Towards General Natural Language Understanding with Probabilistic Worldbuilding

Abulhair Saparov and **Tom M. Mitchell**
Machine Learning Department,
Carnegie Mellon University, USA
{asaparov, tom.mitchell}@cs.cmu.edu

Abstract

We introduce the Probabilistic Worldbuilding Model (PWM), a new fully symbolic Bayesian model of semantic parsing and reasoning, as a first step in a research program toward more domain- and task-general NLU and AI. Humans create internal mental models of their observations that greatly aid in their ability to understand and reason about a large variety of problems. In PWM, the meanings of sentences, acquired facts about the world, and intermediate steps in reasoning are all expressed in a human-readable formal language, with the design goal of interpretability. PWM is Bayesian, designed specifically to be able to generalize to new domains and new tasks. We derive and implement an inference algorithm that reads sentences by parsing and abducing updates to its latent world model that capture the semantics of those sentences, and evaluate it on two out-of-domain question-answering datasets: (1) ProofWriter and (2) a new dataset we call FictionalGeoQA, designed to be more representative of real language but still simple enough to focus on evaluating reasoning ability, while being robust against heuristics. Our method outperforms baselines on both, thereby demonstrating its value as a proof-of-concept.

1 Introduction

Despite recent progress in AI and NLP producing algorithms that perform well on a number of NLP tasks, it is still unclear how to move forward and develop algorithms that understand language as well as humans do. In particular, large-scale language models such as BERT (Devlin et al., 2019), RoBERTa (Liu et al., 2019), GPT-3 (Brown et al., 2020), XLNet (Yang et al., 2019), and others were trained on a very large amount of text and can then be applied to perform many different NLP tasks after some fine-tuning. In the case of GPT-3, some tasks require very few additional training examples to achieve state-of-the-art performance. As a result of training on text from virtually every domain, these models are domain-general. This is in contrast with NLP algorithms that are largely trained on one or a small handful of domains, and as such, are not able to perform well on new domains outside of their training. Despite this focus on domain-generality, there are still a large number of tasks on which these large-scale language models perform poorly (Dunietz et al., 2020). Many limitations of today's state-of-the-art methods become evident when comparing with the human ability to understand language (Lake et al., 2016; Tamari et al., 2020; Bender and Koller, 2020; Gardner et al., 2019; Linzen, 2020). Many cognitive scientists posit that humans create rich mental models of the world from their observations which provide superior explainability, reasoning, and generalizability to new domains and tasks. How do we, as a field, move from today's state-of-the-art to more general intelligence? What are the next steps to develop algorithms that can generalize to new tasks at the same level as humans? The lack of interpretability in many of these models makes these questions impossible to answer precisely. One promising direction is to change the evaluation metric: Brown et al. (2020), Linzen (2020), and many others have suggested *zero-shot* or *few-shot accuracy* to measure the performance of algorithms (i.e., the algorithm is evaluated with a new dataset, wholly separate from its training; or in the case of few-shot learning, save for a few examples). While this shift is welcome, it alone will not solve the above issues.

We introduce the *Probabilistic Worldbuilding Model* (PWM), a probabilistic generative model of reasoning and semantic parsing. Like some past approaches, PWM explicitly builds an internal mental model, which we call the *theory* (Tamari et al., 2020; Hogan et al., 2021; Mitchell et al., 2018; Charniak and Goldman, 1993). The theory constitutes what the algorithm believes to

Transactions of the Association for Computational Linguistics, vol. 10, pp. 325–342, 2022. https://doi.org/10.1162/tacl_a_00463
Action Editor: Hinrich Schütze. Submission batch: 3/2021; Revision batch: 9/2021; Published 4/2022.

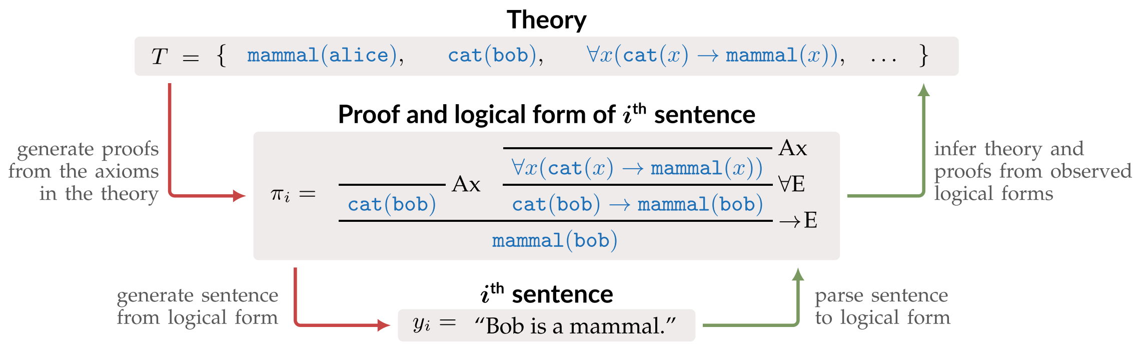
Theory
T = { mammal(alice), cat(bob), ∀x(cat(x) → mammal(x)), ... }
generate proofs from the axioms in the theory
Proof and logical form of i^th sentence
π_i =
cat(bob)
Ax
∀x(cat(x) → mammal(x))
Ax
∀E
cat(bob) → mammal(bob)
→E
mammal(bob)
infer theory and proofs from observed logical forms
generate sentence from logical form
i^th sentence
y_i = "Bob is a mammal."
parse sentence to logical form

Figure 1: The generative process and inference in our model, with an example of a theory, generating a proof of a logical form which itself generates the sentence ''Bob is a mammal.'' During inference, only the sentences are observed, whereas the theory and proofs are latent. Given sentence y_i, the language module outputs the logical form. The reasoning module then infers the proof π_i of the logical form and updates the posterior of the theory T.

be true. PWM is fully symbolic and Bayesian, using a single unified human-readable formal language to represent all meaning, and is therefore *inherently interpretable*. This is in contrast to systems that use subsymbolic representations of meaning for some or all of their components. Every random variable in PWM is well-defined with respect to other random variables and/or grounded primitives. Prior knowledge such as the rules of deductive inference, the structure of English grammar, and knowledge of basic physics and mathematics can be incorporated by modifying the prior distributions of the random variables in PWM. Incorporating prior knowledge can greatly reduce the amount of training data required to achieve sufficient generalizability, as we will demonstrate. Extensibility is key to future research that could enable more general NLU and AI, as it provides a clearer path forward for future exploration.

We present an implementation of inference under the proposed model, called the *Probabilistic Worldbuilding from Language* (PWL). While PWM is an abstract mathematical description of the underlying distribution of axioms, proofs, logical forms, and sentences, PWL is the algorithm that reads sentences, computes logical form representations of their meaning, and updates the axioms and proofs in the theory accordingly. See Figure 1 for a high-level schematic diagram of PWM and PWL. PWM describes the process depicted by the red arrows, whereas PWL is the algorithm depicted by the green arrows. We emphasize that the reasoning in PWL is not a theorem prover and is not purely deductive. Instead, PWL solves a different problem of finding satisfying *abductive* proofs, which is computationally easier than deductive inference: Given a set of observations, work backwards to find a set of axioms that deductively *explain* the observations. It is these abduced axioms that constitute the internal ''mental model.'' Humans often rely on abductive reasoning, for example in commonsense reasoning (Bhagavatula et al., 2020; Furbach et al., 2015).

A core principle of our approach is to ensure *generality by design*. Simplifying assumptions often trade away generality for tractability, such as by restricting the representation of the meanings of sentences, or number of steps during reasoning. PWM is designed to be domain- and task-general, and to this end, uses higher-order logic (i.e., lambda calculus) (Church, 1940) as the formal language, which we believe is sufficiently expressive to capture the meaning of declarative and interrogative sentences in natural language. Furthermore, PWM uses *natural deduction* for reasoning, which is *complete* in that if a logical form ϕ is true, there is a proof of ϕ (Henkin, 1950).

In Section 3, we describe PWM and PWL more precisely. In Section 4, as a proof-of-concept of the value of further research, we run experiments on two question-answering datasets: ProofWriter (Tafjord et al., 2021) and a new dataset, called FictionalGeoQA, which we specifically created to evaluate the ability to reason over short paragraphs while being robust against simpler heuristic strategies. Unlike ProofWriter, the text in FictionalGeoQA was not template-generated and is more realistic, but is still simple enough to focus the evaluation on reasoning rather than parsing, with many sentences having semantics that go beyond the Horn clause fragment of first-order logic. PWL outperforms the baselines with respect to zero-shot accuracy (i.e., without looking at any training examples). Our code and data is freely available at `github.com/asaparov/PWL` and `github.com/asaparov/fictionalgeoqa`.

In summary, the primary contributions of this paper are the following:

- PWM, a new model for more general NLU, and PWL, the implementation that reads sentences, computes their logical forms, and updates its theory accordingly.
- Introducing FictionalGeoQA, a new question-answering dataset designed to evaluate the ability to reason over language.
- Experiments on ProofWriter and FictionalGeoQA demonstrating that PWL outperforms baseline methods on question-answering.

