Semantics Parsing to Probabilistic Programs for Situated Question Answering

Jayant Krishnamurthy
Allen Institute for Artificial Intelligence
2157 N. Northlake Way, Suite 110
Seattle, WA 98103 USA
jayantk@allenai.org

Oyvind Tafjord
Allen Institute for Artificial Intelligence
2157 N. Northlake Way, Suite 110
Seattle, WA 98103 USA
oyvindt@allenai.org

Abstract
Situated question answering is the problem of answering questions about an environment such as an image. This problem requires interpreting both a question and the environment, and is challenging because the set of interpretations is large, typically superexponential in the number of environmental objects. Existing models handle this challenge by making strong – and untrue – independence assumptions. We present Parsing to Probabilistic Programs (P³), a novel situated question answering model that utilizes approximate inference to eliminate these independence assumptions and enable the use of global features of the question/environment interpretation. Our key insight is to treat semantic parses as probabilistic programs that execute nondeterministically and whose possible executions represent environmental uncertainty. We evaluate our approach on a new, publicly-released data set of 5000 science diagram questions, finding that our approach outperforms several competitive baselines.

1 Introduction
Situated question answering is the problem of answering questions about an environment such as an image. Examples of this problem include visual question answering [Antol et al., 2015], robot direction following [Kollar et al., 2010], and even text question answering [Berant et al., 2014]. This problem requires interpreting both a question and the environment, and is challenging because the set of possible interpretations is large and exhibits complex dependence.

1. According to the given food chain, what is the number of organisms that eat deer? (A) 3 (B) 2 (C) 4 (D) 1
2. Based on the given food web, what would happen if there were no insect-eating birds? (A) The grasshopper population would increase. (B) The grasshopper population would decrease. (C) There would be no change in grasshopper number.

Figure 1: Two example food web questions. Answering these questions requires both question and environment (image) interpretation.

Figure 1 shows questions from an 8th grade science test that exemplify this challenge. These questions require the student to interpret an image depicting a food web, which shows the organisms in an ecosystem with arrows indicating what organisms eat. First, a computer could potentially extract many different food webs from the image. If each of the 15 text labels represents an organism, there are $15^2 = 225$ possible eats relations, and hence $2^{225}$ possible food webs. Second, these relations are not independent because food webs have global structural properties – e.g., they are typically directed acyclic graphs. Finally, the question and environment interpretations are dependent – e.g., an inter-
pretation is wrong if it identifies multiple answer options as correct. In order to reason about the large set of interpretations, existing possible worlds models for situated question answering make strong independence assumptions that preclude them from representing these complex dependencies (Matuszek et al., 2012; Krishnamurthy and Kollar, 2013; Malinowski and Fritz, 2014). This paper presents Parsing to Probabilistic Programs (\(P^3\)), a novel situated question answering model that eliminates these strong independence assumptions by embracing approximate inference. Our key idea is to perform semantic parsing and treat the semantic parse as a probabilistic program. Unlike a deterministic program, which always executes the same way, a probabilistic program may include nondeterministic choice primitives that enable it to have multiple executions, each of which may return a different value. \(P^3\) uses this set of executions to represent uncertainty over environment interpretations. Inference over environment representations becomes inference over a sequence of nondeterministic choice points in the program, which we approximate using beam search. This approximate inference procedure has linear running time in the number of choices and enables the model to use arbitrary global features of the question, semantic parse, environment, and current execution.

We present an experimental evaluation of \(P^3\) on a new data set of 8th grade science food web diagram questions (Figure 1). We compare our approach to several baselines, including possible worlds and neural network approaches, finding that \(P^3\) outperforms both. An ablation study demonstrates that global features help the model achieve high accuracy. Finally, we demonstrate that \(P^3\) improves performance on a previously published data set.

