The ontology is specific enough that no concept could yield more than one predicate interpretation. All predicates also include tolerances to account for real-world variations in the input data, but because of the nature of the depth data and the known size of the blocks, there is little noise in the positions of the blocks.

6 Features

The term "features" in the context of Blocks World refers to all quantifiable aspects of blocks and block arrangements. However, we also extend this definition to include potential ways of perceiving, discussing, or processing blocks and groups of blocks. For example, a set of blocks could be considered as a column, a row with or without a particular ordering, or simply a set of blocks with no relation to each other. The values of such features can be integers, real numbers, vectors, or an arrangement. Furthermore, arrangements can have multiple features assigned to them forming a feature group. For example, a sequence arrangement can generate a row feature, a column feature, the count of the number of blocks or structures within, an origin as a vector, and a direction as a vector.

6.1 Feature Mention Extraction

Given the semantic parser output, the reasoning agent parses the features described in multiple passes. First, referring expressions are extracted by finding mentions of objects that the reasoning agent knows how to recognize or instantiate in the environment (i.e., blocks, rows, columns, and spaces), and then storing constraints according to modifiers on its location (represented using predicates). Next, the features are extracted from the same parse tree, which typically contains a feature name as an arrangement name (e.g., a column), a scale (e.g., width-scale), or a number (e.g., the number of blocks in the specified set). Finally, the relevant operator (e.g., less than, at least, equal to) is extracted and sets up the constraint on the values or referenced features mentioned. In certain cases, the TRIPS parser explicitly provides the comparator (e.g., providing an ONT::MIN concept and appropriate arguments for "at least"), and in other cases, the comparator and its arguments must be inferred by the appearance of sets with a specified size parameter.

While certain features, like the size of a set, have an explicitly defined value, we also generate features that have an implicit value that may not be meaningful to the user. For example, linearity can take a value from 0-1 based on the deviation of the elements from a line of best fit. If the user states, "The bottom blocks must be in a line", we calculate the value and compare against a threshold to determine whether the constraint holds, or can compare using an operator against the linearity of another set of blocks. Features of this type are often difficult to explain linguistically or symbolically, and thus lean more on specific visual processing and could be tied to statistical computer vision models in the future.

6.2 Structure Models and Constraint Satisfaction

In the structure learning task, the system learns a set of constraints that describes a structure. As the goal is to teach the system a general concept rather than describe one particular instance, the learned constraints apply as rules that will apply in various configurations, rather than applying to particular blocks currently on the table. Therefore, referring expressions in the constraints for a model are reevaluated each time a particular instance is tested. We currently process four types of constraints: feature, predicate, structure, and existential constraints. Feature constraints, describe a property (such as width or height) that generally holds for the structure as a whole, such as "The height is at least 3 blocks". Predicate constraints enforce that a particular set of blocks satisfy a particular predicate (e.g., "The leftmost column is next to the center column"). Structural constraints enforce that the blocks referred to by a referring expression obeys a feature constraint (e.g., "The leftmost column has at least 3 blocks"). Predicate and structure constraints can also be modified to be satisfied if they are exclusively satisfied by only one grounding of an object type in a referring expression (e.g., "Only the leftmost column has a height greater than 2").

7 Recursive and Compositional Feature Understanding

Recursive and composition representations of features are essential for deep language understanding even in the simplified environment of Blocks

Ontological Concept	Lemmas	# of Arguments
ONT::ABOVE	above	2
W::HIGHER	higher	2
ONT::BELOW	below, beneath, under, underneath	2
W::LOWER	lower	2
ONT::ADJACENT	adjacent (to), next to, beside, by, contiguous (with), flush	2
ONT::CONNECTED	abut, adjoin, connect, touch	1,2
W::TOGETHER	together	1,2
ONT::ON	on, on top of	2
ONT::LEVEL	level with	1,2
ONT::TOP-LOC...	top	1,2
ONT::MIDDLE-LOC...	middle	(1),2
ONT::BOTTOM-LOC...	bottom	1,2
ONT::BETWEEN	(in) between	2
ONT::CENTER	center	(1),2
ONT::LEFT-LOC	left, lefthand, leftmost	(1),2
ONT::RIGHT-LOC	right, righthand, rightmost	(1),2
W::ANYWHERE	anywhere	1

Table 1: The list of predicates understood by the system, with their concept in the TRIPS ontology, the matching lemmas that can resolve to that concept during parsing (designated by hand or from WordNet mappings (Miller, 1995)), and the number of arguments each predicate can take. An argument number in parentheses indicates that the second argument, the reference, is inferred to be the scene complement of the first argument. Predicates like ONT::CONNECTED admit sets of blocks as their single argument.