2 Related Work

Fully symbolic methods were commonplace in earlier AI research (Newell and Simon, 1976; Dreyfus, 1985). However, they were oftentimes brittle: A new observation would contradict the internal theory or violate an assumption, and it was not clear how to resolve the impasse in a principled manner and proceed. But they do have some key advantages: Symbolic approaches that use well-studied human-readable formal languages such as first-order logic, higher-order logic, type theory, etc. enable humans to readily inspect and understand the internal processing of these algorithms, effecting a high degree of interpretability (Dowty, 1981; Gregory, 2015; Cooper et al., 2015). Symbolic systems can be made general by design, by using a sufficiently expressive formal language and ontology. Hybrid methods have been explored to alleviate the brittleness of formal systems while engendering their strengths, such as interpretability and generalizability; for example, the recent work in *neuro-symbolic* methods (Yi et al., 2020; Saha et al., 2020; Tafjord et al., 2021). Neural theorem provers are in this vein (Rocktäschel and Riedel, 2017). However, the proofs considered in these approaches are based on *backward chaining* (Russell and Norvig, 2010), which restricts the semantics to the Horn clause fragment of first-order logic. Sun et al. (2020), Ren et al. (2020), and Arakelyan et al. (2021) extend coverage to the existential positive fragment of first-order logic. In natural language, sentences express more complex semantics such as negation, nested universal quantification, and higher-order structures. Our work explores the other side of the tradeoff between tractability and expressivity/generality. Theorem provers attempt to solve the problem of deduction: finding a proof of a given formula, given a set of axioms. In contrast, the reasoning component of PWM is abductive, and the abduced axioms can be used in downstream tasks, such as question-answering, and to better read new sentences in the context of the world model, as we will demonstrate. We posit that abduction is sufficient for more general NLU (Hobbs, 2006; Hobbs et al., 1993). PWM combines Bayesian statistical machine learning with symbolic representations in order to handle uncertainty in a principled manner, ''smoothing out'' or ''softening'' the rigidity of a purely symbolic approach. In PWM, the internal theory is a random variable, so if a new observation is inconsistent with the theory, there may be other theories in the probability space that are consistent with the observation. The probabilistic approach provides a principled way to resolve these impasses.

PWM is certainly not the first to combine symbolic and probabilistic methods. There is a rich history of *inductive logic programming* (ILP) (Muggleton, 1991; Cropper and Morel, 2021) and probabilistic ILP languages (Muggleton, 1996; Cussens, 2001; Sato et al., 2005; Bellodi and Riguzzi, 2015). These languages could be used to learn a ''theory'' from a collection of observations, but they are typically restricted to learning rules in the form of first-order Horn clauses, for tractability. In natural language, it is easy to express semantics beyond the Horn clause fragment of first-order logic.

Knowledge bases (KBs) and *cognitive architectures* (Kotseruba and Tsotsos, 2020; Hogan et al., 2021; Laird et al., 1987; Mitchell et al., 2018) have attempted to explicitly model domain-general knowledge in a form amenable to reasoning. Cognitive architectures aim to more closely replicate human cognition. Some approaches use probabilistic methods to handle uncertainty (Niepert et al., 2012; Niepert and Domingos, 2015; Jain et al., 2019). However, many of these approaches make strong simplifying assumptions that restrict the expressive power of the formal language that expresses facts in the KB. For example, many KBs can be characterized as graphs, where each entity corresponds to a vertex and every fact corresponds to a labeled edge. For example, the belief plays_sport(s_williams, tennis) is representable as a directed edge connecting the vertex s_williams to the vertex tennis, with the edge label plays_sport. While this assumption greatly aids tractability and scalability, allowing many problems in reasoning to be solved by graph algorithms, it greatly hinders expressivity and generality, and there are many kinds of knowledge that simply cannot be expressed and represented in such KBs. PWM does not make such restrictions on logical forms in the theory, allowing for richer semantics, such as definitions, universally quantified statements, conditionals, etc.

3 Model

In this section, we provide a mathematical description of PWM. At a high level, the process for generating a sentence sampled from this probability distribution is:

1. Sample the theory T from a prior distribution $p(T)$. T is a collection of logical forms in higher-order logic that represent what PWL believes to be true.

2. For each observation i, sample a proof π_i from $p(\pi_i \mid T)$. The conclusion of the proof is the logical form x_i, which represents the meaning of the i^{th} sentence.

3. Sample the i^{th} sentence y_i from $p(y_i \mid x_i)$.

Inference is effectively the inverse of this process, and is implemented by PWL. During inference, PWL is given a collection of observed sentences $y_1, \ldots, y_n$ and the goal is to discern the value of the latent variables: the logical form of each sentence $\boldsymbol{x} \triangleq \{x_1, \ldots, x_n\}$, the proofs for each logical form $\boldsymbol{\pi} \triangleq \{\pi_1, \ldots, \pi_n\}$, and the underlying theory T. Both the generative process and inference algorithm naturally divide into two modules:

- **Language module:** During inference, this module's purpose is to infer the logical form of each observed sentence. That is, given the input sentence y_i, this module outputs the k most-probable values of the logical form x_i (i.e., semantic parsing).

- **Reasoning module:** During inference, this module's purpose is to infer the underlying theory that logically entails the observed logical forms (and their proofs thereof). That is, given an input collection of logical forms $\boldsymbol{x}$, this module outputs the posterior distribution of the underlying theory T and the proofs $\boldsymbol{\pi}$ of those logical forms.

Note that the y_i need not necessarily be sentences, and PWM can easily be generalized to other kinds of data. For example, if a generative model of images is available for $p(y_i \mid x_i)$, then an equivalent ''vision module'' may be defined. This module may be used either in place of, or together with, the language module. In the above generative process, PWM assumes each sentence to be independent. A model of context is required to properly handle inter-sentential anaphora or conversational settings. This can be done by allowing the distribution on y_i to depend on previous logical forms or sentences: $p(y_i \mid x_1, \ldots, x_i)$ (i.e., relaxing the i.i.d. assumption). For simplicity of this proof-of-concept, this is left to future work.

There is a vast design space for symbolic representations of meaning. We are unable to comprehensively list all of our design choices, but we describe two important ones below.

Neo-Davidsonian semantics (Parsons, 1990) is used to represent meaning in all logical forms (both in the theory and during semantic parsing). As a concrete example, a straightforward way to represent the meaning of ''Jason traveled to New York'' could be with the logical form travel(jason, nyc). In neo-Davidsonian semantics, this would instead be represented with three distinct atoms: $\texttt{travel}(c_1)$, $\texttt{arg1}(c_1) = \texttt{jason}$, and $\texttt{arg2}(c_1) = \texttt{nyc}$. Here, c_1 is a constant that represents the ''traveling event,'' whose first argument is the constant representing Jason, and whose second argument is the constant representing New York City. This representation allows the event to be more readily modified by other logical expressions, such as in ''Jason quickly traveled to NYC before nightfall.''

In addition, PWM defers named entity linking to the reasoning module (it is not done during parsing). That is, the semantic parser does not parse ''Jason'' directly into the constant jason.

		Semantic parser output	Possible axioms in theory
	Without neo-Davidsonian semantics, language module performs entity linking	$\exists b(\texttt{book}(b) \wedge \texttt{write}(\texttt{alex}, b))$	$\texttt{book}(c_1), \texttt{write}(\texttt{alex}, c_1).$
	With neo-Davidsonian semantics, language module performs entity linking	$\exists b(\texttt{book}(b) \wedge \exists w(\texttt{write}(w)$ $\wedge\, \texttt{arg1}(w) = \texttt{alex} \wedge \texttt{arg2}(w) = b))$	$\texttt{book}(c_1), \texttt{write}(c_2),$ $\texttt{arg1}(c_2) = \texttt{alex},$ $\texttt{arg2}(c_2) = c_1.$
our approach	*With* neo-Davidsonian semantics, reasoning module resolves named entities	$\exists a(\texttt{name}(a) = \text{“Alex”}$ $\wedge\, \exists b(\texttt{book}(b) \wedge \exists w(\texttt{write}(w)$ $\wedge\, \texttt{arg1}(w) = a \wedge \texttt{arg2}(w) = b)))$	$\texttt{book}(c_1), \texttt{write}(c_2),$ $\texttt{name}(c_3) = \text{“Alex”},$ $\texttt{arg1}(c_2) = c_3,$ $\texttt{arg2}(c_2) = c_1.$

Table 1: Design choices in the representation of the meaning of ‘‘Alex wrote a book.’’

Rather, named entities are parsed into existentially quantified expressions, for example, $\exists j(\texttt{name}(j) = \text{‘‘Jason’’} \wedge \ldots)$. This simplifies the parser's task and allows reasoning to aid in entity linking. Table 1 details these design options.