2 Prior Work

Possible world approaches to situated question answering combine both semantic parsing and environment interpretation models using a possible worlds semantics. Most work has assumed that these models are independent and the environment interpretation consists of independent predicate instances (Matuszek et al., 2012; Krishnamurthy and Kollar, 2013; Malinowski and Fritz, 2014). Seo et al., (2015) incorporate hard constraints on the joint question/environment interpretation; however, their approach does not generalize to soft constraints or arbitrary logical forms.

Another line of work has assumed a semantic parser is given, focusing exclusively on environment interpretation. This work includes robot direction following (Kollár et al., 2010; Fellex et al., 2011; Howard et al., 2014b; Howard et al., 2014a) and question answering against knowledge representations extracted from text (Berant et al., 2014; Krishnamurthy and Mitchell, 2015).

Neural networks have been recently used for visual question answering (Antol et al., 2015; Malinowski et al., 2015; Yang et al., 2015; Andreas et al., 2016a). Our work is most closely related to DNMNs (Andreas et al., 2016b), the only neural model that learns to semantically parse the question. The key distinction is that DNMNs use a continuous environment representation whereas we use a discrete representation. Continuous and discrete representations are well-suited to different tasks – in our case, food webs are naturally discrete. \(P^3\) can also use declaratively-specified background knowledge in the form of predefined functions.

Preliminaries for our work are semantic parsing and probabilistic programming. Semantic parsing is used to interpret questions against a deterministic execution model (e.g., a database), for applications such as question answering (Zettlemoyer and Collins, 2005; Liang et al., 2011), direction following (Chen and Mooney, 2011; Artzi and Zettlemoyer, 2013), and information extraction (Krishnamurthy and Mitchell, 2012). Our work extends semantic parsing to settings without a deterministic execution model.

Probabilistic programming languages extend deterministic languages with primitives for nondeterministic choice (Goodman and Stuhlmüller, 2014). We express logical forms in a probabilistic variant of Scheme similar to Church (Goodman et al., 2008); however, in this paper we use Python-like pseudocode for clarity. The language has a single nondeterministic primitive called \texttt{choose} that nondeterministically returns one of its arguments. For example \texttt{choose(1, 2, 3)} has three executions where it returns either 1, 2, or 3. Multiple calls to \texttt{choose} can be combined in a single program; for example,
choose(1,2) + choose(1,2) has four executions that return 2, 3, 3 and 4. Calls to choose are ordered such that the possible executions of a logical form can be viewed as a tree where each node represents a call to choose and each outbound edge represents a value that can be returned.

3 Parsing to Probabilistic Programs ($P^3$)

This section describes our model, Parsing to Probabilistic Programs ($P^3$). The input to the model is a question and an environment and its output is a denotation, which is a formal answer to the question. $P^3$ has two factors, a semantic parser and an execution model. The semantic parser scores syntactic parses and logical forms for the question; Figure 2 shows an example parse. These logical forms are probabilistic programs with multiple possible executions, each of which may return a different denotation. The set of executions is determined by an initialization program that defines the logical form’s functions using choose. Figure 3 shows pseudocode for the food web initialization program and Figure 4 shows how a logical form is executed. The execution model assigns a score to each of these executions given the environment. We defer additional application-specific discussion to Section 4.

Formally, $P^3$ is a loglinear model that predicts a denotation $\gamma$ for a question $q$ in an environment $v$ using three latent variables:

$$
P(\gamma|v, q; \theta) = \sum_{e, \ell, t} P(e, \ell, t|v, q; \theta) \mathbf{1}(\text{ret}(e) = \gamma)
$$

$$
P(e, \ell, t|v, q; \theta) = \frac{1}{Z_{q,v}} f_{ex}(e, \ell|v; \theta_{ex}) f_{prs}(\ell, t|q; \theta_{prs})
$$

The model is composed of two factors. $f_{prs}$ represents a semantic parser that scores logical forms $\ell$ and syntactic parse trees $t$ given question $q$ and parameters $\theta_{prs}$. $f_{ex}$ represents a nondeterministic execution model for logical forms. Given parameters $\theta_{ex}$, this factor assigns a score to a logical form $\ell$ and its execution $e$. The denotation $\gamma$, i.e., the formal answer to the question, is simply the value returned by $e$. $Z_{q,v}$ represents the model’s partition function.