World. Take for example the utterance “lengthen the first column of the row by 2”. Such an utterance refers to multiple features both for identifying the relevant set of blocks and for the desired action. However, beyond identifying the set of relevant blocks, it also enforces a conceptual model on the blocks that is necessary for the interpretation of “lengthen”, which requires a sequence rather than a set. Similarly, the notion of “first” implies an ordering of the row, taken in reading order (left-to-right) unless another context, such as a specified placement order, overwrites it.

The fact that these representations arise from simple interactions and often without explicit definition motivates our notion of such concepts as “features”. Thus in the above example, the placement of the blocks admits a “row” feature group consisting of a direction, a length, and a sequence of “column” features, each also having a sequence of blocks (which itself has a length feature) and an upward direction, as well as a “height”.

We also represent the composition of these features for utterances such as “place the columns in a row with increasing height”. Our main method for composition is projection, which in this context

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: 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 compared against thresholds or other sets of blocks.

we take to mean the reduction of features (with the number of features being the dimensionality of the concept) to enable composition with other features of the appropriate type. In this case, the phrase "increasing height" generates an increasing sequence of integers which is then projected onto the row to replace the individual height features of each of its constituent columns, generating the desired structure. Note that while the row itself could have a height as a structure as a whole, this single value would not be compatible with the sequence. This process is illustrated in Figure 2.

The example in Figure 2 also illustrates the advantage of such a technique in providing robustness in the face of linguistic ambiguity. For example, the "increasing height" could be modifying the placement of the columns, the columns themselves, or, as the TRIPS parser outputs, the row. Because of the restrictions on projection, we can correctly apply the modification to the relevant features even when the target of the modification is not directly modifiable in a way that parallels the semantic interpretation. In some cases, the ontological interpretation of projection-indicating terms differ from our interpretation, and such information must be discarded. For example, while the parser may extract the ONT::IN-LOC concept from the word "in", in the above example the ONT::MANNER concept is more appropriate. The projection restrictions allow us to determine the correct sense regardless of the specific concept while not being strictly dependent on the lemma.

7.1 Conceptual Features and Context

Moving away from the notion of sets as the only output of a referring expression confers an additional benefit in providing context for placement actions. Take, for example, the utterance "add another one". Treating the current set of blocks as a set of blocks would not provide the intended location of the next block (or group of blocks). To interpret such an utterance, we make use of discourse context, goal context, and the conceptual context of the last command. If the previous command involved an ordered sequence of some type of structure, "add another one" would make use of the conceptual context of the sequence which should be appended. In the case of a row, for example, we would pick the last element in the sequence, and place a duplicate in the next point when the direction vector is extended.

In certain cases, the conceptual context may not be available or sufficient. If the last utterance was "Place a block on top of the row", then "another one" might refer to either another block on top of the row or a block on top of the just placed block. In this case, we can make use of the overarching goal context. If the system is aware that the user is building a tower, increasing the height of the structure would be the expected next step. The system can make use of this context even without explicitly knowing the process for building a tower if the user provides a definition of the structure (e.g., "A tower is a structure taller than its diameter"), by choosing actions which bring the constraints closer to satisfaction.

8 Simulation and Querying Capabilities

While our system does not include a planner, we can nevertheless create a structure according to a set of constraints provided that the structure conforms to a grid-based structure. We generate a set of multiple iterations with blocks randomly dropped in a grid with the size determined either by default dimensions or constrained by global features of width and height. Once we find an arrangement that satisfies the constraints, we return the structure to the user to ask if the example is correct. In the 2D plane, the number of possible structures is constrained enough to generate an example satisfying the constraints in real-time. While we can then provide this structure to the user or assistant in the 3D view, we currently do not support generation of natural language descriptions for the placement of each block.