3.1 Reasoning Module

3.1.1 Generative Process for the Theory *p*(T)

The theory T is a collection of axioms $a_1, a_2, \ldots$ represented in higher-order logic. We choose a fairly simple prior $p(T)$ for rapid prototyping, but it is straightforward to substitute with a more complex prior. Specifically $a_1, a_2, \ldots$ are distributed according to a distribution G_a which is sampled from a *Dirichlet process* (DP) (Ferguson, 1973), an exchangeable non-parametric distribution.

$$G_a \sim \text{DP}(H_a, \alpha), \tag{1}$$

$$a_1, a_2, \ldots \sim G_a, \tag{2}$$

where H_a is the *base distribution* and $\alpha = 0.1$ is the concentration parameter. An equivalent perspective of the DP that better illustrates how the samples are generated is the *Chinese restaurant process* (Aldous, 1985):

$$a_i \sim H_a \text{ with probability } \frac{\alpha}{\alpha + i - 1}, \tag{3}$$

$$a_i = a_j \text{ with probability } \frac{\sum_{k=1}^{i-1} \mathbb{1}\{a_k = a_j\}}{\alpha + i - 1}. \tag{4}$$

That is, the i^{th} sample is drawn from H_a with probability proportional to α, or it is set to a previous sample with probability proportional to the number of times that sample has been drawn.

The base distribution H_a recursively generates logical forms in higher-order logic. Because any formula can be written as a tree, they can be generated top-down, starting from the root. The type of each node (conjunction $\wedge$, disjunction $\vee$, negation $\neg$, quantification $\forall x$, etc.) is sampled from a categorical distribution. If the type of the node is selected to be an atom (e.g., $\texttt{book}(c_1)$), then its predicate is sampled from a non-parametric distribution of predicate symbols H_p. The atom's argument(s) are each sampled as follows: If n_V is the number of available variables (from earlier generated quantifiers), then sample a variable uniformly at random with probability $\frac{1}{n_V+1}$; otherwise, with probability $\frac{1}{n_V+1}$, sample a constant from a non-parametric distribution of constant symbols H_c. For brevity, we refer the reader to our code for the specific forms of H_p and H_c.

Since PWM uses a neo-Davidsonian representation, another node type that H_a can generate is an event argument (e.g., $\texttt{arg1}(c_1) = \texttt{jason}$). When this is selected, the event constant (c_1 in the example) is sampled in the same way an atom's argument is sampled, as described above: first by trying to sample a variable, and otherwise sampling a constant from H_c. The right side of the equality ($\texttt{jason}$ in the example) can either be a variable, constant, string, or number, so PWM first selects its type from a categorical distribution. If the type is chosen to be a number, string, or variable, its value is sampled uniformly. If the type is chosen to be a constant, it is sampled from H_c.

Names of entities are treated specially in this prior: The number of names available to each entity is sampled according to a very light-tailed distribution i.i.d.: for entity c_i the number of names $n_N(c_i) \triangleq \#\{s : \texttt{name}(c_i) = s\}$ is distributed according to $p(n_N(c_i) = k) \propto \lambda^{k^2}$. This ensures that entities tend not to have too many names.

Sets are also treated specially in this prior: One kind of axiom that can be generated is one that declares the size of a set—for example,

$\texttt{size}(\lambda x.\texttt{planet}(x)) = 8$ denotes that the size of the set of planets is 8. In the prior, the size of each set is distributed according to a geometric distribution with parameter 10^{-4}. Sets can have arity not equal to 1, in which case their elements are tuples.

Deterministic Constraints: We also impose hard constraints on the theory T. Most importantly, T is required to be *globally consistent*. While this is a conceptually simple requirement, it is computationally expensive (generally undecideable even in first-order logic). PWL enforces this constraint by keeping track of the known sets in the theory (i.e., a set is known if its set size axiom is used in a proof, or if the set appears as a subset/superset in a universally-quantified axiom, such as in $\forall x(\texttt{cat}(x) \to \texttt{mammal}(x))$ where the set $\lambda x.\texttt{cat}(x)$ is a subset of $\lambda x.\texttt{mammal}(x)$). For each set, PWL computes which elements are provably members of that set. If the number of provable members of a set is greater than its size, or if an element is both provably a member and not a member of a set, the theory is inconsistent. Relaxing this constraint would be valuable in future research, perhaps instead by only considering the *relevant* sets rather than all sets in the theory, or deferring consistency checks altogether. We place a handful of other constraints on the theory T: The name of an entity must be a string (and not a number or a constant). All constants are distinct; that is, $c_i \neq c_j$ for all $i \neq j$. This helps to alleviate identifiability issues, as otherwise, there would be a much larger number of logically equivalent theories. No event can be an argument of itself (e.g., there is no constant c_i such that $\texttt{arg1}(c_i) = c_i$). If a theory T satisfies all constraints, we write ''T valid.''

These constraints do slightly complicate computation of the prior, since the generative process for generating T is *conditioned* on T being valid:

$$p(T \mid T \text{ valid}) = p(T)/p(T \text{ valid}), \tag{5}$$

$$\text{where } p(T \text{ valid}) = \sum_{T':T' \text{ valid}} p(T'), \tag{6}$$

and the denominator is intractable to compute. However, we show in Section 3.1.3 that for inference, it suffices to be able to efficiently compute the ratio of prior probabilities:

$$\frac{p(T_1|T_1 \text{ valid})}{p(T_2|T_2 \text{ valid})} = \frac{p(T_1)p(T_2 \text{ valid})}{p(T_2)p(T_1 \text{ valid})} = \frac{p(T_1)}{p(T_2)}. \tag{7}$$

Additionally note that because the above constraints do not depend on the *order* of the axioms, constants, and so forth (i.e., the constraints themselves are exchangeable), the distribution of T conditioned on T being valid is exchangeable.

Properties of the Prior $p(T)$: We emphasize that these distributions were chosen for simplicity and ease of implementation, and they worked well enough in experiments. However, there are likely many distributions that would work just as well. The parameters in the above distributions are not learned; they were set and fixed a priori. Nevertheless, this prior does exhibit useful properties for a domain- and task-general model of reasoning:

- *Occam's razor:* Smaller/simpler theories are given higher probability than larger and more complex theories.
- *Consistency:* Inconsistent theories are discouraged or impossible.
- Entities tend to have a unique name. Our prior above encodes one direction of this prior belief: Each entity is unlikely to have many names. However, the prior does not discourage one name from referring to multiple entities.
- Entities tend to have a unique type. Note, however, that this does not discourage types provable by subsumption. For example, if the theory has the axioms $\texttt{novel}(c_1)$ and $\forall x(\texttt{novel}(x) \to \texttt{book}(x))$, even though $\texttt{book}(c_1)$ is provable, it is not an axiom in this example and the prior only applies to axioms.

3.1.2 Generative Process for Proofs $p(\pi_i \mid T)$

PWM uses *natural deduction*, a well-studied proof calculus, for the proofs (Gentzen, 1935, 1969). Pfenning (2004) provides an accessible introduction. Figure 2 illustrates a simple example of a natural deduction proof. Each horizontal line is a deduction step, with the (zero or more) formulas above the line being its *premises*, and the one formula below the line being its conclusion. Each deduction step has a label to the right of the line. For example, the ''$\wedge$I'' step denotes *conjunction introduction*: given that A and B are true, this step concludes that $A \wedge B$ is true, where A and B can be any formula. A natural deduction proof can rely on axioms (denoted by ''Ax''). We can write

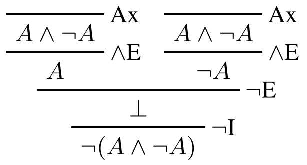

Figure 2: An example of a proof of $\neg(A \wedge \neg A)$.

any natural deduction proof π_i as a sequence of deduction steps $\pi_i \triangleq (\pi_{i,1}, \ldots, \pi_{i,k})$ by traversing the proof tree in prefix order. We define a simple generative process for π_i:

1. First sample the length of the proof k from a Poisson distribution with parameter 20.
2. For each $j = 1, \ldots, k$: Select a deduction rule from the proof calculus with a categorical distribution. If the Ax rule is selected, then simply take the next available axiom from the theory $T = a_1, a_2, \ldots$ If the deduction rule requires premises, then each premise is selected uniformly at random from $\pi_{i,1}, \ldots, \pi_{i,j-1}$.[1]

The above generative process may produce a forest rather than a single proof tree. Thus, π_i is sampled conditioned on π_i being a valid proof. Just as with $p(T)$ in equation 5, this conditioning causes $p(\pi_i \mid T)$ to be intractable to compute. However, only the ratio of the prior probability is needed for inference, which can be computed efficiently:

$$\frac{p(\pi_i \mid T, \pi_i \text{ valid})}{p(\pi_i' \mid T, \pi_i' \text{ valid})} = \frac{p(\pi_i \mid T)p(\pi_i' \text{ valid} \mid T)}{p(\pi_i' \mid T)p(\pi_i \text{ valid} \mid T)} = \frac{p(\pi_i \mid T)}{p(\pi_i' \mid T)}. \quad (8)$$

Although PWL was initially implemented assuming classical logic, it is easy to adapt PWL to use other logics, such as *intuitionistic logic*. Intuitionistic logic is identical to classical logic except that the *law of the excluded middle* $A \vee \neg A$ is not a theorem (see Figure 3 for an example where the two logics disagree). The interpretable nature of the reasoning module makes it easy to adapt it to other kinds of logic or proof calculi. PWL supports both classical and intuitionistic logic.