The following sections describe $P^3$ in more detail.

3.1 Semantic Parsing

The factor $f_{prs}$ represents a Combinatory Categorial Grammar (CCG) semantic parser (Zettlemoyer and Collins, 2005) that scores logical forms for a question. Given a lexical mapping words to syntactic categories and logical forms, CCG defines a set of possible syntactic parses $t$ and logical forms $\ell$ for a question $q$. Figure 2 shows an example CCG parse. $f_{prs}$ is a loglinear model over parses $(\ell, t)$:

$$
f_{prs}(\ell, t|q; \theta_{prs}) = \exp\{\theta_{prs}^{T}\phi(\ell, t, q)\}
$$

\(^1\)In our experiments, we automatically learn the lexicon in a preprocessing step (Section 5.2).
The function \( \phi \) maps parses to feature vectors. We use a rich set of features similar to those for syntactic CCG parsing (Clark and Curran, 2007); a full description is provided in an online appendix.

### 3.2 Execution Model

The factor \( f_{\text{ex}} \) is a loglinear model over the executions of a logical form given an environment. Recall that logical forms in \( P^3 \) are probabilistic programs with a set of possible executions, each of which may return a different value. Each execution is a sequence, \( e = [e_0, e_1, e_2, \ldots, e_n] \), where \( e_0 \) is the program’s starting state, \( e_i \) represents the state immediately after the \( i \)th call to choose, and \( e_n \) is the state at termination. The score of an execution is:

\[
f_{\text{ex}}(e, \ell; \theta_{\text{ex}}) = \prod_{i=1}^{n} \exp\{\theta_{\text{ex}}^T \phi(e_{i-1}, e_i, \ell, v)\}
\]

In the above equation, \( \theta_{\text{ex}} \) represents the model’s parameters and \( \phi \) represents a feature function that produces a feature vector representing the difference between sequential program states \( e_{i-1} \) and \( e_i \) given environment \( v \) and logical form \( \ell \). Importantly, \( \phi \) can include arbitrary features of the execution, logical form, and environment. Such features are important in diagram question answering, for example, to detect cycles in the food web (see Section 4.4).

### 3.3 Inference

\( P^3 \) is designed to rely on approximate inference: our goal is to use rich features to accurately make local decisions, as in linear-time parsers (Nivre et al., 2006). We perform approximate inference using a two-stage beam search. Given a question \( q \), the first stage performs a beam search over CCG parses to produce a list of logical forms scored by \( f_{\text{prs}} \). This step is performed by using a CKY-style chart parsing algorithm then marginalizing out the syntactic parses. The second stage performs a beam search over executions of each logical form. This search first rewrites the logical form into continuation-passing style, then maintains a beam of continuations – each of which represents a partial execution – scored by \( f_{\text{ex}} \). The continuation-passing transformation allows choose to be implemented as a function that adds multiple continuations – one per return value – to the search queue (Goodman and Stuhlmüller, 2014). Executing a continuation runs the program until either the next call to choose or termination. Each step of the search executes every continuation in the beam, adding new continuations to the beam for the next step and storing any terminated executions in a separate queue. In our experiments, we use a beam size of 100 in the semantic parser, executing each of the 10 highest-scoring logical forms with a beam of 100 continuations.

### 3.4 Training

\( P^3 \) is trained by optimizing loglikelihood with stochastic gradient ascent. The training data \( \{(q^i, v^i, c^i)\}_{i=1}^{n} \) is a collection of questions \( q^i \) and environments \( v^i \) paired with supervision oracles \( c^i \). \( c^i(e) = 1 \) for a correct execution \( e \) and \( c^i(e) = 0 \) otherwise. The oracle \( c^i \) can be used to implement various kinds of supervision, including: (1) labeled denotations, by verifying the return value of \( e \) at termination and (2) labeled environments, by verifying decisions made by \( e \) about the environment interpretation. The oracle for diagram question answering combines both forms of supervision (Section 4.5).

