Few-Shot Novel Concept Learning for Semantic Parsing

Soham Dan and **Osbert Bastani** and **Dan Roth**
{sohamdan,obastani,danroth}@seas.upenn.edu
University of Pennsylvania

Abstract

Humans are capable of learning novel concepts from very few examples; in contrast, state-of-the-art machine learning algorithms typically need thousands of examples to do so. In this paper, we propose an algorithm for learning novel concepts by representing them as programs over existing concepts. This way the concept learning problem is naturally a program synthesis problem and our algorithm learns from a few examples to synthesize a program representing the novel concept. In addition, we perform a theoretical analysis of our approach for the case where the program defining the novel concept over existing one is context-free. We show that given a learned grammar-based parser and a novel production rule, we can augment the parser with the production rule in a way that provably generalizes. We evaluate our approach by learning concepts in the semantic parsing domain extended to the few-shot novel concept learning setting, showing that our approach significantly outperforms end-to-end neural semantic parsers.

1 Introduction

A key feature of human intelligence is few-shot learning—namely, their ability to learn novel concepts from as few as one or two examples. In contrast, current deep learning approaches face significant challenges with such tasks. This is a limitation of deep learning across many applications, since in many domains, there are a large number of concepts, and for many, there are only a few examples available for learning.

There has been substantial recent interest in studying few-shot learning in the context of semantic parsing (Lake, 2019; Lake and Baroni, 2018; Loula et al., 2018), which is the task of mapping natural language utterances to an executable meaning representation (Mooney, 2007). This setting provides a rich opportunity for concept learning since concepts can naturally be grounded in symbolic representations in the form of programs. As a consequence, there is an opportunity to learn novel, unseen concepts in a compositional way: the novel concept can be represented as a composition of existing concepts, and can then be composed with existing concepts to form more complex ones. While this compositional structure exists in many natural language tasks, it is explicit in semantic parsing.

As a concrete example, consider the novel concept *4 times* in the utterance *run 4 times*, where *run* is an existing concept. The program representing *4 times* is the program λx . (REPEAT 4 x), which is composed with the program RUN representing *run* to form the program (REPEAT 4 RUN).

In this paper, we propose a novel algorithm for learning novel concepts for semantic parsing. At a high level, leveraging the fact that concepts can be represented as programs, our algorithm is based on *program synthesis* (also known as *program induction*). In particular, given a set of input-output examples, synthesis algorithms search over the space of possible programs to identify one that is consistent with these examples; they can often find the correct program from very few examples.

The key challenge in applying program synthesis to our setting is that the given training examples are for the whole semantic parsing task rather than for the specific sub-program corresponding to the novel concept. In more detail, we consider the problem of learning a semantic parser from denotations alone—i.e., each training example consists of an utterance (the user's input) labeled with the execution of the corresponding program (the user's desired output) rather than the program itself. We assume that the utterance contains a *single* new natural language concept (e.g., a word or phrase) that we are trying to learn (i.e., synthesize a program representing that word or phrase). To address this issue, our algorithm proceeds as follows:

http://cogcomp.org/page/publication_view/952

Findings of the Association for Computational Linguistics: EMNLP 2021, pages 2064–2075
November 7–11, 2021. ©2021 Association for Computational Linguistics

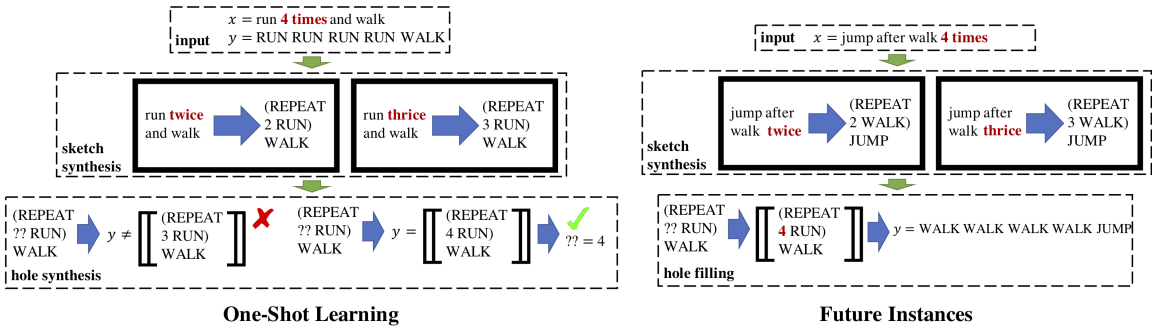

Figure 1: A pictorial overview of our proposed algorithm for concept learning. Using a single teaching example, our semantic parser is able to learn the *4 times* concept and use it in unseen contexts.

- **Sketch synthesis:** Since we already know the remaining concepts in the utterance, we want to avoid synthesizing them as well. To this end, our algorithm first synthesizes a *sketch* (Solar-Lezama, 2008), which is an incomplete program where one of its expressions has been left as a *hole* that remains to be filled in. This hole is supposed to be filled by the novel concept we are trying to synthesize.
- **Hole synthesis:** Next, our algorithm enumerates through possible sub-programs to use to fill the hole, with the goal of identifying one such that the *entire* program evaluates to the user's desired output. The resulting sub-program is our synthesized representation of the novel concept.

Finally, whenever the novel concept is encountered in future examples, we can substitute it with the synthesized sub-program.

Next, we perform a theoretical analysis of our approach, which is designed to more generally elucidate why an approach such as ours can enable few-shot learning of novel concepts. In our analysis, we assume that a representation of the novel concept has already been synthesized; instead, our goal is to illustrate why augmenting the existing model with this new concept can generalize well. In particular, the main issue is that the novel concept can result in a shift in the distribution of decisions that must be learned by the model (e.g., application of parsing rules). We focus on the problem of parsing context-free grammars, which is simpler since it is a classification problem instead of a structured prediction problem. Then, we show that assuming the learned model is grammar-based (i.e., learn which production rules are in the grammar), then augmenting it with the novel concept (i.e., a novel production rule) can generalize well.

Finally, we experimentally evaluate our approach on two semantic parsing benchmarks, SCAN (Lake and Baroni, 2018) and GeoQuery (Zelle, 1996), extended to our problem of few-shot novel concept learning. We demonstrate that while end-to-end deep learning baselines fail to learn the novel concepts, our approach does so effectively.