The system is also able to generate questions about structures when learning, in order to extract clear constraints from the user. When appropriate in dialogue, the system generates a random underspecified constraint that has not yet been mentioned, typically concerning general features (e.g., width or height), or more specific constraints (e.g., the placement restrictions of the top block). Specific constraints dealing with specific structures are generated based on user examples. For example, the system would ask about the placement of the top block only if there was a single top block in a previously shown example. In our architecture, we are then able to interpret a response fragment, fill the constraint parameters, and add the constraint to the model. We find that such questions greatly increase the quality of the user's given con-

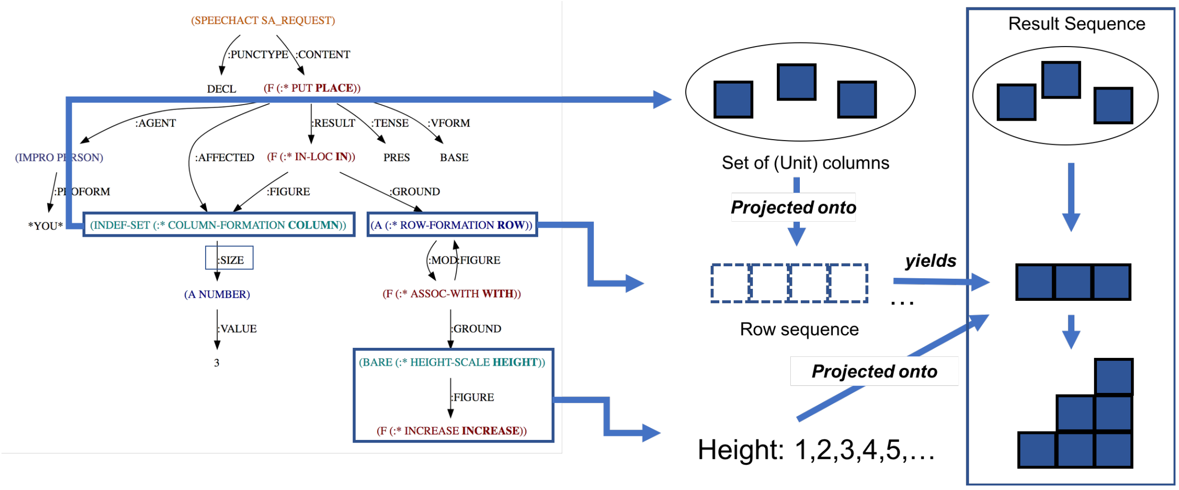

Figure 2: An example of the projection processing for the utterance "Place three columns in a row with increasing height." The features (boxed in the parse tree) are extracted and used to generate new individual instances with appropriate features. The columns are then projected into the appropriate feature of the row, and the height sequence projects onto the row of columns to create the final structure.

straints, as the questions provide an example of the types of constraints the system is most capable of handling and provide guidance for the user in organizing their conception of the structure class.

9 Evaluation

Currently evaluation is in preliminary stages, with an emphasis on expanding capabilities in terms of the variety of structures able to be built and recognized. A comprehensive evaluation task can be difficult for this system, given its symbolic backing. As there is no statistical learning, the usefulness of the system is primarily determined by the coverage of understood linguistic constructions at two levels – at the semantic parser level and at the level of interpretation given a correct semantic parse. One challenge faced in accurately evaluating the system is that users in a dialogue can be biased to choose language that the system understands, thereby reducing the average expressivity and linguistic complexity of their utterances. To partially address this, we have begun evaluations of our structure learning task, as we believe this task better illustrates the variety of language used to describe spatial concepts and structures, compared to the structure building task, which often consists solely of simple "place a block ⟨location⟩" utterances. Users are provided with positive and negative examples of a structure class (Figure 3) and must teach the system the concept.

Our initial evaluation to determine possible areas of improvement before actual trials began with

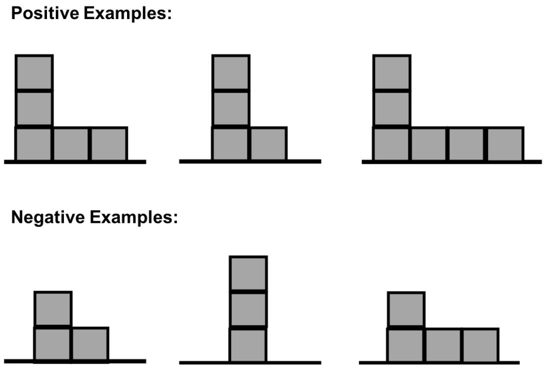

Figure 3: In the structure learning task, the user must determine the patterns followed by the positive examples that do not match the negative examples and explain these constraints to the system.