The objective function \( O \) is the loglikelihood of predicting a correct execution:

\[
O(\theta) = \sum_{i=1}^{n} \log \sum_{e, l, t} c^i(e) P(e, l, t|q^i, v^i; \theta)
\]

We optimize this objective function using stochastic gradient ascent, using the approximate inference algorithm from Section 3.3 to estimate the necessary marginals. When computing the marginal distribution over correct executions, we filter each step of the beam search using the supervision oracle \( c^i \) to improve the approximation.

### 4 Diagram Question Answering with \( P^3 \)

As a case study, we apply \( P^3 \) to the task of answering diagram questions from an 8th grade science domain. There are a few steps required to apply \( P^3 \). First, we write an initialization program that defines the constituent functions of logical forms, the environment representation, and the corresponding uncertainty. Second, we create a component to select an answer given a denotation. Finally, we define the model’s features and supervision oracle.
4.1 Food Web Diagram Questions

We consider the task of answering food web diagram questions. The input consists of an image depicting a food web, a natural language question and a list of natural language answer options (Figure 1). The goal is to select the correct answer from these options. A food web represents the energy flow in an ecosystem as a directed graph where each vertex \( x \) represents an organism and an edge from \( x \) to \( y \) indicates that organism \( y \) eats organism \( x \). Many different kinds of questions can be asked about a food web, from simply what eats what, to the roles of animals and the consequences of population changes. This task has many regularities that require global features: for example, food webs are usually acyclic, and certain animals usually have certain roles (e.g., mice are herbivores). We have collected and released a data set for this task (Section 5.1).

We preprocess the images in the data set using a computer vision system that identifies candidate diagram elements (Kembhavi et al., 2016). This system extracts a collection of text labels (via OCR), arrows, arrowheads, and objects, each with corresponding scores. It also extracts a collection of scored linkages between these elements. These extractions are noisy and contain many discrepancies such as overlapping text labels and spurious linkages. We use these extractions to define a set of candidate organisms (using the text labels), and also to define features of the execution model.

4.2 Initialization Program

The initialization program for diagram questions defines a distribution over partial food webs given a logical form. It assumes that each diagram has a known set of entities, which in our case is the set of extracted text labels. The program defines a collection of learned predicates over these entities to represent depicted food web; these predicates invoke choose to represent uncertainty. The program also includes deterministic functions that encode background knowledge. Figure 4 shows pseudocode for a portion of the initialization program.

Food webs are represented using two learned predicates: ORGANISM\( (x) \) indicates whether the text label \( x \) is an organism (as opposed to, e.g., the image title); and EATS\( (x, y) \) indicates whether organism \( x \) eats organism \( y \). One way to represent uncertainty over food webs is to use choose to nondeterministically select the truth value of every predicate instance in the initialization program, thereby defining a possible worlds-style distribution over all food webs. However, this representation may resolve more uncertainty than necessary for a particular logical form; intuitively, executing \( \lambda x. \)EATS\( (x, \text{MOUSE}) \) only requires the values of a handful of predicate instances involving the mouse. We instead represent uncertainty using a just-in-time approach that only chooses values for predicate instances necessary to produce the logical form’s denotation. This approach initializes every predicate instance’s value to undefined, delaying choosing its value until an execution tries to use it.