In summary, our key contributions are:

- We propose a novel algorithm, Substitution-Driven Concept Learning (SDCL), for synthesizing programmatic representations of novel concepts in the context of semantic parsing.
- We prove generalization bounds on our approach adapted to context-free parsing; the key challenge is bounding the distribution shift induced by adding the novel concept.
- We empirically validate our approach on the extended SCAN and GeoQuery datasets, showing that SDCL substantially outperforms end-to-end deep learning approaches.

Example. Consider Fig. 1. The user provides a single example of the novel concept *4 times*—i.e., an utterance *run 4 times and walk* and its denotation RUN RUN RUN RUN WALK (but *not* the program (REPEAT 4 RUN) WALK). First, we infer the type of *4 times* by substituting it with other concepts; in particular, when *4 times* is substituted with *twice* or *thrice*, we observe that the resulting sentence is grammatical. For these substitutions, the semantic parser produces programs (REPEAT 2 RUN) WALK and (REPEAT 3 RUN) WALK, respectively. Then, we compute the difference of these programs to derive the sketch (REPEAT ?? RUN) WALK. Finally, we enumerate implementations of ??; filling ?? with 4 produces (REPEAT 4 RUN) WALK, whose denotation is RUN RUN RUN RUN WALK, as

desired. In the future, given an unlabeled utterance *jump after walk 4 times*, we produce sketch (REPEAT ?? WALK) JUMP, substitute ?? with 4, and return WALK WALK WALK WALK JUMP.

2 Related Work

Semantic parsing. Traditional semantic parsers consist of a grammar, which defines the space of derivations from the input utterances to output logical forms, along with a model, which ranks derivations in the grammar in order of likelihood (Kwiatkowksi et al., 2010; Kate and Mooney, 2006; Artzi and Zettlemoyer, 2013). More recently, deep learning approaches have treated semantic parsing as a sequence-to-sequence problem directly mapping the input to the output without use of a grammar (Dong and Lapata, 2016; Jia and Liang, 2016); this formulation provides a high degree of flexibility since it does not require designing a grammar for every new domain. One promising recent approach is to have the neural network decode a sequence of grammar rules to produce the output sequence (Krishnamurthy et al., 2017; Dasigi et al., 2019; Yin and Neubig, 2018); this strategy only requires a grammar over logical forms (which is typically available), not one over utterances. In this work, we investigate an important shortcoming of deep learning approaches—namely, few-shot learning of novel concepts. We provide an algorithm that enables learning novel concepts from a few examples in the context of deep learning approaches.

Systematicity in deep learning. Prior work has demonstrated that neural networks to not possess *systematicity* (Fodor and Pylyshyn, 1988; Lake and Baroni, 2018; Lake et al., 2019; Loula et al., 2018), a property where the capacity to learn certain concepts implies the capacity to learn novel, structurally-related concepts. Lake and Baroni (2018) empirically investigates novel concept learning on the SCAN dataset. They consider models that have not seen the concept *jump* during training, but only in isolation, where *jump* is represented by the program JUMP, whose denotation is also JUMP. At test time, the models have to correctly predict the output sequence for *jump* in various contexts, e.g., *jump and look*. Models are expected to be able to do so since they have seen *walk and look* and *run and look* during training, and are expected to extrapolate these examples to *jump*. They find that several sequence-to-sequence models perform very poorly on this task. However, the kind of concepts they use to assess generalization are very limited. For instance, in SCAN, the *jump* primitive is an independent concept with no relation to the existing concepts such as *walk*, *look*, *run*, *after*, *twice*, *around*. In contrast, we consider a more general notion of a novel concept as a program that may be composed of existing concepts, and compare the ability of different models as well as our approach to learn such concepts from few examples.

Data augmentation. Recently, there have been several approaches attempting to improve systematicity of deep neural networks by using data augmentation. One approach, called data recombination (Jia and Liang, 2016), is to substitute concepts with other words of the same type. However, their approach assumes the type of the concept word is known, whereas we do not make this assumption. Furthermore, their approach is restricted to shallow concepts similar to the *jump* concept in (Lake and Baroni, 2018), and does not extend to higher-order concepts. Another approach is "Good Enough Compositional Data Augmentation" (Andreas, 2020), which uses overlap with other concepts of the same type in the training data to perform data augmentation. In our setting, because we only provide a single teaching example, there is no context overlap with other concepts of the same type in the training data, so their approach is unable to produce any new examples; thus, they perform the same as the end-to-end approach.

3 Substitution-Driven Concept Learning

In this section, we describe our neurosymbolic algorithm for synthesizing programmatic representations of novel concepts from few examples.

3.1 Problem Formulation

We assume that we have already have a trained semantics parser $f_\theta : \Sigma^* \to \Pi$, which maps utterances $x \in X = \Sigma^*$ to programs $\pi = f_\theta(x)$, along with denotational semantics $[\![\cdot]\!] : \Pi \to Y$ that maps programs π to denotations $y = [\![\pi]\!]$.

Now, we consider given a novel concept c, which is a word (or phrase) that does not occur in the data used to train f_θ. In particular, we assume given a utterance $x \in x$ such that $x = x_0\, c\, x_1$, but f_θ cannot be used to parse x. In addition, we assume the user provides the desired denotation $y \in Y$—i.e., $y = [\![\pi]\!]$, where $\pi = f^*(x)$ is the desired program (which we are not given). Then, our goal

Algorithm 1 Substitution-Driven Concept Learning (SDCL) Algorithm

procedure LEARNCONCEPT(Semantic parser f_θ, Known concepts C_{train}, Example (x, y) with novel concept c)
  $\hat{\pi} \leftarrow \text{SKETCHSYNTH}(f_\theta, C_{\text{train}}, x)$
  $h \leftarrow \text{HOLESYNTH}(x, y, \hat{\pi})$
  return h
end procedure

procedure LEVERAGECONCEPT(Semantic parser f_θ, Known concepts C_{train}, Utterance x with learned concept c, Program h_c for c)
  $\pi_0 \,??\, \pi_1 \leftarrow \text{SKETCHSYNTH}(f_\theta, C_{\text{train}}, x)$
  return $[\![\pi_0\, h_c\, \pi_1]\!]$
end procedure