Context: "The switch is on. The circuit has the bell. If the circuit has the switch and the switch is on then the circuit is complete. If the circuit does not have the switch then the circuit is complete. If the circuit is complete and the circuit has the light bulb then the light bulb is glowing. If the circuit is complete and the circuit has the bell then the bell is ringing. If the circuit is complete and the circuit has the radio then the radio is playing."

Query: "The circuit is complete." **true, false, unknown?**

Figure 3: An example from the Electricity1 section in the ProofWriter dataset. Its label is unknown. Under classical logic, the query is provably true from the information in the 1st, 3rd, and 4th sentences.

3.1.3 Inference

Having described the generative process for the theory T and proofs $\boldsymbol{\pi}$, we now describe inference. Given logical forms $\boldsymbol{x}$, the goal is to compute the posterior distribution of T and $\boldsymbol{\pi}$ such that the conclusion of the each proof π_i is x_i. That is, PWL aims to recover the latent theory and proofs that explain/entail the given observed logical forms. To this end, PWL uses Metropolis-Hastings (MH) (Hastings, 1970; Robert and Casella, 2004). PWL performs inference in a streaming fashion, starting with the case $n = 1$ to obtain MH samples from $p(\pi_1, T|x_1)$. Then, for every new logical form x_n, PWL uses the last sample from $p(\pi_1, \ldots, \pi_{n-1}, T|x_1, \ldots, x_{n-1})$ as a starting point and then obtains MH samples from $p(\pi_1, \ldots, \pi_n, T|x_1, \ldots, x_n)$. This warm-start initialization serves to dramatically reduce the number of iterations needed to mix the Markov chain. To obtain the MH samples, the proof of each new logical form $\pi_n^{(0)}$ is initialized using Algorithm 1, whereas the proofs of previous logical forms are kept from the last MH sample. The axioms in these proofs constitute the theory sample $T^{(0)}$. Then, for each iteration $t = 1, \ldots, N_{\text{iter}}$, MH proposes a mutation to one or more proofs in $\boldsymbol{\pi}^{(t)}$. The possible mutations are listed in Table 2. This may change axioms in $T^{(t)}$. Let T', π_i' be the newly proposed theory and proofs. Then, compute the acceptance probability:

$$\frac{p(T')}{p(T^{(t)})} \prod_{i=1}^{n} \frac{p(\pi_i'|T')}{p(\pi_i^{(t)}|T^{(t)})} \frac{g(T^{(t)}, \boldsymbol{\pi}^{(t)}|T', \boldsymbol{\pi}')}{g(T', \boldsymbol{\pi}'|T^{(t)}, \boldsymbol{\pi}^{(t)})}, \quad (9)$$

where $g(T', \boldsymbol{\pi}'|T^{(t)}, \boldsymbol{\pi}^{(t)})$ is the probability of proposing the mutation from $T^{(t)}, \boldsymbol{\pi}^{(t)}$ to $T', \boldsymbol{\pi}'$, and $g(T^{(t)}, \boldsymbol{\pi}^{(t)}|T', \boldsymbol{\pi}')$ is the probability of the *inverse* of this mutation. Because this quantity

[1] Some deduction rules require additional parameters, and we refer the reader to our code for details on how these parameters are sampled.

depends only on the *ratio* of probabilities, it can be computed efficiently (see equations 3.1.1 and 8). Once this quantity is computed, sample from a Bernoulli with this quantity as its parameter. If it succeeds, MH accepts the proposed theory and proofs as the next sample: $T^{(t+1)} = T'$ and $\pi_i^{(t+1)} = \pi'_i$. Otherwise, reject the proposal and keep the old sample: $T^{(t+1)} = T^{(t)}$ and $\pi_i^{(t+1)} = \pi_i^{(t)}$. If every possible theory and proof is reachable from the initial theory by a sequence of mutations, then with sufficiently many iterations, the samples $T^{(t)}$ and $\pi_i^{(t)}$ will be distributed according to the true posterior $p(T, \boldsymbol{\pi}|\boldsymbol{x})$. If only a subset of possible theories and proofs are reachable from the initial theory, the MH samples will be distributed according to the true posterior *conditioned* on that subset. This may suffice for many applications, particularly if the theories in the subset have desirable properties such as better tractability. But the subset cannot be too small because then PWL would lose generality.

The function `init_proof` in Algorithm 1 recursively calls `init_disproof`. Due to space limitations, we refer the reader to our code for this function; it closely mirrors the structure of `init_proof`. The purpose of `init_proof` is to find *some* proof of a given higher-order formula, or return null if none exists. Its task is finding a satisfying abductive proof, which is easier than theorem proving, since it can create new axioms as needed. The returned proof need not be ''optimal'' because it serves as the initial state for MH, which will further refine the proof. The validity of the proofs is guaranteed by the fact that `init_proof` only returns valid proofs and the MH proposals preserve validity.

Algorithm 1: Pseudocode for proof initialization. If any new axiom violates the deterministic constraints in Section 3.1.1, the function returns null.

```
1  function init_proof(formula $A$)
2    if $A$ is a conjunction $B_1 \wedge \ldots \wedge B_n$
3      for $i = 1$ to $n$ do
4        $\phi_i$ = init_proof($B_i$)
5      return $\dfrac{\phi_1 \quad \ldots \quad \phi_n}{B_1 \wedge \ldots \wedge B_n}\wedge I$
6    else if $A$ is a disjunction $B_1 \vee \ldots \vee B_n$
7      $\mathcal{I}$ = shuffle($1, \ldots, n$)
8      for $i \in \mathcal{I}$ do
9        $\phi_i$ = init_proof($B_i$)
10       if $\phi_i \neq$ null
11         return $\dfrac{\phi_i}{B_1 \vee \ldots \vee B_n}\vee I$
12   else if $A$ is a negation $\neg B$
13     return init_disproof($B$)
14   else if $A$ is an implication $B_1 \to B_2$
15     if using classical logic
16       $\mathcal{I}$ = shuffle(1,2)
17       for $i \in \mathcal{I}$ do
18         if $i = 1$
19           $\phi_1$ = init_disproof($B_1$)
20           if $\phi_1$ = null continue
21           return $\dfrac{\dfrac{\dfrac{\phi_1 \quad \dfrac{}{B_1}Ax}{\bot}\neg E}{B_2}\bot E}{B_1 \to B_2}\to I$
22         else
23           $\phi_2$ = init_proof($B_2$)
24           if $\phi_2$ = null continue
25           return $\dfrac{\phi_2}{B_1 \to B_2}\to I$
26     else if using intuitionistic logic
27       return $\dfrac{}{B_1 \to B_2}Ax$
28   else if $A$ is an existential quantification $\exists x.f(x)$
29     let $\mathcal{C}$ be the set of known constants, numbers, and strings
         in $T$, and the new constant $c^*$
30     $\mathcal{I}$ = swap[2](shuffle($\mathcal{C}$))
31     for $c \in \mathcal{I}$ do
32       $\phi_c$ = init_proof($f(c)$)
33       if $\phi_c \neq$ null
34         return $\dfrac{\phi_c}{\exists x.f(x)}\exists I$
35   else if $A$ is a universal quantification $\forall x.f(x)$
36     return $\dfrac{}{\forall x.f(x)}Ax$
37   else if $A$ is an equality $B_1 = B_2$
38     return $\dfrac{}{B_1 = B_2}Ax$
39   else if $A$ is an atom (e.g. book(great_gatsby))
40     return $\dfrac{}{A}Ax$
41   else return null
```

3.2 Language Module

For the language module, PWM uses the probabilistic model of Saparov et al. (2017). The generative nature of their semantic parsing model allows it to fit seamlessly into PWM and PWL.

[2] `swap` randomly selects an element in its input list to swap with the first element. The probability of moving an element c to the front of the list is computed as follows: Recursively inspect the atoms in the formula $f(c)$ and count the number of ''matching'' atoms: The atoms $t(c)$ or $c(t)$ is considered ''matching'' if it is provable in T. Next, count the number of ''mismatching'' axioms: for each atom $t(c)$ in the formula $f(c)$, an axiom $t'(c)$ is ''mismatching'' if $t \neq t'$. And similarly for each atom $c(t)$ in the formula $f(c)$, an axiom $c(t')$ is ''mismatching'' if $t \neq t'$. Let n be the number of ''matching'' atoms and m be the number of ''mismatching'' axioms, then the probability of moving c to the front of the list is proportional to $\exp\{n - 2m\}$. This greatly increases the chance of finding a high-probability proof in the first iteration of the loop on line 31, and since this function is also used in an MH proposal, it dramatically improves the acceptance rate. This reduces the number of MH iterations needed to sufficiently mix the Markov chain.

