A. Appendix

A.1. Semantics of specifications

We define the semantics of a specification $S = \{(T_1, \delta_1), \ldots, (T_n, \delta_n)\}$ (such that $T_i = (\varphi_i, f_i)$) as follows. Given a string $x = x_1 \ldots x_m$, a string $y$ is in the perturbations space $S(x)$ if:

1. there exists matches $\{(l_1, r_1), \ldots, (l_k, r_k)\}$ (we assume that matches are sorted in ascending order of $l_i$) such that for every $i \leq k$ we have that $(l_i, r_i)$ is a valid match of $\varphi_j$, in $x$;
2. the matches are not overlapping: for every two distinct $i_1$ and $i_2$, $r_{i_1} \leq l_{i_2}$ or $r_{i_2} < l_{i_1}$;
3. the matches respect the $\delta$ constraints: for every $j' \leq n$, $|\{(l_i, r_i), j_i \} | j_i = j'\} | \leq \delta_{j'}$.
4. the string $y$ is the result of applying an appropriate transformation to each match: if for every $i \leq k$ we have $s_i \in f_{j_i}(x_{l_i} \ldots x_{r_i})$, then $y = x_1 \ldots x_{l_1-1} s_1 x_{r_1+1} \ldots x_{l_2-1} s_2 x_{r_2+1} \ldots x_m$.

A.2. Proof of Theorem 1

We give the following definition of a convex set:

**Definition 1. Convex set**: A set $C$ is convex if, for all $x$ and $y$ in $C$, the line segment connecting $x$ and $y$ is included in $C$.

**Proof.** We first state and prove the following lemma.

**Lemma 2.** Given a set of points $\{p_0, p_1, \ldots, p_t\}$ and a convex set $C$ such that $\{p_0, p_1, \ldots, p_t\} \subseteq C$. These points define a set of vectors $p_0 p_1, p_0 p_2, \ldots, p_0 p_t$. If a vector $\overrightarrow{p_0 p}$ can be represented as a sum weighed by $\alpha_i$:

$$\overrightarrow{p_0 p} = \sum_{i=1}^{t} \alpha_i \cdot \overrightarrow{p_0 p_i},$$

where $\alpha_i$ respect to constraints:

$$\sum_{i=1}^{t} \alpha_i \leq 1 \land \forall 1 \leq i \leq t. \alpha_i \geq 0,$$

then the point $p$ is also in the convex set $C$.

**Proof.** We prove this lemma by induction on $t$.

- Base case: $t = 1$, if $\overrightarrow{p_0 p} = \alpha_1 \cdot \overrightarrow{p_0 p_1}$ and $0 \leq \alpha_1 \leq 1$, then $p$ is on the segment $p_0 p_1$. By the definition of the convex set (Definition 1), the segment $p_0 p_1$ is inside the convex, which implies $p$ is inside the convex: $p \in p_0 p_1 \subseteq C$.

- Inductive step: Suppose the lemma holds for $t = r$. If a vector $\overrightarrow{p_0 p'}$ can be represented as a sum weighed by $\alpha_i$:

$$\overrightarrow{p_0 p'} = \sum_{i=1}^{r+1} \alpha_i \cdot \overrightarrow{p_0 p_i}$$

where $\alpha_i$ respect to constraints:

$$\sum_{i=1}^{r+1} \alpha_i \leq 1, \quad \forall 1 \leq i \leq r+1. \alpha_i \geq 0.$$

We divide the sum in Eq 6 into two parts:

$$\overrightarrow{p_0 p'} = \sum_{i=1}^{r} \alpha_i \cdot \overrightarrow{p_0 p_i} + \sum_{i=r+1}^{r+1} \alpha_i \cdot \overrightarrow{p_0 p_{r+1}}$$

Because from Inequality 7, we know that

$$\sum_{i=1}^{r} \alpha_i \leq 1 - \alpha_{r+1},$$

which is equivalent to

$$\sum_{i=1}^{r} \frac{\alpha_i}{1 - \alpha_{r+1}} \leq 1.$$  

This inequality enables the inductive hypothesis, and we know point $p'$ is in the convex set $C$. From Eq 11, we know that the point $p$ is on the segment of $p' p_{r+1}$, since both two points $p'$ and $p_{r+1}$ are in the convex set $C$, then the point $p$ is also inside the convex set $C$.

To prove Theorem 1, we need to show that every perturbed sample $y \in S(x)$ lies inside the convex hull of $\text{abstract}(S, x)$.