procedure SKETCHSYNTH($f_\theta, C_{\text{train}}, x$)
  $x_0\, c\, x_1 \leftarrow x$
  $E \leftarrow \varnothing$
  for $c' \in C_{\text{train}}$ s.t. $\text{TYPE}(c') = \text{TYPE}(c)$ **do**
    $\pi_{c'} \leftarrow f_\theta(x_0\, c'\, x_1)$
    $E \leftarrow E \cup \{(c', \pi_{c'})\}$
  end for
  Compute (π_0, π_1) s.t. for all $(c', \pi_{c'}) \in E$,
    $\pi_{c'} = \pi_0\, h_{c'}\, \pi_1$ for some $h_{c'} \in \Pi_{\text{TYPE}(c')}$
  return $\pi_0 \,??\, \pi_1$
end procedure

procedure HOLESYNTH($x, y, \hat{\pi}$)
  $x_0\, c\, x_1 \leftarrow x$
  $\pi_0 \,??\, \pi_1 \leftarrow \hat{\pi}$
  for $h \in \Pi_{\text{TYPE}(c)}$ **do**
    $\pi \leftarrow \pi_0\, h\, \pi_1$
    if $[\![\pi]\!] = y$ **then**
      return h
    end if
  end for
end procedure

is to infer the program π representing novel concept c from the given example (x, y).

We assume access to two additional pieces of information. First, we assume we have a repository of other concepts $c' \in C_{\text{train}}$, such that f_θ can correctly parse utterances composed of these concepts. More importantly, we also assume we know the *type* of each concept c' as well as the given concept c; we describe a heuristic for inferring the type of c below. In particular, the type of a concept c is the type of the value $[\![f^*(c)]\!]$. This information is used by our algorithm to substitute c in x with other concepts $c' \in C_{\text{train}}$ of the same type as c; as described below, these substitutions are needed to help construct a sketch for x.

3.2 Overall Algorithm

Our algorithm, which we call Substitution-Driven Concept Learning (SDCL), is summarized in Algorithm 1. As can be seen, it is divided into two steps: sketch synthesis (the SKETCHSYNTH subroutine) and hole synthesis (the HOLESYNTH subroutine).

Sketch synthesis. At a high level, the sketch synthesis subroutine computes a *sketch* $\hat{\pi}$, which is an incomplete program $\hat{\pi} = \pi_0 \,??\, \pi_1$ with a hole represented by the symbol $??$. We use $\hat{\pi}$ to denote incomplete programs, and π to denote complete programs. For simplicity, we represent programs (and partial programs) as sequences, but in our implementation, we represent them as trees.

To synthesize a sketch, this subroutine first substitutes the novel concept c in $x = x_0\, c\, x_1$ with each known concept $c' \in C_{\text{train}}$ that have the same type as c, to obtain a modified utterance $x' = x_0\, c'\, x_1$. Since the type of c' is the same, the resulting utterance x' is valid, and furthermore, all concepts in x' are known; thus, we can run the learned semantic parser on x' to obtain a program $\pi_{c'} = f_\theta(\pi_c)$. By compositionality, the concept c' should parse to a program $h_{c'}$ such that the overall program $\pi_{c'}$ omitting $h_{c'}$ is independent of c'—i.e.,

$$\pi_{c'} = \pi_0\, h_{c'}\, \pi_1$$

for some π_0, π_1. Here, $h_{c'}$ is a complete program; we have used h instead of π to indicate that it is the representative of concept c'. Intuitively, the pair (π_0, π_1) represent the portion of the program corresponding to the context (x_0, x_1) of c' in x'.

Finally, this subroutine returns the sketch $\hat{\pi} = \pi_0 \,??\, \pi_1$ for the original utterance x; in particular, the hole $??$ represents the missing portion of the program that is supposed to be filled with the programmatic representation of concept c.

Hole synthesis. Next, our algorithm searches over possible programs h that can be used to fill the hole in $\hat{\pi} = \pi_0 \,??\, \pi_1$. In particular, it enumerates programs h of the same type as c, constructs the complete program $\pi = \pi_0\, h\, \pi_1$, executes π to obtain $y' = [\![\pi]\!]$, and checks if $y = y'$, where y is the desired result provided by the user. If so, then it returns h; otherwise, it continues its search.

The search order over programs h is important, since multiple programs might evaluate to the desired denotation y. A typical strategy is to search for the smallest program h; intuitively, this choice serves as regularization, since smaller programs correspond to simpler functionalities. In this case, we enumerate h using breadth first search (assuming the possibilities are represented by a context-free grammar over programs), which ensures that we identify the smallest one.

Leveraging learned concepts. Finally, given a new utterance x with novel concept c, we parse it

as follows. First, we synthesize a sketch π_0 ?? π_1 for x as before. Then, we fill the hole ?? with h_c, where h_c is the program we synthesized that represents c. Finally, we obtain $[\![\pi_0\, h_c\, \pi_1]\!]$.

3.3 Algorithm Details

Multiple examples. We have described our approach for using a single example (x, y) to learn the novel concept; it can easily be extended to multiple examples $(x_1, y_1), ..., (x_n, y_n)$. In particular, we run sketch synthesis independently for each example, obtaining sketches $\hat{\pi}_1, ..., \hat{\pi}_n$. Hole synthesis (only run once) is given all of these sketches; then, it replaces the condition $[\![\pi_0\, h\, \pi_1]\!] = y$ with the condition $\bigwedge_{i=1}^{n}([\![\pi_{i,0}\, h\, \pi_{i,1}]\!] = y_i)$, where $\hat{\pi}_i = \pi_{i,0}$?? $\pi_{i,1}$ for each $i \in \{1, ..., n\}$. That is, it ensures that h is consistent with *all* examples. Using multiple examples can reduce ambiguity (i.e., there may be multiple programs h that are consistent with a single example (x, y)).

Grammaticality-based substitution. So far, we have assumed that the type of the novel concept c is known. We describe a strategy for inferring its type; note that we continue to assume that the types for existing concepts $c' \in C_{\text{train}}$ are known. At a high level, we separately train a dedicated model to detect whether a given substitution is grammatical. In particular, for each type τ, let $C^{\tau}_{\text{train}} \subseteq C_{\text{train}}$ denote the known concepts of type τ. Now, for each type τ, we substitute c with each concept $c' \in C^{\tau}_{\text{train}}$ into $x = x_0\, c\, x_1$ to obtain $x' = x_0\, c'\, x_1$. Then, run a model $p_\theta(x') \in [0, 1]$ that predicts the probability that x' is grammatical. We choose the type τ such that these substitutions are grammatical with the highest confidence—i.e.,

$$\tau^* = \arg\max_{\tau} \frac{1}{|C^{\tau}_{\text{train}}|} \sum_{c' \in C^{\tau}_{\text{train}}} p_\theta(x_0\, c'\, x_1).$$

To train p_θ, we generate training data using our known concepts C_{train}, including both valid substitutions (labeled 1) and invalid ones (labeled 0).