Proposal	Probability of selecting proposal
Select a grounded atomic axiom (e.g., $\texttt{square}(c_1)$) and propose to replace it with an instantiation of a universal quantification (e.g., $\forall x(\texttt{rectangle}(x) \land \texttt{rhombus}(x) \to \texttt{square}(x))$), where the antecedent conjuncts are selected uniformly at random from the other grounded atomic axioms for the constant c_1: $\texttt{rectangle}(c_1)$, $\texttt{rhombus}(c_1)$, etc.	$\frac{1}{N}$
The inverse of the above proposal: select an instantiation of a universal quantification and replace it with a grounded atomic axiom.	$\frac{1}{N}$
Select an axiom that declares the size of a set (e.g., of the form $\texttt{size}(\texttt{us\_states}) = 50$), and propose to change the size of the set by sampling from the prior distribution, conditioned on the maximum and minimum consistent set size.	$\frac{1}{N}$
Select a node from a proof tree of type $\lor$I, $\to$ I, or $\exists$I.[3] These nodes were created in Algorithm 1 on lines 7, 16, and 30, respectively, where for each node, a single premise was selected out of a number of possible premises. This proposal naturally follows from the desire to explore other selections by re-sampling the proof: it simply calls $\texttt{init\_proof}$ again on the formula at this proof node.	$\frac{1}{N}$
Merge: Select a ''mergeable'' event; that is, three constants (c_i, c_j, c_k) such that $\texttt{arg1}(c_i) = c_j$, $\texttt{arg2}(c_i) = c_k$, and $t(c_i)$ for some constant t are axioms, and there also exist constants $(c_{i'}, c_{j'}, c_{k'})$ such that $i' > i$, $\texttt{arg1}(c_{i'}) = c_{j'}$, $\texttt{arg2}(c_{i'}) = c_{k'}$, and $t(c_{i'})$ are axioms. Next, propose to merge $c_{i'}$ with c_i by replacing all instances of $c_{i'}$ with c_i in the proof trees, $c_{j'}$ with c_j, and $c_{k'}$ with c_k. This proposal is not necessary in that these changes are reachable with other proposals, but those proposals may have low probability, and so this can help to more easily escape local maxima.	$\frac{\alpha}{N}$
Split: The inverse of the above proposal.	$\frac{\beta}{N}$

Table 2: A list of the Metropolis-Hastings proposals implemented in PWL thus far. N, here, is a normalization term: $N = \lvert A\rvert + \lvert U\rvert + \lvert C\rvert + \lvert P\rvert + \alpha\lvert M\rvert + \beta\lvert S\rvert$ where: A is the set of grounded atomic axioms in T (e.g., $\texttt{square}(c_1)$), U is the set of universally-quantified axioms that can be eliminated by the second proposal, C is the set of axioms that declare the size of a set (e.g., $\texttt{size}(A) = 4$), P is the set of nodes of type $\lor$I, $\to$ I, or $\exists$I[3] in the proofs $\boldsymbol{\pi}$, M is the set of ''mergeable'' events (described above), and S is the set of ''splittable'' events. In our experiments, $\alpha = 2$ and $\beta = 0.001$.

The logical forms in their model are distributed according to a *semantic prior*, which we replace with our distribution of logical forms conditioned on the theory $p(\pi_i|T)$. Their parser is probabilistic and finds the k-best logical forms that maximize $p(y_i|x_i, T)$ for a given input sentence. Combined with our reasoning module's ability to compute the probability of a logical form, the parser can resolve ambiguous interpretations of sentences by exploiting acquired knowledge. We will demonstrate the utility of this property in resolving lexical ambiguity.

However, the semantic grammar in Saparov et al. (2017) was designed for a DATALOG representation of logical forms. Thus, we designed and implemented a new grammar for our more domain-general formalism in higher-order logic. Though their model induces preterminal production rules from data (e.g., N → ''cat''), we must manually specify the nonterminal production rules (e.g., NP → ADJP NP). This allows us to encode prior knowledge of the English language into PWM, dramatically improving its statistical efficiency and obviating the need for massive training sets to learn English syntax. It is nonetheless tedious to design these rules while maintaining domain-generality. Once specified, however, these rules can be re-used in new tasks and domains with minimal or no changes. We also improved their model to generalize over inflected forms of words. In the generative process, instead of generating sentence tokens directly (e.g., ''I am sleeping''), PWM generates word roots with flags indicating their inflection (e.g., ''I be[1ST,SG] sleep[PRS,PTCP]''). During parsing, this has the effect of performing morphological and semantic parsing jointly. We extracted the necessary comprehensive morphology information from Wiktionary (Wikimedia Foundation 2020).

We train this new grammar to learn the parameters that govern the conditional distributions and the preterminal production rules. To do so, we construct a small *seed training set* consisting of 55

[3] Also disproofs of conjunctions, if using classical logic.

labeled sentences, 47 nouns, 55 adjectives, and 20 verbs.[4] We wrote and labeled these sentences by hand, largely in the domain of astronomy, with the aim to cover a diverse range of English syntactic constructions. This small training set was sufficient thanks to the statistical efficiency of PWM.

While PWL uses the same parsing algorithm of Saparov et al. (2017), we provide an easier-to-understand presentation. Given an input sentence y_i, the parser aims to find the logical form(s) x_i and derivation trees t_i that maximize the posterior probability $p(x_i, t_i | y_i, T)$. This discrete optimization is performed using *branch-and-bound* (Land and Doig, 1960): The algorithm starts by considering the set of all derivation trees and partitions it into a number of subsets (the ''branch'' step). For each subset S, the parser computes an upper bound on the log probability of any derivation in S (the ''bound'' step). Having computed the bound for each subset, the parser puts them into a priority queue, prioritized by the bound. The parser then dequeues the subset with the highest bound and repeats this process, further subdividing this set, computing the bound for each subdivision, and adding them to the queue. Eventually, the parser will dequeue a subset containing a single derivation whose log probability is at least the highest priority in the queue. This derivation is optimal. The algorithm can be continued to obtain the top-k derivations/logical forms. Because this algorithm operates over *sets* of logical forms (where each set is possibly infinite), we implemented a data structure to sparsely represent such sets of higher-order formulas, as well as algorithms to perform set operations, such as intersection and subtraction.

4 Experiments

4.1 ProofWriter

To demonstrate our implementation as a proof-of-concept, we evaluate it on two question-answering tasks. The first is the ProofWriter dataset (Tafjord et al., 2021), which itself is based on the earlier RuleTaker dataset (Clark et al., 2020). To evaluate and demonstrate the out-of-domain language understanding and reasoning ability of PWL, we use the Birds-Electricity ''open-world''[5] portion of the dataset, as the authors evaluated their method on this portion zero-shot, just as we do (i.e., the algorithm did not see any example from this portion during training). This portion of the data is subdivided into 6 sections, each with varying degrees of difficulty. An example from this dataset is shown in Figure 3. For each example, PWL reads the context and abduces a theory. Next, it parses the query sentence y_{n+1} into a logical form x_{n+1} and estimates its *unnormalized* probability:

$$p(x_{n+1} \mid \boldsymbol{x}) \propto p(x_1, \ldots, x_{n+1}), \tag{10}$$

$$= \sum_{T,\pi_i \text{ proof of } x_i} p(T) \prod_{i=1}^{n+1} p(\pi_i \mid T), \tag{11}$$

$$\approx \sum_{T^{(t)},\pi_i^{(t)} \sim T,\boldsymbol{\pi}|\boldsymbol{x}} p(T^{(t)}) \prod_{i=1}^{n+1} p(\pi_i^{(t)} \mid T^{(t)}). \tag{12}$$

Here, $\boldsymbol{x}$ are the previously read logical forms (the context). Since the quantity in equation 11 is intractable to compute, PWL approximates it by sampling from the posterior $T, \pi_1, \ldots, \pi_{n+1} \mid x_1, \ldots, x_{n+1}$ and summing over distinct samples. Although this approximation seems crude, the sum is dominated by a small number of the most probable theories and proofs, and MH is an effective way to find them, as we observe in experiments. MH is run for 400 iterations, and at every 100th iteration, PWL re-initializes the Markov chain by performing 20 ''exploratory'' MH steps (i.e., consisting of only the third and fourth proposals in Table 2 and accepting every proposal). This re-initialization is analogous to a random restart and can help to escape from local maxima. However, it may be promising to explore other approaches to compute this quantity, such as Luo et al. (2020). Once PWL has computed this probability for the query sentence, it does the same for the negation of the sentence. These unnormalized probabilities are compared, and if they are within 2000 in log probability, PWL returns the label unknown. If the first probability is sufficiently larger than the second, PWL returns true, and otherwise, returns false. The parameters in the prior were set by hand initially by choosing values that we thought were reasonable (e.g., the average length of a natural deduction proof

[4] The grammar, morphology data, code, as well as the seed training set are available in our Github repository.

[5] The dataset comes in two flavors: one that makes the closed-world assumption, and one that does not.