The initialization program also includes several other deterministic functions of the learned predicates. One set of functions consists of animal role predicates, such as HERBIVORE\( (x) \), that are defined in terms of EATS. Another set of functions represents population change events and reasons about their consequences. Finally, the program also defines operators such as COUNT.
4.3 Answer Selection

The answer selection component uses string match heuristics and an LSTM to produce a distribution over multiple choice answers given a distribution over denotations predicted by \( P^3 \). The string match heuristics score each answer option given a denotation then select the highest scoring answer, abstaining in the case of a tie. The score computation depends on the denotation’s type. If the denotation is a set of entities, the score is an approximate count of the number of entities in the denotation that are mentioned in the answer based on a fuzzy string match. If the denotation is a set of change events, the score is a fuzzy match of both the change direction and the animal name. If the denotation is a number, string matching is straightforward. Applying these heuristics and marginalizing out denotations yields a distribution over answer options.

A limitation of the above approach is that it does not directly incorporate linguistic prior knowledge about likely answers. For example, “snake” is usually a good answer to “what eats mice?” regardless of the diagram. Such knowledge is known to be essential for visual question answering (Antol et al., 2015; Andreas et al., 2016b) and important in our task as well. We incorporate this knowledge in a standard way, by training a neural network on question/answer pairs (without the diagram) and combining its predictions with the string match heuristics above. The network is a sequence LSTM that is applied to the question concatenated with each answer option \( a \) to produce a 50-dimensional vector \( v_a \) for each answer. The distribution over answers is the softmax of the inner product of these vectors with a learned parameter vector \( w \). For simplicity, we combine these two components using a 50/50 mix of their answer distributions.

4.4 Execution Features

The execution model uses three sets of features: instance features, predicate features, and denotation features. Instance features treat each predicate instance independently, while the other features are functions of multiple predicate instances and the logical form. We provide a complete listing of features in an online appendix.

Instance features fire whenever an execution chooses a truth value for a predicate instance. These features are similar to the per-predicate-instance features used in prior work to produce a distribution over possible worlds. For \( \text{ORGANISM}(x) \), our features are the vision model’s extraction score for \( x \) and indicator features for the number of tokens in \( x \). For \( \text{EATS}(x, y) \), our features are various combinations of the vision model’s scores for arrows that may connect the text labels \( x \) and \( y \).

Predicate features fire based on the global assignment of truth values to all instances of a single predicate. The features for \( \text{ORGANISM} \) count occurrences of overlapping text labels among true instances. The features for \( \text{EATS} \) include cycle count features for various cycle lengths and arrow reuse features. The cycle count features help the model learn that food webs are typically, but not always, acyclic and the arrow reuse features aim to prevent the model from predicting two different \( \text{EATS} \) instances on the basis of a single arrow.

Denotation features fire on the return value of an execution. There are two kinds of denotation features: size features that count the number of entities in denotations of various types; and denotation element features for specific logical forms. The second kind of feature can be used to learn that the denotation of \( \lambda x. \text{HERBIVORE}(x) \) is likely to contain the entity \( \text{MOUSE} \), but unlikely to contain \( \text{WOLF} \).

4.5 Supervision Oracle

The supervision oracle for diagram question answering combines supervision of both answers and environment interpretations. We assume that each diagram has been labeled with a food web. An execution is correct if and only if (1) all of its global variables encoding the food web are consistent with the labeled food web, and (2) string match answer selection applied to its denotation chooses the correct answer. The first constraint guarantees that every logical form has at most one correct execution for any given diagram.

5 Evaluation

Our evaluation compares \( P^3 \) to both possible worlds and neural network approaches on our data set of food web diagram questions. An ablation study demonstrates that both sets of global features im-
prove accuracy. Finally, we demonstrate $P^3$'s generality by applying it to a previously-published data set, obtaining state-of-the-art results.

We have included the data set with the submission, and will release code if the paper is published.

5.1 FOODWEBS Data Set

FOODWEBS consists of $\sim 500$ food web diagrams and $\sim 5000$ questions designed to imitate actual questions encountered on 8th grade science exams. The train/validation/test sets contain $\sim 300/100/100$ diagrams and their corresponding questions. The data set has three kinds of annotations in addition to the correct answer for each question. First, each diagram is annotated with the food web that it depicts using the ORGANISM and EATS predicates. Second, each diagram has predictions from a vision system for various diagram elements such as arrows and text labels (Kembhavi et al., 2016). These are noisy predictions, not ground truth. Finally, each question is annotated by the authors with a logical form or null if its meaning cannot be represented using our predicates. These logical forms are not used to train $P^3$ but are useful to measure per-component error.