Wizard of Oz experiments. An advantage of a Wizard of Oz task is that the user maintains the complexity of their utterances as they see positive results from their descriptions, providing us with valuable insight into the various types of utterances our system must understand. We conducted 3 tasks with each of the 3 participants who were unaware of the details of the project, yielding 90 user utterances. A researcher took the place of the system by asking questions and presenting examples. By using a tablet that generated common questions and responses to be spoken by the avatar, these experiments could be carried out without the participants aware of the researcher's role as a stand-in for the system. To evaluate the system's performance at the semantic parser level, we ran the parser on the users' utterances in these experiments and correctly parsed 89% of the utterances. In evaluating the grounded interpretation level, we determine which of the correct parsed trees yield the desired constraint. Of the correctly parsed utterances, 90% yielded the desired constraint in the current state of the system with some development occurring after these trials. Further testing is required once the system's capabilities are finalized to see the performance of the system in an unrestricted setting with new users. Typical failures of the grounded interpretation level of the system involve utterances containing ordinal references (e.g., the second column), multiple values for constraints, ranges for feature values, or negative examples. All of these system abilities are near-future work. During this evaluation we also found that dialogue actions such as asking about features greatly simplified user utterances and therefore improved performance.

We also have begun trials of the complete dialogue system with promising results. The system is able to understand a number of different structure classes through dialogue, such as a staircase, an L shape, and a U shape. An example dialogue is shown in Figure 4, where the user was presented with the visual examples in Figure 3.

User: I want to teach you a shape.
System: Okay. Can you tell me something about the structure?
User: The bottom row must be at least 2.
System: Hmm, I didn't quite get that.
User: The bottom row has at least 2 blocks.
System: Okay. What is the greatest the width can be?
User: 5.
System: Okay, got it. Can you show me an example?
User: Yes. This is an example. ⟨*user shows example*⟩
System: Okay. Where can the top block be?
User: The top block is always on the left.
System: Okay.
User: ⟨*user builds example*⟩ Is this right?
System: 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 4: A user dialogue taken from initial runs of the evaluation.

10 Future Work

We plan to continue working towards improving robustness in evaluations, implementing new ways of discussing structure building and structural constraints, and providing more dialogue actions to guide the user through their explanation and deal with errors or misunderstood assertions. In addition, we will be creating a database of predicate and feature definitions with multimodal groundings to begin our long-term goal of extending these physical groundings of concepts into abstract domains. Currently the TRIPS architecture is used as a base for a number of domains involving dialogue-assisted creation, such as biological models, music composition, and automated movie direction, and therefore provides a strong base for extending such concepts.

11 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, Kevin J Shih, Yejin Choi, and Daniel Marcu. 2018. Learning Interpretable Spatial Operations in a Rich 3D Blocks World. In *Proceedings of the 32th AAAI Conference on Artificial Intelligence*.

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.

Gilles. Fauconnier. 1997. *Mappings in Thought and Language*. Cambridge University Press.

Ben Hixon, Peter Clark, and Hannaneh Hajishirzi. 2015. Learning Knowledge Graphs for Question Answering through Conversational Dialog. *NAACL*, pages 851–861.

Joohyun Kim and Raymond J Mooney. 2012. Unsupervised PCFG Induction for Grounded Language Learning with Highly Ambiguous Supervision. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing and Natural Language Learning (EMNLP-CoNLL '12)*, July, pages 433–444.

Thomas Kollar, Jayant Krishnamurthy, and Grant Strimel. 2013. Toward interactive grounded language acquisition. In *Robotics: Science and Systems (RSS)*.

George A. Miller. 1995. WordNet: a lexical database for English. *Communications of the ACM*, 38(11):39–41.

Dipendra Misra, John Langford, and Yoav Artzi. 2017. Mapping Instructions and Visual Observations to Actions with Reinforcement Learning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1004–1015.

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. *Proceedings of the SIGDIAL 2014 Conference*, (June):89–97.

Stefanie Tellex, Thomas Kollar, and Steven Dickerson. 2011. Understanding Natural Language Commands for Robotic Navigation and Mobile Manipulation. *AAAI*.

Sida I Wang, Percy Liang, and Christopher D Manning. 2016. Learning Language Games through Interaction. In *The 54th Annual Meeting of the Association for Computational Linguistics*, pages 2368–2378.

Terry Winograd. 1971. Procedures as a Representation for Data in a Computer for Understanding Natural Language. Technical report, Massachusetts Institute of Technology Artificial Intelligence.