SECTION	*N*	ProofWriter-All	ProofWriter-Iter	PWL (classical)	(intuitionistic)
Electricity1	162	98.15	98.77	92.59	**100.00**
Electricity2	180	91.11	90.00	90.00	**100.00**
Electricity3	624	91.99	94.55	88.46	**100.00**
Electricity4	4224	91.64	99.91	94.22	**100.00**
Birds1	40	**100.00**	95.00	**100.00**	**100.00**
Birds2	40	**100.00**	95.00	**100.00**	**100.00**
Average	5270	91.99	98.82	93.43	**100.00**

Table 3: Zero-shot accuracy of PWL and baselines on the ProofWriter dataset.

for a sentence containing a simple subject noun phrase, object noun phrase, and transitive verb is around 20 steps, which is why the Poisson parameter for the proof length is set to 20). The values were tweaked as necessary by running the algorithm on toy examples during debugging. Note that the sentences ''Bill is a bird'' and ''Bill is not a bird'' can still both be true if each ''Bill'' refers to distinct entities. To avoid this, we chose an extreme value of the prior parameter such that the log prior probability of a theory with two entities having the same name is 2000 less than that of a theory where the name is unique. It is for this reason 2000 was chosen as the threshold for determining whether a query is true/false vs unknown. This prior worked well enough in our experiments, but the goal is to have a single prior work well with any task, so further work to explore which priors work better across a wider variety of tasks is welcome. We evaluated PWL using both classical and intuitionistic logic, even though the ground truth labels in the dataset were generated using *intuitionistic logic*.

Table 3 lists the zero-shot accuracy of PWL, comparing with baselines based on the T5 transformer (Raffel et al., 2020). We emphasize here that PWL is not perfectly comparable to the baseline, because they aim to demonstrate that their method can *learn* to reason. We instead aim to demonstrate that PWL's ability to parse and reason end-to-end generalizes to an out-of-domain question-answering task. The baseline is trained on other portions of the ProofWriter data, whereas PWL is trained only on its seed training set. PWL performed much better using intuitionistic logic than classical logic, as expected since the ground truth labels were generated using intuitionistic semantics. However, most real-world reasoning tasks would take the law of the excluded middle to be true, and classical logic would serve as a better default. Although the task is relatively simple, it nevertheless demonstrates the proof-of-concept and the promise of further research.

4.2 FictionalGeoQA

The sentences in the ProofWriter experiment are template-generated and have simple semantics. For the sake of evaluation more representative of real-world language, we introduce a new question-answering dataset called FictionalGeoQA.[6] To create this dataset, we took questions from GeoQuery (Zelle and Mooney, 1996), and for each question, we wrote a paragraph context containing the information necessary to answer the question. We added distractor sentences to make the task more robust against heuristics. Whenever possible, the sentences in this paragraph were taken from Simple English Wikipedia. However, some facts, such as the lengths of rivers, are not expressed in sentences in Wikipedia (they typically appear in a table on the right side of the page), so we wrote those sentences by hand: We took questions from GEOQUERY that expressed the desired fact in interrogative form (e.g., ''What is the length of `<river name>`?'') and converted them into declarative form (e.g., ''The length of `<river name>` is `<length>`.''). The resulting dataset contains 600 examples, where 67.4% of the sentences are from Simple English Wikipedia, and 90% of the examples contain at least one sentence *not* from Wikipedia. We replaced all place names with fictional ones to remove any confounding effects from pretraining. To keep the focus of the evaluation on reasoning ability, we chose to restrict the complexity of the language. In particular, each sentence is independent and can be

[6] Available at `github.com/asaparov/fictionalgeoqa`.

	all	*superlative*	*subjective concept def.*	*objective concept def.*	*lexical ambiguity*	*negation*	*large context*	*arithmetic*	*counting*	*0 subsets*	*1 subset*	*2 subsets*	*3 subsets*	*4 subsets*
N	600	210	150	170	180	102	100	20	30	85	213	187	85	30
UnifiedQA	33.8	29.5	7.3	33.5	32.8	14.7	43.0	10.0	**20.0**	41.2	47.9	27.8	8.2	**23.3**
Boxer + E	9.7	0.0	12.0	11.8	0.0	15.7	14.0	10.0	0.0	7.1	17.8	5.3	4.7	0.0
Boxer + Vampire	9.7	0.0	12.0	11.8	0.0	15.7	14.0	10.0	0.0	7.1	17.8	5.3	4.7	0.0
PWL parser + E	5.0	0.0	13.3	2.9	0.0	15.7	4.0	10.0	0.0	1.2	7.0	5.3	4.7	0.0
PWL parser + Vampire	9.0	0.0	13.3	11.2	0.0	15.7	4.0	10.0	0.0	12.9	13.6	5.3	4.7	0.0
PWL	**43.1**	**40.5**	**33.3**	33.5	**34.4**	**23.5**	**45.0**	10.0	0.0	**43.5**	**62.9**	**39.0**	**17.6**	0.0

LEGEND: **superlative** The subset of the dataset with examples that require reasoning over superlatives, i.e., ''longest river.''
subjective concept def. Subset with definitions of ''subjective'' concepts, i.e., ''Every river longer than 500 km is major.''
objective concept def. Subset with definitions of ''objective'' concepts, i.e., the population of a location is the number of people living there.
lexical ambiguity Subset with lexical ambiguity, i.e., ''has'' means different things in ''a state has a city named'' vs ''a state has an area of...''
negation Subset with examples that require reasoning with classical negation (negation-as-failure is insufficient).
large context Subset of examples where there are at least 100 sentences in the context.
arithmetic Subset with examples that require simple arithmetic. **counting** Subset with examples that require counting.
***n* subset(s)** Examples that belong to exactly *n* of the above subsets (no example is a member of more than 4 subsets).

Table 4: Zero-shot accuracy of PWL and baselines on the FictionalGeoQA dataset.

Context: "River Giffeleney is a river in Wulstershire. River Wulstershire is a river in the state of Wulstershire. River Elsuir is a river in Wulstershire. The length of River Giffeleney is 413 kilometers. The length of River Wulstershire is 830 kilometers. The length of River Elsuir is 207 kilometers. Every river that is shorter than 400 kilometers is not major."

Query: "What rivers in Wulstershire are not major?"

Figure 4: An example from FictionalGeoQA, a new fictional geography question-answering dataset that we created to evaluate reasoning in natural language understanding.

understood in isolation (e.g., no cross-sentential anaphora). The sentences *are* more complex than those in ProofWriter, having more of the complexities of real language, such as synonymy, lexical ambiguity (e.g., what is the semantics of ''has'' in ''a state has city'' vs ''a state has area''; or whether ''largest state'' refers to area or population), and syntactic ambiguity. This increased difficulty is evident in the results. This dataset is meant to evaluate out-of-domain generalizability, so we do not provide a separate training set for fine-tuning. An example is shown in Figure 4.

We compare PWL (using classical logic) with a number of baselines: (1) UnifiedQA (Khashabi et al., 2020), a QA system based on large-scale neural language models, (2) Boxer (Bos, 2015), a wide-coverage semantic parser, combined with Vampire 4.5.1 (Kovács and Voronkov, 2013), a theorem prover for full first-order logic, (3) Boxer combined with E 2.6 (Schulz et al., 2019), another theorem prover for full first-order logic, (4) the language module of PWL combined with Vampire, and (5) the language module of PWL combined with E. The results are shown in Table 4, along with a breakdown across multiple subsets of the dataset. UnifiedQA performs relatively well but fares more poorly on questions with negation and subjective concept definitions (e.g., ''Every river longer than 500km is major. . . What are the major rivers?''). Humans are easily able to understand and utilize such definitions, and the ability to do so is instrumental in learning about new concepts or words in new domains. PWL is able to fare better than UnifiedQA in examples with lexical ambiguity, as a result of the language module's ability to exploit acquired knowledge to resolve ambiguities. We find that Boxer has significantly higher coverage than PWL (100% vs 79.8%) but much lower precision. For instance, Boxer uses the semantic representation in the Parallel Meaning Bank (Abzianidze et al., 2017) which has a simpler representation of superlatives, and is thus unable to capture the correct semantics of superlatives in examples of this dataset. We also find that for most examples, Boxer produces different semantics for the question vs. the context sentences, oftentimes predicting the incorrect semantic role for the interrogative words, which leads to the theorem provers being unable to find a proof for these extra semantic roles. We also experimented with replacing our reasoning module with a theorem prover and found that for almost all examples, the search of the theorem prover would explode combinatorially. This was due to the fact that our semantic representation relies heavily on *sets*, so a number of simple set theoretic axioms are required for the theorem provers, but this quickly causes the deduction problem to become undecideable. Our reasoning module instead performs abduction, and is able to create axioms to more quickly find an initial proof, and then refine that proof using MH. Despite our attempt to maximize the generalizability of the grammar in PWL, there are a number of linguistic phenomena that we did not yet implement, such as interrogative subordinate clauses, wh-movement, spelling or grammatical mistakes, and so forth, and this led to the lower coverage on this dataset. Work remains to be done to implement these missing production rules in order to further increase the coverage of the parser.