We collected FOODWEBS in two stages. First, we collected a number of food web diagrams using an image search engine. Second, we generated questions for these diagrams using Mechanical Turk. Workers were shown a food web diagram and a real exam question for inspiration and asked to write a new question and its answer options. We validated each generated question by asking 3 workers to answer it, discarding questions where at least 2 did not choose the correct answer. The authors also manually corrected any ambiguous (e.g., two answer options are correct) and poorly-formatted (e.g., two answer options have the same letter) questions.

5.2 Baseline Comparison

Our first experiment compares $P^3$ with several baselines for situated question answering. The first baseline, WORLD, is a possible worlds model based on Malinowski and Fritz (2014). This baseline learns a semantic parser $P(\ell, q | g)$ and a distribution over food webs $P(w | v)$, then evaluates $\ell$ on $w$ to produce a distribution over denotations. This model is implemented by independently training $P^3$'s CCG parser (on question/answer pairs and labeled food webs) and a possible-worlds execution model (on labeled food webs). The CCG lexicon for both $P^3$ and WORLD was generated by applying PAL (Krishnamurthy, 2016) to the same data. Finally, both models select answers as described in Section 4.3.

We also compared $P^3$ to several neural network baselines. The first baseline, LSTM, is the text-only answer selection model described in Section 4.3. The second baseline, DQA, is a diagram question answering model that uses a predicted “diagram parse” of the image (Kembhavi et al., 2016). This model is trained with question/answer pairs and diagram parses, which are roughly comparable to labeled food webs. The third baseline, VQA, is a neural network for visual question answering. This model represents each image as a vector by using the final layer of a pre-trained VGG19 model (Simonyan and Zisserman, 2014) and applying a single fully-connected layer. It scores answer options by using a sequence LSTM to encode the question/answer pair, then computing a dot product between the text and image vectors.

Table 1 compares the accuracy of $P^3$ to these baselines. Accuracy is the fraction of questions answered correctly. LSTM performs well on this data set, suggesting that many questions can be answered without using the image. This result is consistent
with other multiple-choice situated question answering tasks (Antol et al., 2015). Only \( P^3 \) substantially improves accuracy over this strong baseline. We conducted a second experiment that limits the value of linguistic prior knowledge in order to better understand this behavior. We ran each model on a test set with unseen organisms created by reversing the animal names in the question and diagram. On this test set, VQA and DQA again perform similarly to LSTM, suggesting that these models largely learn the same linguistic prior. However, WORLDS and \( P^3 \) outperform LSTM because they learn to interpret the diagram. On both data sets, \( P^3 \) outperforms WORLDS due to its global features.

5.3 Ablation Study

We performed an ablation study to further understand the impact of LSTM answer selection and global features. Table 2 shows the accuracy of \( P^3 \) trained without these components. We find that LSTM answer selection improves accuracy by 9 points, as expected due to the importance of linguistic prior knowledge. Global features improve accuracy by 7 points, which is roughly comparable to the delta between \( P^3 \) and WORLDS in Table 1.

5.4 Component Error Analysis

Our third experiment analyses the sources of error by training and evaluating \( P^3 \) while providing the gold logical form, food web, or both as input. Table 3 shows the accuracy of these three models. The final entry shows the maximum accuracy possible given our logical form language and answer selection. The larger performance improvement with gold food webs suggests that the execution model is responsible for more error than semantic parsing, although both components contribute.