4 Generalization Bounds

In this section, we prove generalization bounds on our approach adapted to context-free parsing.

4.1 Problem Formulation

We consider the problem of parsing—i.e., given a sentence $x \in X = \Sigma^*$, decide whether $x \in L(C^*)$. Here, $C^* = (V, \Sigma, R, S)$ is an *unknown* context-free grammar (CFG), where V are the nonterminals, Σ are the terminals, R are the productions, and $S \in V$ is the start symbol.[1] We assume that C^* is normalized—i.e., all productions are either unary $A \to B$ or binary $A \to BD$. For simplicity, we consider fixed-length sentences (i.e., $X = \Sigma^K$ for some $K \in \mathbb{N}$); we also assume all productions in C and $\tilde{C}$ are binary (i.e., there are no unary productions). In addition, we assume given a probability distribution $\mathcal{P}(x)$ over sentences; then, our goal is to achieve good performance for sentences $x \sim \mathcal{P}$.

We consider a novel concept in the form of a single production $\tilde{r} = \tilde{A} \to \tilde{B}\tilde{D}$ added to C^* to obtain a modified CFG $\tilde{C}^* = (V, \Sigma, \tilde{R}, S)$—i.e.,

$$\tilde{R} = R \cup \{\tilde{r}\}.$$

That is, $\tilde{C}^*$ equals C^* but with an extra production $\tilde{r}$. For our theoretical analysis, we assume given

- A learned model $g : \Sigma^* \to \{0, 1\}$ such that $g(w) \approx \mathbb{1}(w \in L(C^*))$ (more precisely, they are equal with high probability).
- The novel production $\tilde{r} = \tilde{A} \to \tilde{B}\tilde{C}$.

Then, our goal is to augment g with r to obtain a new classifier $\tilde{g}$ that works well for $\tilde{C}^*$ for $x \sim \mathcal{P}$.

4.2 Grammar-Based Approach

Next, we propose and analyze an algorithm for augmenting a learned grammar-based parser with the given novel production $\tilde{r}$.

Model. This strategy encodes the CFG as a function $\phi : W^3 \to \{0, 1\}$, where $W = V \cup \Sigma$. The corresponding CFG is $C_\phi = (V, \Sigma, R_\phi, S)$, where

$$R_\phi = \{A \to BD \in W^3 \mid \phi(A \to BD) = 1\}.$$

In other words, ϕ is the indicator function over all $|W|^3$ possible productions. Then, given a CFG C_ϕ, we construct a classifier $f_\phi : X \to \{0, 1\}$ by $f_\phi(x) = \mathbb{1}(x \in C_\phi)$. To implement this check, we assume f_ϕ runs a CYK parser on the input $x = \sigma_1 ... \sigma_K$—i.e., for each $(i, j) \in [K]^2$ (where $[K] = \{1, ..., K\}$) such that $i \le j$, it constructs the set $V^{i,j}_{\phi,x} \subseteq V$ inductively to be $V^{i,i}_{\phi,x} = \{\sigma_i\}$, and

$$V^{i,j}_{\phi,x} = \bigcup_{k=i}^{j-1} \left\{ A \in V \;\middle|\; \begin{array}{c} \exists B \in V^{i,k}_{\phi,x},\ D \in V^{k+1,j}_{\phi,x} \\ \text{s.t. } \phi(A \to BD) = 1 \end{array} \right\}$$

[1] The goal of semantic parsing is to compute the most probable parse; we consider the corresponding decision problem.

for $i < j$. Then, f_ϕ checks whether the start symbol is contained in the parse of the input x—i.e.,

$$f_\phi(x) = \mathbb{1}(S \in V_\phi^{1,K}(x))),$$

Algorithm. We consider an algorithm that takes as input a pretrained model f_ϕ designed to parse C, along with the novel production $\tilde{r}$; then, this algorithm returns the modified model $f_{\tilde{\phi}}$, where

$$\tilde{\phi}(r) = \begin{cases} 1 & \text{if } r = \tilde{r} \\ \phi(r) & \text{otherwise.} \end{cases}$$

That is, this algorithm simply overrides ϕ when $r = \tilde{r}$, thus augmenting it with the novel production.

4.3 Theoretical Analysis

Bounded error assumption. To obtain generalization bounds, we need to assume the accuracy of the given model f_ϕ is bounded. Specifically, we assume the accuracy of ϕ is bounded, and then bound the accuracy of f_ϕ in terms of the accuracy ϕ. In particular, we say ϕ is ϵ-correct if

$$\mathbb{P}_{p(r)}[\phi(r) = \phi^*(r)] \geq 1 - \epsilon,$$

where $\phi^*(r) = \mathbb{1}(r \in R)$, and $p(r)$ is the distribution over productions encountered by the CYK algorithm when compute $f_\phi(x)$ for a random sample $x \sim \mathcal{P}$; see Appendix A.1. That is, ϕ is ϵ-correct if it equals the ground truth ϕ^* at least $1 - \epsilon$ of the time. Similarly, we say f_ϕ is ϵ-correct if

$$\mathbb{P}_{\mathcal{P}(x)}[f_\phi(x) = f^*(x)] \geq 1 - \epsilon,$$

where $f^*(x) = \mathbb{1}(x \in L(C^*))$. Now, we have:

Lemma 1. *If ϕ is ϵ-correct, then the overall parsing model f_ϕ is $K^3|V|^3\epsilon$-correct.*

We give a proof in Appendix A.2.