5 Conclusions and Future Work

We introduced PWM, a fully symbolic Bayesian model of semantic parsing and reasoning, which we hope serves as a compelling first step in a research program toward more domain- and task-general NLU. We derived PWL, an efficient inference algorithm that reads sentences by parsing and abducing updates to its latent world model that capture the semantics of those sentences, and empirically demonstrated its ability to generalize to two out-of-domain question-answering tasks. To do so, we created a new question-answering dataset, FictionalGeoQA, designed specifically to evaluate reasoning ability while capturing more of the complexities of real language and being robust against heuristic strategies. PWL is able to read and understand sentences with richer semantics, such as definitions of new concepts. In contrast with past deductive reasoning approaches, PWL performs abduction, which is computationally easier. The highly underspecified nature of the problem of abduction is alleviated by the probabilistic nature of PWL, as it gives a principled way to find the most probable theories. We present an inference strategy where Metropolis-Hastings (MH) is performed on each sentence, in sequence, where the previous sample of the theory and proofs provides a warm-start for inference of the next sentence, reducing the number of MH iterations.

There are many avenues for future work: A simple prior was used for proofs $p(\pi_i|T)$, and an alternative is to use a compositional exchangeable prior such as adaptor grammars (Johnson et al., 2006).

The first MH proposal in Table 2 is simple but restrictive: The antecedent conjuncts and the consequent are restricted to be atomic. MH would be able to explore a much larger and semantically richer set of theories if the antecedent or consequent could contain more complex formulas, including quantified formulas. In addition, the inference algorithm sometimes becomes stuck in local maxima. One way to improve the efficiency of inference is to add a new MH proposal that specifically proposes to split or merge types. For example, if the theory has the axioms $\texttt{cat}(c_1)$ and $\texttt{dog}(c_1)$, this proposal would split c_1 into two concepts: $\texttt{cat}(c_1)$ and $\texttt{dog}(c_2)$. This kind of type-based Markov chain Monte Carlo is similar in principle to Liang et al. (2010).

As mentioned earlier, a model of context is necessary in the language module to properly handle cross-sentential anaphora and conversational contexts. Real language very rarely consists of sentences that are independent of context. There are also many research questions on the issue of *scalability*. Although PWL is able to scale to examples in FictionalGeoQA with more than 100 sentences, there are two main bottlenecks currently preventing it from scaling to significantly larger theories: (1) the maintenance of global consistency, and (2) the unfocused nature of the current MH proposals. When checking for consistency of a new axiom, rather than considering all other axioms/sets in the theory, it would be preferable to only consider the portion of the theory relevant to the new axiom. Additionally, the current MH proposals do not take into account the goal of reasoning. For example, if the current task is to answer a question about geography, then MH proposals for proofs unrelated to geography are wasteful, and would increase the number of MH steps needed. A more clever goal-aware approach for selecting proofs to mutate would help to alleviate this problem and improve scalability. PWM provides a path to incorporate information from additional modalities in principled fashion: for example by adding a generative model of images, which would serve as a separate ‘‘vision module.'' In addition, even though PWL is fully-symbolic, non-symbolic methods could be used for expressive prior/proposal distributions or approximate inference. There are many fascinating research paths to pursue from here.

Acknowledgments

We thank the anonymous reviewers and the Action Editor for their invaluable feedback. We also thank Peter Clark, William W. Cohen, Rik van Noord, Johan Bos, and Emmanouil A. Platanios for their insightful and helpful discussion. This research was funded in part by the Air Force Office of Scientific Research under AFOSR grant FA95502010118.

References

Lasha Abzianidze, Johannes Bjerva, Kilian Evang, Hessel Haagsma, Rik van Noord, Pierre Ludmann, Duc-Duy Nguyen, and Johan Bos. 2017. The parallel meaning bank: Towards a multilingual corpus of translations annotated with compositional meaning representations. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017, Valencia, Spain, April 3-7, 2017, Volume 2: Short Papers*, pages 242–247. Association for Computational Linguistics. https://doi.org/10.18653/v1/E17-2039

David J. Aldous. 1985. Exchangeability and related topics. In *Lecture Notes in Mathematics*, pages 1–198, Springer Berlin Heidelberg. https://doi.org/10.1007/BFb0099421

Erik Arakelyan, Daniel Daza, Pasquale Minervini, and Michael Cochez. 2021. Complex query answering with neural link predictors. In *International Conference on Learning Representations*.

Elena Bellodi and Fabrizio Riguzzi. 2015. Structure learning of probabilistic logic programs by searching the clause space. *Theory and Practice of Logic Programming*, 15(2):169–212. https://doi.org/10.1017/S1471068413000689

Emily M. Bender and Alexander Koller. 2020. Climbing towards NLU: On meaning, form, and understanding in the age of data. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 5185–5198. Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.acl-main.463

Chandra Bhagavatula, Ronan Le Bras, Chaitanya Malaviya, Keisuke Sakaguchi, Ari Holtzman, Hannah Rashkin, Doug Downey, Wen-tau Yih, and Yejin Choi. 2020. Abductive commonsense reasoning. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26–30, 2020*.

Johan Bos. 2015. Open-domain semantic parsing with boxer. In *Proceedings of the 20th Nordic Conference of Computational Linguistics, NODALIDA 2015, Institute of the Lithuanian Language, Vilnius, Lithuania, May 11-13, 2015*, volume 109 of *Linköping Electronic Conference Proceedings*, pages 301–304. Linköping University Electronic Press / Association for Computational Linguistics.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. *CoRR*, abs/2005.14165v4.

Eugene Charniak and Robert P. Goldman. 1993. A Bayesian model of plan recognition. *Artificial Intelligence*, 64(1):53–79.

Alonzo Church. 1940. A formulation of the simple theory of types. *Journal of Symbolic Logic*, 5(2):56–68.

Peter Clark, Oyvind Tafjord, and Kyle Richardson. 2020. Transformers as soft reasoners over language. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020*, pages 3882–3890. ijcai.org.

Robin Cooper, Simon Dobnik, Shalom Lappin, and Staffan Larsson. 2015. Probabilistic type theory and natural language semantics. In *Linguistic Issues in Language Technology, Volume 10, 2015*. CSLI Publications. https://doi.org/10.33011/lilt.v10i.1357

Andrew Cropper and Rolf Morel. 2021. Learning programs by learning from failures. *Machine Learning*, 110(4):801–856. https://doi.org/10.1007/s10994-020-05934-z

James Cussens. 2001. Parameter estimation in stochastic logic programs. *Machine Learning*, 44(3):245–271. https://doi.org/10.1023/A:1010924021315

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics.

D. R. Dowty. 1981. *Introduction to Montague Semantics*, 1st ed. Studies in Linguistics and Philosophy, 11. Springer Netherlands, Dordrecht. https://doi.org/10.1007/978-94-009-9065-4_1

H. L. Dreyfus. 1985. From micro-worlds to knowledge representation: AI at an impasse. In R. J. Brachman and H. J. Levesque, editors, *Readings in Knowledge Representation*, pages 71–93. Kaufmann, Los Altos, CA.

Jesse Dunietz, Gregory Burnham, Akash Bharadwaj, Owen Rambow, Jennifer Chu-Carroll, and David A. Ferrucci. 2020. To test machine comprehension, start by defining comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5–10, 2020*, pages 7839–7859. Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.acl-main.701

Thomas S. Ferguson. 1973. A Bayesian analysis of some nonparametric problems. *The Annals of Statistics*, 1(2):209–230. https://doi.org/10.1214/aos/1176342360.

Ulrich Furbach, Andrew S. Gordon, and Claudia Schon. 2015. Tackling benchmark problems of commonsense reasoning. In *Proceedings of the Workshop on Bridging the Gap between Human and Automated Reasoning - A workshop of the 25th International Conference on Automated Deduction (CADE-25), Berlin, Germany, August 1, 2015*, volume 1412 of *CEUR Workshop Proceedings*, pages 47–59. CEUR-WS.org.

Matt Gardner, Jonathan Berant, Hannaneh Hajishirzi, Alon Talmor, and Sewon Min. 2019. On making reading comprehension more comprehensive. In *Proceedings of the 2nd Workshop on Machine Reading for Question Answering, MRQA@EMNLP 2019, Hong Kong, China, November 4, 2019*, pages 105–112. Association for Computational Linguistics. https://doi.org/10.18653/v1/D19-5815

G. Gentzen. 1935. Untersuchungen über das logische schließen i. *Mathematische Zeitschrift*, 39:176–210. https://doi.org/10.1007/BF01201363

G. Gentzen. 1969. Investigations into Logical Deduction. In M. E. Szabo, editor, *The Collected Papers of Gerhard Gentzen*, pages 68–213. North-Holland, Amsterdam. https://doi.org/10.1016/S0049-237X(08)70822-X

Howard Gregory. 2015. *Language and Logics: An Introduction to the Logical Foundations of Language*. Edinburgh University Press.

W. K. Hastings. 1970. Monte Carlo sampling methods using markov chains and their applications. *Biometrika*, 57(1):97–109. https://doi.org/10.1093/biomet/57.1.97

Leon Henkin. 1950. Completeness in the theory of types. *Journal of Symbolic Logic*, 15(2):81–91. https://doi.org/10.2307/2266967

Jerry R. Hobbs. 2006. Abduction in Natural Language Understanding, chapter 32. John Wiley & Sons, Ltd.