5.5 SCENE Experiments

Our final experiment applies \( P^3 \) to the SCENE data set of Krishnamurthy and Kollar (2013). The task in this data set is to identify the set of objects in an image denoted by a natural language expression, such as “blue mug to the left of the monitor.” We use the provided CCG lexicon and predicate vocabulary, creating a learned predicate in the initialization program for each. We use the provided instance features, adding predicate and denotation size features.

| Model | Accuracy | \( \Delta \) |
|-------|----------|-------------|
| \( P^3 \) | 69.1 | |
| + gold logical form | 75.1 | +6.0 |
| + gold food webs | 82.3 | +13.2 |
| + both | 91.6 | +22.5 |

Table 3: Accuracy of \( P^3 \) when trained and evaluated with labeled logical forms, food webs, or both.

| Model | QA | QA+E | QA+E+LF |
|-------|----|------|--------|
| \( P^3 \) | 68 | 75 | – |
| KK2013 | 67 | – | 70 |

Table 4: Accuracy on the SCENE data set. KK2013 results are from Krishnamurthy and Kollar (2013).

Table 4 compares \( P^3 \) to prior work on SCENE. We consider three different supervision conditions: QA trains with question/answer pairs, QA+E further includes labeled environments, and QA+E+LF further includes labeled logical forms. We trained \( P^3 \) in the first two conditions, while prior work trained in the first and third conditions. \( P^3 \) slightly outperforms in the QA condition and \( P^3 \) trained with labeled environments outperforms prior work trained with labeled environments and logical forms.

6 Conclusion

We present Parsing to Probabilistic Programs (\( P^3 \)), a novel model for situated question answering that embraces approximate inference to enable the use of arbitrary features of the language and environment. \( P^3 \) trains a semantic parser to predict logical forms that are probabilistic programs whose possible executions represent environmental uncertainty. We demonstrate this model on a challenging new data set of 5000 science diagram questions, finding that it outperforms several competitive baselines and that its global features improve accuracy.

\( P^3 \) has several advantageous properties. First, \( P^3 \) can be easily applied to new problems: one simply has to write an initialization program and define the execution features. Second, the initialization program can be used to encode a wide class of assumptions about the environment. For example, the model can assume that every noun refers to a single object. The combination of semantic parsing and probabilistic programming makes \( P^3 \) an expressive and flexible model with many potential applications.
References

[Andreas et al.2016a] Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. 2016a. Deep compositional question answering with neural module networks. In CVPR.

[Andreas et al.2016b] Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. 2016b. Learning to compose neural networks for question answering. In NAACL.

[Antol et al.2015] Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C. Lawrence Zitnick, and Devi Parikh. 2015. VQA: Visual question answering. In International Conference on Computer Vision (ICCV).

[Artzi and Zettlemoyer2013] Yoav Artzi and Luke Zettlemoyer. 2013. Weakly supervised learning of semantic parsers for mapping instructions to actions. Transactions of the Association for Computational Linguistics.

[Berant et al.2014] Jonathan Berant, Vivek Srikumar, Pei-Chun Chen, Abby Vander Linden, Brittany Harding, Brad Huang, Peter Clark, and Christopher D. Manning. 2014. Modeling biological processes for reading comprehension. In Proceedings of EMNLP.

[Chen and Mooney2011] David L. Chen and Raymond J. Mooney. 2011. Learning to interpret natural language navigation instructions from observations. In Proceedings of the 25th AAAI Conference on Artificial Intelligence.

[Clark and Curran2007] Stephen Clark and James R. Curran. 2007. Wide-coverage efficient statistical parsing with CCG and log-linear models. Computational Linguistics, 33(4):493–552.

[Goodman and Stuhlmüller2014] Noah D Goodman and Andreas Stuhlmüller. 2014. The Design and Implementation of Probabilistic Programming Languages. http://dippl.org. Accessed: 2016-2-25.

[Goodman et al.2008] Noah D. Goodman, Vikash K. Mansinghka, Daniel M. Roy, Keith Bonawitz, and Joshua B. Tenenbaum. 2008. Church: A language for generative models. In Uncertainty in Artificial Intelligence.