Bounded shift assumption. In addition, we need to ensure that the shift from R to $\tilde{R}$ does not induce too large of a shift in terms of the distribution over productions—i.e., the distribution $p(r)$ of productions encountered while parsing $x \sim \mathcal{P}$ using C^* is not too different from the distribution $\tilde{p}(r)$ of productions encountered while parsing x using $\tilde{C}^*$. To this end, we need to bound the degree to which the novel production $\tilde{r}$ affects $p(r)$. In particular, we say $\tilde{r}$ is α-*bounded* if

$$\mathbb{E}_{\mathcal{P}(x)}\left[\tilde{B} \in V_{\phi^*,x}^{i,k} \wedge \tilde{D} \in V_{\phi^*,x}^{k+1,j}\right] \leq \alpha,$$

for all $i < k < j$. In other words, when parsing using C^*, the probability that $\tilde{r} = \tilde{A} \to \tilde{B}\tilde{D}$ would apply is bounded by α. Then:

Lemma 2. *If $\tilde{r}$ is α-bounded, then*

$$\sum_{r \in R} |\tilde{p}(r) - p(r)| \leq K^3|R|\alpha.$$

We give a proof in Appendix A.3.

Main result. Finally, our main result bounds the error of the modified model $f_{\tilde{\phi}}$ on $\tilde{C}^*$.

Theorem 1. *If ϕ is ϵ-correct for C^* and $\tilde{r}$ is α-bounded, then $f_{\tilde{\phi}}$ is $K^3|V|^3(\epsilon + K^3|R|\alpha)$-correct for $\tilde{C}^*$—i.e., letting $\tilde{f}^*(x) = \mathbb{1}(x \in L(\tilde{C}^*))$, then*

$$\mathbb{P}_{\mathcal{P}(x)}[f_{\tilde{\phi}}(x) \neq \tilde{f}^*(x)] \leq K^3|V|^3(\epsilon + K^3|R|\alpha).$$

We give a proof in Appendix A.4. Intuitively, the original error bound $K^3|V|^3\epsilon$; this result has an added factor of $(K^3|V|^3)^2\alpha$ (since $|R| \leq |V|^3$), so the error from the distribution shift grows roughly quadratically compared to the original error.

5 Experiments

In this section, we provide empirical evidence that our approach can perform few-shot novel concept learning. We use two existing datasets, SCAN and GeoQuery, which we have extended to our setting.

5.1 The HigherSCAN Dataset

Dataset. The SCAN dataset is a benchmark for evaluating systematicity in neural networks (Lake and Baroni, 2018; Loula et al., 2018). We extend SCAN to include different categories of novel concepts. The SCAN benchmark tests whether models can learn to understand instructions involving a novel primitive action such as *jump* without having seen *jump* in any context during training. However, *jump* is a terminal concept since it maps directly to the output token JUMP. We augment SCAN with higher-order novel concepts, where the novel concepts are programs composed of primitive concepts, and thus affect the structure of the output sequence. We consider the following novel concepts:

- **Extended Quantification:** We introduce the *n-times* input token, whose semantics are to repeat a given action n times.
- **Composite Actions**: We introduce a new input token whose denotation is a composite action—i.e., a sequence of primitive actions. For example, consider the novel concept $[\![jog]\!]$ = WALK RUN. The input instruction *jog twice and run* should have denotation

to WALK RUN WALK RUN RUN. We introduce several composite actions of varying complexity (i.e., length of its denotation).

The dataset including novel concepts is generated from the SCAN grammar augmented with these concepts. The modified training set consists of the original SCAN dataset, which does not include any of our novel concepts, along with a single example using the novel concept. The test set consists of examples that use our novel concepts. The original SCAN grammar generates 20910 unique examples. In HigherSCAN, we have 7706 new utterances corresponding to each novel composite action concept, and 11594 new utterances corresponding to each novel extended quantification concept.[2]

Our approach. We first train the neural semantic parser, which has a sequence-to-sequence encoder-decoder architecture (same as the baseline described below), on the original SCAN dataset. Then, for each novel concept, we run SDCL (Algorithm 1) with this semantic parser and a single training example for that novel concept. To train the grammaticality model, we randomly substitute words of a each type with those of different types in the original SCAN training examples to generate ungrammatical sentences, sampling a number of such ungrammatical sentences equal to the number of SCAN training examples. Then, we train a one-layered LSTM (with 50 hidden units) to predict the probability an instruction is grammatical.

Baseline. We compare to an end-to-end approach that uses a sequence-to-sequence neural network as the semantic parser, trained on the modified training set. We tried several architecture choices: LSTM cells vs. GRU cells, one vs. two layers, 100 vs. 200 hidden units, and with vs. without attention. We report results for the one-layered LSTM with 100 hidden units and with attention, which performed best on our validation set.

In addition, we compare to two variants of our approach SDCL: (i) one that uses oracle substitutions (i.e., the type of x is known, and (ii) one that uses confidence based substitutions. For (ii), we try two approaches: (a) train a separate model to predict whether the substitution x' is grammatical and (b) simply use the confidence of the semantic parser—i.e., we take $p_\theta(x')$ to be the confidence of our semantic parser in its predicted program for x'. Also, we use the product of the confidences rather than the average, which we find to work better.

[2]We sample 3000 examples for the test set, averaged over 5 draws. Importantly, there is no overlap between the test and training sets even after substituting similar-typed concepts.

Approach	*4-times*	*5-times*	*jog*	*gallop*
SDCL	**98.41**	**98.41**	**97.83**	**97.83**
SDCL (parser confidence)	65.35	65.35	39.72	39.72
Seq2Seq	3.34	1.32	0.32	0.00
SDCL (oracle)	99.12	99.12	99.40	99.40

Table 1: One-shot concept learning of extended quantification: *4-times*, *5-times* and composite actions: *jog* (WALK RUN), *gallop* (WALK RUN WALK RUN).

Results. In Table 1, we compare the performance of our algorithm SDCL to the baselines, for each of the concept categories. If we know the type of the novel concept (i.e., the oracle), then we are able to achieve near perfect accuracy. Furthermore, using a separate model trained to detect grammatical sentences from ungrammatical ones is highly effective. For the ablation; even the crude approach using the parser confidence to determine the type of the novel concept significantly outperforms end-to-end learning for the one-shot concept learning task.

Next, we demonstrate that end-to-end models cannot perform well even with a significantly larger number of examples. In Fig. 2, we show the performance of a sequence-to-sequence encoder-decoder model on the extended quantification and composite actions novel concepts, as a function of the number of times the novel concept is seen during training. In particular, we vary the number of training examples for the concept from $\{1, 2, 4, 8, 16, 32\}$. To ensure the novel concept training examples are balanced with the original SCAN dataset, we upweight the novel concept training examples. In particular, we fix the total size of the novel concept training set at 1600; when showing 8 different training examples, each one is included 200 times.

As can be seen, sequence-to-sequence models perform very poorly in the few-shot setting, and performance gradually improves as more examples of the novel concept are given.