Jerry R. Hobbs, Mark E. Stickel, Douglas E. Appelt, and Paul A. Martin. 1993. Interpretation as abduction. *Artificial Intelligence*, 63(1–2):69–142. https://doi.org/10.1016/0004-3702(93)90015-4

Aidan Hogan, Eva Blomqvist, Michael Cochez, Claudia d'Amato, Gerard de Melo, Claudio Gutiérrez, Sabrina Kirrane, José Emilio Labra Gayo, Roberto Navigli, Sebastian Neumaier, Axel-Cyrille Ngonga Ngomo, Axel Polleres, Sabbir M. Rashid, Anisa Rula, Lukas Schmelzeisen, Juan F. Sequeda, Steffen Staab, and Antoine Zimmermann. 2021. Knowledge graphs. *ACM Computing Surveys*, 54(4):71:1–71:37. https://doi.org/10.1145/3447772

Arcchit Jain, Tal Friedman, Ondrej Kuzelka, Guy Van den Broeck, and Luc De Raedt. 2019. Scalable rule learning in probabilistic knowledge bases. In *1st Conference on Automated Knowledge Base Construction, AKBC 2019, Amherst, MA, USA, May 20–22, 2019*.

Mark Johnson, Thomas L. Griffiths, and Sharon Goldwater. 2006. Adaptor grammars: A framework for specifying compositional nonparametric bayesian models. In *Advances in Neural Information Processing Systems 19, Proceedings of the Twentieth Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 4–7, 2006*, pages 641–648. MIT Press.

Daniel Khashabi, Sewon Min, Tushar Khot, Ashish Sabharwal, Oyvind Tafjord, Peter Clark, and Hannaneh Hajishirzi. 2020. Unifiedqa: Crossing format boundaries with a single QA system. Trevor Cohn, Yulan He, and Yang Liu, editors, In *Findings of the Association for Computational Linguistics: EMNLP 2020, Online Event, 16-20 November 2020*, volume EMNLP 2020 of *Findings of ACL*, pages 1896–1907. Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.findings-emnlp.171

Iuliia Kotseruba and John K. Tsotsos. 2020. 40 years of cognitive architectures: Core cognitive abilities and practical applications. *Artificial Intelligence Review*, 53(1):17–94. https://doi.org/10.1007/s10462-018-018-9646-y

Laura Kovács and Andrei Voronkov. 2013. First-order theorem proving and vampire. In *Computer Aided Verification - 25th International Conference, CAV 2013, Saint Petersburg, Russia, July 13-19, 2013. Proceedings*, volume 8044 of *Lecture Notes in Computer Science*, pages 1–35. Springer.

John E. Laird, Allen Newell, and Paul S. Rosenbloom. 1987. SOAR: An architecture for general intelligence. *Artificial Intelligence*, 33(1):1–64.

Brenden M. Lake, Tomer D. Ullman, Joshua B. Tenenbaum, and Samuel J. Gershman. 2016. Building machines that learn and think like people. *CoRR*, abs/1604.00289v3. https://doi.org/10.1016/0004-3702(87)90050-6

A. H. Land and A. G. Doig. 1960. An automatic method of solving discrete programming problems. *Econometrica*, 28(3):497. https://doi.org/10.2307/1910129

Percy Liang, Michael I. Jordan, and Dan Klein. 2010. Type-based MCMC. In *Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics, Proceedings, June 2-4, 2010, Los Angeles, California, USA*, pages 573–581. Association for Computational Linguistics.

Tal Linzen. 2020. How can we accelerate progress towards human-like linguistic generalization? In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 5210–5217. Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.acl-main.465

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692v1.

Yucen Luo, Alex Beatson, Mohammad Norouzi, Jun Zhu, David Duvenaud, Ryan P. Adams, and Ricky T. Q. Chen. 2020. SUMO: Unbiased estimation of log marginal probability for latent variable models. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26–30, 2020*. OpenReview.net.

Tom M. Mitchell, William W. Cohen, Estevam R. Hruschka Jr., Partha P. Talukdar, Bo Yang, Justin Betteridge, Andrew Carlson, Bhavana Dalvi Mishra, Matt Gardner, Bryan Kisiel, Jayant Krishnamurthy, Ni Lao, Kathryn Mazaitis, Thahir Mohamed, Ndapandula Nakashole, Emmanouil A. Platanios, Alan Ritter, Mehdi Samadi, Burr Settles, Richard C. Wang, Derry Wijaya, Abhinav Gupta, Xinlei Chen, Abulhair Saparov, Malcolm Greaves, and Joel Welling. 2018. Never-ending learning. *Communications of ACM*, 61(5):103–115. https://doi.org/10.1145/3191513

Stephen Muggleton. 1991. Inductive logic programming. *New Generation Computing*, 8(4):295–318. https://doi.org/10.1007/BF03037089

Stephen Muggleton. 1996. Stochastic logic programs. In *Advances in Inductive Logic Programming*, pages 254–264.

Allen Newell and Herbert A. Simon. 1976. Computer science as empirical inquiry: Symbols and search. *Commun. ACM*, 19(3):113–126. https://doi.org/10.1145/360018.360022

Mathias Niepert and Pedro M. Domingos. 2015. Learning and inference in tractable probabilistic knowledge bases. In *Proceedings of the Thirty-First Conference on Uncertainty in Artificial Intelligence, UAI 2015, July 12-16, 2015, Amsterdam, The Netherlands*, pages 632–641. AUAI Press.

Mathias Niepert, Christian Meilicke, and Heiner Stuckenschmidt. 2012. Towards distributed MCMC inference in probabilistic knowledge bases. In *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction, AKBC-WEKEX@NAACL-HLT 2012, Montrèal, Canada, June 7-8, 2012*, pages 1–6. Association for Computational Linguistics.

Terence Parsons. 1990. *Events in the Semantics of English*. MIT Press, Cambridge, MA.

Frank Pfenning. 2004. Natural deduction. Lecture notes in 15-815 Automated Theorem Proving.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21:140:1–140:67.

Hongyu Ren, Weihua Hu, and Jure Leskovec. 2020. Query2box: Reasoning over knowledge graphs in vector space using box embeddings. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Christian P. Robert and George Casella. 2004. *Monte Carlo Statistical Methods*. Springer Texts in Statistics, Springer. https://doi.org/10.1007/978-1-4757-4145-2

Tim Rocktäschel and Sebastian Riedel. 2017. End-to-end differentiable proving. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 3788–3800.

Stuart J. Russell and Peter Norvig. 2010. *Artificial Intelligence - A Modern Approach, Third International Edition*. Pearson Education.

Swarnadeep Saha, Sayan Ghosh, Shashank Srivastava, and Mohit Bansal. 2020. Prover: Proof generation for interpretable reasoning over rules. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 122–136. Association for Computational Linguistics.

Abulhair Saparov, Vijay A. Saraswat, and Tom M. Mitchell. 2017. A probabilistic generative grammar for semantic parsing. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017), Vancouver, Canada, August 3–4, 2017*, pages 248–259. Association for Computational Linguistics. https://doi.org/10.18653/v1/K17-1026

Taisuke Sato, Yoshitaka Kameya, and Neng-Fa Zhou. 2005. Generative modeling with failure in PRISM. In *IJCAI-05, Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence, Edinburgh, Scotland,* *UK, July 30 - August 5, 2005*, pages 847–852. Professional Book Center,

Stephan Schulz, Simon Cruanes, and Petar Vukmirovic. 2019. Faster, higher, stronger: E 2.3. In *Automated Deduction - CADE 27 – 27th International Conference on Automated Deduction, Natal, Brazil, August 27-30, 2019, Proceedings*, volume 11716 of *Lecture Notes in Computer Science*, pages 495–507. Springer. https://doi.org/10.1007/978-3-030-29436-6_29

Haitian Sun, Andrew O. Arnold, Tania Bedrax-Weiss, Fernando Pereira, and William W. Cohen. 2020. Faithful embeddings for knowledge base queries. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Oyvind Tafjord, Bhavana Dalvi, and Peter Clark. 2021. Proofwriter: Generating implications, proofs, and abductive statements over natural language. In *Findings of the Association for Computational Linguistics: ACL/IJCNLP 2021, Online Event, August 1–6, 2021*, volume ACL/IJCNLP 2021 of *Findings of ACL*, pages 3621–3634. Association for Computational Linguistics. https://doi.org/10.18653/v1/2021.findings-acl.317

Ronen Tamari, Chen Shani, Tom Hope, Miriam R. L. Petruck, Omri Abend, and Dafna Shahaf. 2020. Language (re)modelling: Towards embodied language understanding. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 6268–6281. Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.acl-main.559

Wikimedia Foundation. 2020. Wiktionary data dumps.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 5754–5764.

Kexin Yi, Chuang Gan, Yunzhu Li, Pushmeet Kohli, Jiajun Wu, Antonio Torralba, and Joshua B. Tenenbaum. 2020. CLEVRER: Collision events for video representation and reasoning. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*.

John M. Zelle and Raymond J. Mooney. 1996. Learning to parse database queries using inductive logic programming. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence and Eighth Innovative Applications of Artificial Intelligence Conference, AAAI 96, IAAI 96, Portland, Oregon, USA, August 4–8, 1996, Volume 2*, pages 1050–1055. AAAI Press / The MIT Press.