[Howard et al.2014a] Thomas M. Howard, Istvan Chung, Oron Propp, Matthew R. Walter, and Nicholas Roy. 2014a. Efficient natural language interfaces for assistive robots. In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) Workshop on Rehabilitation and Assistive Robotics, September.

[Howard et al.2014b] Thomas M Howard, Stefanie Tellex, and Nicholas Roy. 2014b. A natural language planner interface for mobile manipulators. In 2014 IEEE International Conference on Robotics and Automation (ICRA).

[Kembhavi et al.2016] Aniruddha Kembhavi, Mike Salvato, Eric Kolve, Min Joon Seo, Hannaneh Hajishirzi, and Ali Farhadi. 2016. A diagram is worth a dozen images. CoRR, abs/1603.07396.

[Kollar et al.2010] Thomas Kollar, Stefanie Tellex, Deb Roy, and Nicholas Roy. 2010. Toward understanding natural language directions. In Proceedings of the 5th ACM/IEEE International Conference on Human-Robot Interaction.

[Krishnamurthy and Kollár2013] Jayant Krishnamurthy and Thomas Kollár. 2013. Jointly learning to parse and perceive: Connecting natural language to the physical world. Transactions of the Association of Computational Linguistics – Volume 1.

[Krishnamurthy and Mitchell2012] Jayant Krishnamurthy and Tom M. Mitchell. 2012. Weakly supervised training of semantic parsers. In Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning.

[Krishnamurthy and Mitchell2015] Jayant Krishnamurthy and Tom M. Mitchell. 2015. Learning a compositional semantics for freebase with an open predicate vocabulary. Transactions of the Association for Computational Linguistics, 3:257–270.

[Krishnamurthy2016] Jayant Krishnamurthy. 2016. Probabilistic models for learning a semantic parser lexicon. In NAACL.

[Liang et al.2011] Percy Liang, Michael I. Jordan, and Dan Klein. 2011. Learning dependency-based compositional semantics. In Proceedings of the Association for Computational Linguistics.

[Malinowski and Fritz2014] Mateusz Malinowski and Mario Fritz. 2014. A multi-world approach to question answering about real-world scenes based on uncertain input. In Advances in Neural Information Processing Systems.

[Malinowski et al.2015] Mateusz Malinowski, Marcus Rohrbach, and Mario Fritz. 2015. Ask your neurons: A neural-based approach to answering questions about images. In International Conference on Computer Vision.

[Matuszek et al.2012] Cynthia Matuszek, Nicholas FitzGerald, Luke Zettlemoyer, Liefeng Bo, and Dieter Fox. 2012. A joint model of language and perception for grounded attribute learning. In Proceedings of the 29th International Conference on Machine Learning.

[Nivre et al.2006] Joakim Nivre, Johan Hall, and Jens Nilsson. 2006. Maltparser: A data-driven parser-generator for dependency parsing. In Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics.
[Seo et al.2015] Minjoon Seo, Hannaneh Hajishirzi, Ali Farhadi, Oren Etzioni, and Clint Malcolm. 2015. Solving geometry problems: Combining text and diagram interpretation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

[Simonyan and Zisserman2014] Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556.

[Tellex et al.2011] Stefanie Tellex, Thomas Kollar, Steven Dickerson, Matthew Walter, Ashis Banerjee, Seth Teller, and Nicholas Roy. 2011. Understanding natural language commands for robotic navigation and mobile manipulation. In *AAAI Conference on Artificial Intelligence*.

[Yang et al.2015] Zichao Yang, Xiaodong He, Jianfeng Gao, Li Deng, and Alexander J. Smola. 2015. Stacked attention networks for image question answering. *arXiv preprint arXiv:1511.02274*.

[Zettlemoyer and Collins2005] Luke S. Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form: structured classification with probabilistic categorial grammars. In *UAI ’05, Proceedings of the 21st Conference in Uncertainty in Artificial Intelligence*. 