One important observation is that both categories of novel concepts can make the length of the output program longer compared to examples in the original training data, which poses a challenge for end-to-end sequence models, especially when the concept has been seen only in a few instructions during training. Poor length extrapolation has also been observed to cause poor generalization in a different context (Lake and Baroni, 2018).

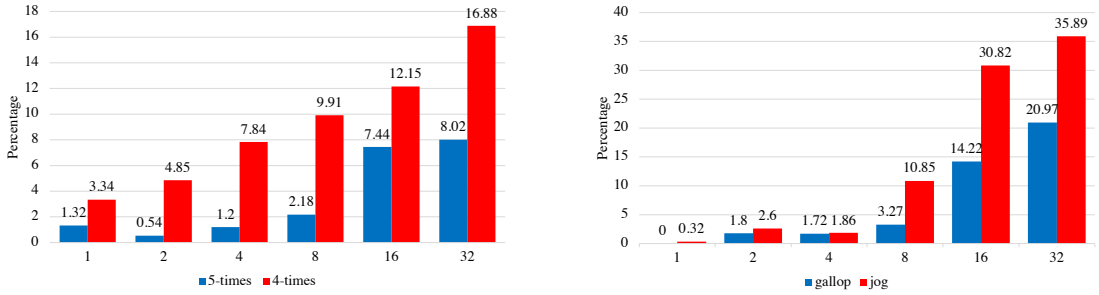

Figure 2: Accuracy as a function of the number of training examples for the extended quantification concepts *4-times* and *5-times* (left) and the composite action concepts *gallop* and *jog* from the HigherSCAN dataset.

5.2 The GeoQuery Dataset

Dataset. To evaluate the our approach beyond the synthetic SCAN dataset, we consider a modification of the GeoQuery dataset (Zelle, 1996) to include the extended quantification concepts. In particular, extended GeoQuery has concepts such as n^{th}-longest/shortest/largest/smallest river/mountain/state, etc. (we also change the corresponding predicates in the logical forms to include an argument n). As an example, the question "What are the major cities in the smallest state in the us?", which has corresponding logical form

$$(A,\ (\text{major(A)}, \text{city(A)}, \text{loc(A,B)},\\ \text{smallest}(B,\ (\text{state}(B),\ \text{loc}(B, C),\\ \text{const}(C,\ \text{countryid(USA)}))))))$$

is extended to the question "What are the major cities in the n^{th} smallest state in the US?", which coresponds to the logical form

$$(A,\ (\text{major(A)}, \text{city(A)}, \text{loc(A,B)},\\ \text{smallest}(n, B,\ (\text{state}(B),\ \text{loc}(B, C),\\ \text{const}(C,\ \text{countryid(USA)}))))))).$$

The GeoQuery dataset has a train/test split of 480/280 examples. After introducing the extended quantification concepts, the overall training set has 868 examples and the test set has 133 examples (for each extended quantification concept).

Our approach. We extend the GeoQuery training set with the extended quantification examples for $n = \{1, 2, 3\}$, and test on $n = \{4, 5\}$. First, we train a neural semantic parser (with same architecture as the baseline described below) on the extended training set using supervised learning.[3]

[3]Learning the semantic parser from only weak supervision (i.e., denotations instead of logical forms) is orthogonal to our goals, and is well-studied (Krishnamurthy et al., 2017).

Approach	*4-times*	*5-times*
SDCL	**71.42**	**71.42**
Seq2Seq	14.96	14.96
SDCL (oracle)	71.42	71.42

Table 2: Comparison on the one-shot concept learning task of extended quantification: *4-times* and *5-times* for the extended GeoQuery dataset.

Then, we run SDLC with this semantic parser and a single teaching example of novel concept, averaging results over 5 different choices of this example.

Baselines. We compare to the end-to-end model from (Jia and Liang, 2016), which is a single-layer sequence-to-sequence encoder-decoder architecture with attention, with 200 hidden units and trained for 30 epochs using stochastic gradient descent (with a learning rate of 0.1 which is halved after every 5 epochs starting from epoch 15). The baseline model is trained on the extended training set and the teaching example (repeated 24 times).

Results. Table 2 shows the accuracy of each approach on the extended quantification *4 times* and *5 times* concepts. As before, the end-to-end model is unable to learn the novel concepts from the a single training example, whereas SDCL is able to learn the novel concepts with high accuracy. For this dataset, the grammaticality model for substitution is able to perfectly identify the correct type.

6 Conclusion

We have proposed a novel approach for few-shot novel concept learning in semantic parsing. Our approach, SDCL, leverages substitutions to infer a sketch of the target program, and then uses program synthesis to infer the sub-program corresponding to the novel concept. Thus, SDCL incorporates symbolic techniques that are able to learn from few

examples into flexible end-to-end deep learning models. We have provided a theoretical analysis of how SDCL enables few-shot learning. Finally, we have empirically demonstrated that SDCL can learn novel concepts from a single example on two semantic parsing benchmarks, which we have extended to the novel concept learning setting.

Acknowledgements

Research was sponsored by the Army Research Office and was accomplished under Grant Number W911NF-20-1-0080. This work was supported by Contract FA8750-19-2-0201 with the US Defense Advanced Research Projects Agency (DARPA). The views expressed are those of the authors and do not reflect the official policy or position of the Department of Defense, the Army Research Office or the U.S. Government.

References

Jacob Andreas. 2020. Good-enough compositional data augmentation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7556–7566.

Yoav Artzi and Luke Zettlemoyer. 2013. Weakly supervised learning of semantic parsers for mapping instructions to actions. *Transactions of the Association for Computational Linguistics*, 1:49–62.

Pradeep Dasigi, Matt Gardner, Shikhar Murty, Luke Zettlemoyer, and Eduard Hovy. 2019. Iterative search for weakly supervised semantic parsing. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2669–2680.

Li Dong and Mirella Lapata. 2016. Language to logical form with neural attention. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 33–43.

Jerry A Fodor and Zenon W Pylyshyn. 1988. Connectionism and cognitive architecture: A critical analysis. *Cognition*, 28(1-2):3–71.

Robin Jia and Percy Liang. 2016. Data recombination for neural semantic parsing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12–22.

Rohit Kate and Raymond Mooney. 2006. Using string-kernels for learning semantic parsers. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 913–920.

Jayant Krishnamurthy, Pradeep Dasigi, and Matt Gardner. 2017. Neural semantic parsing with type constraints for semi-structured tables. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1516–1526.

Tom Kwiatkowksi, Luke Zettlemoyer, Sharon Goldwater, and Mark Steedman. 2010. Inducing probabilistic ccg grammars from logical form with higher-order unification. In *Proceedings of the 2010 conference on empirical methods in natural language processing*, pages 1223–1233.

Brenden Lake and Marco Baroni. 2018. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In *International Conference on Machine Learning*, pages 2873–2882. PMLR.

Brenden M Lake. 2019. Compositional generalization through meta sequence-to-sequence learning. *Advances in Neural Information Processing Systems*, 32.

Brenden M Lake, Tal Linzen, and Marco Baroni. 2019. Human few-shot learning of compositional instructions. In *Proceedings of the 41st Annual Conference of the Cognitive Science Society*.

João Loula, Marco Baroni, and Brenden Lake. 2018. Rearranging the familiar: Testing compositional generalization in recurrent networks. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 108–114.

Raymond J Mooney. 2007. Learning for semantic parsing. In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 311–324. Springer.

Armando Solar-Lezama. 2008. *Program synthesis by sketching*. Citeseer.

Pengcheng Yin and Graham Neubig. 2018. Tranx: A transition-based neural abstract syntax parser for semantic parsing and code generation. *arXiv preprint arXiv:1810.02720*.

John M Zelle. 1996. Learning to parse database queries using inductive logic programming.

A Proofs

A.1 Preliminaries

To facilitate our analysis, we let $R^{i,j}_{\phi,x}$ be the productions relevant to constructing $V^{i,j}_{\phi,x}$ from $V^{i,k}_{\phi,x}$ and $V^{k+1,j}_{\phi,x}$ (for $i \le k < j$)—i.e., $R^{i,i}_{\phi,x} = \varnothing$, and

$$R^{i,j}_{\phi,x} = \bigcup_{k=i}^{j-1} \left\{ A \to BD \;\middle|\; \begin{array}{l} A \in V,\ B \in V^{i,k}_{\phi,x}, \\ D \in V^{k+1,j}_{\phi,x} \in W \end{array} \right\}$$

for $i < j$. Then, the set of all productions relevant to parsing input x is

$$R_{\phi,x} = \bigcup_{i<j} R^{i,j}_{\phi,x}.$$

In addition, we let

$$\begin{aligned} \pi^{i,j}_{\phi,x}(A) &= \mathbb{1}(A \in V^{i,j}_{\phi,x}) \\ \pi^{i,j}_{\phi,x}(r) &= \mathbb{1}(r \in R^{i,j}_{\phi,x}) \\ \pi_{\phi,x}(r) &= \mathbb{1}(r \in R_{\phi,x}). \end{aligned}$$

In general, we use $\tilde{X}$ to denote the variant of X defined using $\tilde{R}$ instead of R. Also, we omit ϕ when $\phi = \phi^*$. For example, we have $\tilde{\pi}_x(r) = \mathbb{1}(r \in \tilde{R}_{\phi^*,x})$. Finally, the distribution over productions used is

$$p(r) = \sum_{x \in X} p(r \mid x) \cdot \mathcal{P}(x),$$

where

$$p(r \mid x) = \text{Uniform}(r; R_{\phi,x}).$$

That is, we need to correctly predict all productions considered by the CYK algorithm to avoid an error.

A.2 Proof of Lemma 1

We use the notation established in Appendix A.1. First, we define $N^{i,j}_{\phi,x}$ to be the number of times ϕ is used to construct $V^{i,j}_{\phi,x}$; in particular, we have $N^{i,i}_{\phi,x} = 0$, and

$$N^{i,j}_{\phi,x} = |R^{i,j}_{\phi,x}| = \sum_{k=i}^{j-1} |V| \cdot |V^{i,k}_{\phi,x}| \cdot |V^{k+1,j}_{\phi,x}| \le K|V|^3$$

for $i < j$, where the inequality follows since $|V^{i,i}_{\phi,x}| = 1$ and $|V^{i,j}_{\phi,x}| \le |V|$ for $i < j$ (and assuming $|V| \ge 1$). Note that the total number of applications of ϕ when parsing input x is

$$N_{\phi,x} = |R_{\phi,x}| = \sum_{i<j} N^{i,j}_{\phi,x} \le K^3|V|^3.$$

Thus, we have

$$\begin{aligned} \mathbb{P}_{\mathcal{P}(x)}[f_\phi(x) \neq f^*(x)] &\le \mathbb{P}_{\mathcal{P}(x)}\left[\exists r \in R_{\phi,x} \,.\, \phi(r) \neq \phi^*(r)\right] \\ &\le \sum_{x \in X} \sum_{r \in R_{\phi,x}} \mathbb{1}\left(\phi(r) \neq \phi^*(r)\right) \cdot \mathcal{P}(x) \\ &\le \sum_{x \in X} |R_{\phi,x}| \cdot \mathbb{P}_{p(r|x)}\left[\phi(r) \neq \phi^*(r)\right] \cdot \mathcal{P}(x) \\ &\le K^3|V|^3 \sum_{x \in X} \mathbb{P}_{p(r|x)}\left[\phi(r) \neq \phi^*(r)\right] \cdot \mathcal{P}(x) \\ &= K^3|V|^3 \cdot \mathbb{P}_{p(r)}\left[\phi(r) \neq \phi^*(r)\right] \\ &\le K^3|V|^3\epsilon, \end{aligned}$$

as claimed. □

A.3 Proof of Lemma 2

We use the notation established in Appendix A.1. First, we show that

$$\max_{A\in V}\max_{j-i\le t}|\tilde{\pi}_x^{i,j}(A)-\pi_x^{i,j}(A)|\le\max_{j-i\le t}\max_k\pi_x^{i,k}(\tilde{B})\pi_x^{k,j}(\tilde{D})=:\alpha_x,$$

where $j-i\le t$ denotes the set $i\in\{1,...,K\}$ and $j\in\{i,...,i+t\}$, and we implicitly assume $k\in\{i,...,j-1\}$. To this end, note that $\pi_x^{i,i}(A)=\mathbb{1}(A=\sigma_i)$, where $x=\sigma_1...\sigma_K$, and

$$\begin{aligned}\pi_x^{i,j}(A)&=\max_{B,D\in V}\max_k\pi_x^{i,k}(B)\pi_x^{k,j}(D)\cdot\mathbb{1}(A\to BD\in R)\\ \pi_x^{i,j}(A\to BD)&=\max_k\pi_x^{i,k}(B)\pi_x^{k,j}(D)\cdot\mathbb{1}(A\to BD\in R)\\ \pi_x(A\to BD)&=\max_{j-i\le K}\pi_x^{i,j}(A\to BD).\end{aligned}$$

Now, we proceed by induction. In particular, we have

$$\begin{aligned}\max_{A\in V}\max_{j-i\le t}|\tilde{\pi}_x^{i,j}(A)-\pi_x^{i,j}(A)|&\le\max\left\{\max_{A'\in V}\max_{j'-i'\le t-1}|\tilde{\pi}_x^{i',j'}(A')-\pi_x^{i',j'}(A')|,\max_k\tilde{\pi}_x^{i,k}(\tilde{B})\tilde{\pi}_x^{k,j}(\tilde{D})\right\}\\ &\le\max\left\{\max_{A'\in V}\max_{j'-i'\le t-1}|\tilde{\pi}_x^{i',j'}(A')-\pi_x^{i',j'}(A')|,\max_k\pi_x^{i,k}(\tilde{B})\pi_x^{k,j}(\tilde{D})\right\}\\ &\le\max\left\{\max_{A'\in V}\max_{j'-i'\le t-1}\max_{k'}\pi_x^{i',k'}(\tilde{B})\pi_x^{k',j'}(\tilde{D}),\max_k\pi_x^{i,k}(\tilde{B})\pi_x^{k,j}(\tilde{D})\right\}\\ &\le\max_{j-i\le t}\max_k\pi_x^{i,k}(\tilde{B})\pi_x^{k,j}(\tilde{D})\\ &=\alpha_x,\end{aligned}$$

where the second inequality follows since if $\tilde{\pi}_x^{i,k}(\tilde{B})\neq\pi_x^{i,k}(\tilde{B})$, then the first term in the max equals one (and similarly if $\tilde{\pi}_x^{k,j}(\tilde{D})\neq\pi_x^{k,j}(\tilde{D})$), and the third inequality follows by the inductive hypothesis. Now,

$$\begin{aligned}\max_{r\in R}\left|\frac{\tilde{\pi}_x(r)}{\sum_{r'\in R}\tilde{\pi}_x(r')}-\frac{\pi_x(r)}{\sum_{r'\in R}\pi_x(r')}\right|&\le\max_{r\in R}|\tilde{\pi}_x(r)-\pi_x(r)|\\ &\le\max_{r\in R}\max_{j-i\le t}|\tilde{\pi}_x^{i,j}(r)-\pi_x^{i,j}(r)|\\ &\le\max_{A\in V}\max_{j-i\le t}|\tilde{\pi}_x^{i,j}(A)-\pi_x^{i,j}(A)|\\ &\le\alpha_x,\end{aligned}$$

where the first inequality follows since the denominators are unequal only if $\tilde{\pi}_x(r')\neq\pi_x(r')$ for some $r'\in R$, since we are taking the max over $r\in R$ on the right-hand side, and since all these values are in $\{0,1\}$ so they can only be nonzero if they are unequal, the second inequality follows since $\tilde{\pi}_x(r)\neq\pi_x(r)$ only if $\tilde{\pi}_x^{i,j}(r)\neq\pi_x^{i,j}(r)$ for some $r\in R$ and $j-i\le t$, and the third inequality follows since $\tilde{\pi}_x^{i,j}(r)\neq\pi_x^{i,j}(r)$ only if $\tilde{\pi}_x^{i,j}(A)\neq\pi_x^{i,j}(A)$ for all $j-i\le t$ and $A\in V$. Finally, note that

$$p(r)=\sum_{x\in X}\text{Uniform}(r;R_{\phi,x})\cdot\mathcal{P}(x)=\sum_{x\in X}\frac{\tilde{\pi}_x(r)}{\sum_{r'\in R}\tilde{\pi}_x(r')}\cdot\mathcal{P}(x),$$

so

$$\max_{r\in R}|\tilde{p}(r)-p(r)|\le\sum_{x\in X}\max_{r\in R}\left|\frac{\tilde{\pi}_x(r)}{\sum_{r'\in R}\tilde{\pi}_x(r')}-\frac{\pi_x(r)}{\sum_{r'\in R}\pi_x(r')}\right|\cdot\mathcal{P}(x)\le\mathbb{E}_{\mathcal{P}(x)}[\alpha_x].$$

Finally, we have

$$\mathbb{E}_{\mathcal{P}(x)}[\alpha_x]=\mathbb{E}_{\mathcal{P}(x)}\left[\max_{j-i\le t}\max_k\pi_x^{i,k}(\tilde{B})\pi_x^{k,j}(\tilde{D})\right]\le\sum_{j-i\le t}\sum_k\mathbb{E}_{\mathcal{P}(x)}\left[\pi_x^{i,k}(\tilde{B})\pi_x^{k,j}(\tilde{D})\right]\le K^3\alpha,$$

so the claim follows. □

A.4 Proof of Theorem 1

Note that

$$\begin{aligned}\mathbb{P}_{\tilde{p}(r)}[\tilde{\phi}(r) \neq \tilde{\phi}^*(r)] &= \mathbb{P}_{p(r)}[\tilde{\phi}(r) \neq \tilde{\phi}^*(r)] + \sum_{r \in R} \mathbb{1}(\tilde{\phi}(r) \neq \tilde{\phi}^*(r)) \cdot (\tilde{p}(r) - p(r)) \\ &\leq \mathbb{P}_{\tilde{p}(r)}[\phi(r) \neq \phi^*(r)] + \sum_{r \in R} |\tilde{p}(r) - p(r)| \\ &\leq \epsilon + K^3 |R| \alpha,\end{aligned}$$

where the second inequality follows since $\tilde{\phi}(r) = \tilde{\phi}^*(r)$ if either $r = \tilde{r}$ or $\phi(r) = \phi^*(r)$, so either way $\phi(r) = \phi^*(r)$ implies $\tilde{\phi}(r) = \tilde{\phi}^*(r)$, and the third inequality follows by Lemma 2 and by the assumption that ϕ is ϵ-correct. Thus, the claim follows by Lemma 1 and the assumption that $\tilde{r}$ is α-bounded. □