**We first describe the perturbed sample $y$.** The perturbed sample $y$ as a string is defined in the semantics of specification $S$ (see the Appendix A.1). In the rest of this proof, we use a function $E : \Sigma^m \rightarrow \mathbb{R}^{m \times d}$ mapping from a string with length $m$ to a point in $m \times d$-dimensional space, e.g., $E(y)$ represents the point of the perturbed sample $y$ in the
We then prove the Theorem 1. To prove $E(y)$ lies in the convex hull of $\text{abstract}(S, x)$, we need to apply Lemma 2. Notice that a convex hull by definition is also a convex set. Because from Eq 14, we have

$$E(x)E(y) = \sum_{i=1}^{k} \Delta_{((l_i,r_i),j_i,s_i)}.$$ 

We further define $\Delta_{((l_i,r_i),j_i,s_i)}$ as the vector $E(x_{((l_i,r_i),j_i,s_i)}) - E(x) = E(x)E(x_{((l_i,r_i),j_i,s_i)})$:

$$\Delta_{((l_i,r_i),j_i,s_i)} = \begin{pmatrix} 0, \ldots, 0, E(s), \ldots, 0 \end{pmatrix}.$$ 

A perturbed sample $y$ defined by matches $\langle (l_1, r_1), \ldots, (l_k, r_k), j_k \rangle$ and for every $i \leq k$ we have $s_i \in f_{j_i}(x_{l_i} \ldots x_{r_i})$, then

$$y = x_1 \ldots x_{l_1-1} s_1 x_{r_1+1} \ldots x_{l_k-1} s_k x_{r_k+1} \ldots x_m.$$ 

The matches respect the $\delta$ constraints: for every $j' \leq n$, $|\{(l_i,r_i),j_i,s_i) \mid j_i = j'\}| \leq \delta_{j'}$. Thus, the size of the matches $k$ also respect the $\delta$ constraints:

$$k = \sum_{j'=1}^{n} |\{(l_i,r_i),j_i,s_i) \mid j_i = j'\}| \leq \sum_{j'=1}^{n} \delta_{j'}.$$ (13)

In the embedding space,

$$E(x)E(y) = \begin{pmatrix} 0, \ldots, 0, E(s_1), \ldots, 0 \end{pmatrix} - E(x_{l_1} \ldots x_{r_1}),$$

$$0, \ldots, 0, E(s_k) - E(x_{l_k} \ldots x_{r_k}), \ldots, 0 \end{pmatrix}.$$ (14)

Thus, we can represent $E(x)E(y)$ using $\Delta_{((l_i,r_i),j_i,s_i)}$:

$$\sum_{i=1}^{k} \Delta_{((l_i,r_i),j_i,s_i)}.$$ (14)

We then describe the convex hull of $\text{abstract}(S, x)$. The convex hull of $\text{abstract}(S, x)$ is constructed by a set of points $E(x)$ and $E(v_{((l_i,r_i),j_i,s_i)})$, where points $E(v_{((l_i,r_i),j_i,s_i)})$ are computed by:

$$E(v_{((l_i,r_i),j_i,s_i)}) = E(x) + \sum_{i=1}^{n} \delta_{i} (E(x_{((l_i,r_i),j_i,s_i)}) - E(x)).$$

Alternatively, using the definition of $\Delta_{((l_i,r_i),j_i,s_i)}$, we get

$$E(x)E(v_{((l_i,r_i),j_i,s_i)}) = \sum_{i=1}^{n} \delta_{i} \Delta_{((l_i,r_i),j_i,s_i)}.$$ (15)

A3. Details of Experiment Setup

For AG dataset, we trained a smaller character-level model than the one used in Huang et al. (2019). We followed the setup of the previous work: use lower-case letters only and truncate the inputs to have at most 300 characters. The model consists of an embedding layer of dimension 64, a 1-D convolution layer with 64 kernels of size 10, a ReLU layer, a 1-D average pooling layer of size 10, and two fully-connected layers with ReLUs of size 64, and a linear layer. We randomly initialized the character embedding and updated it during training.

For SST2 dataset, we trained the same word-level model as the one used in Huang et al. (2019). The model consists of an embedding layer of dimension 300, a 1-D convolution layer with 100 kernels of size 5, a ReLU layer, a 1-D average pooling layer of size 5, and a linear layer. We used the pre-trained Glove embedding (Pennington et al., 2014) with dimension 300 and fixed it during training.
For SST2 dataset, we trained the same character-level model as the one used in Huang et al. (2019). The model consists of an embedding layer of dimension 150, a 1-D convolution layer with 100 kernels of size 5, a ReLU layer, a 1-D average pooling layer of size 5, and a linear layer. We randomly initialized the character embedding and updated it during training.

For all models, we used Adam (Kingma & Ba, 2015) with a learning rate of 0.001 for optimization and applied early stopping policy with patience 5.

A.3.1. Perturbations

We provide the details of the string transformations we used:

- \( T_{\text{SubAdj}}, T_{\text{InsAdj}} \): We allow each character substituting to one of its adjacent characters on the QWERTY keyboard.
- \( T_{\text{DelStop}} \): We choose \{and, the, a, to, of\} as our stop words set.
- \( T_{\text{SubSyn}} \): We use the synonyms provided by PPDB (Pavlick et al., 2015). We allow each word substituting to its closest synonym when their part-of-speech tags are also matched.

A.3.2. Baseline

Random augmentation performs adversarial training using a weak adversary that simply picks a random perturbed sample from the perturbation space. For a specification \( S = \{(T_1, \delta_1), \ldots, (T_n, \delta_n)\} \), we produce \( z \) by uniformly sampling one string \( z_1 \) from a string transformation \( (T_1, \delta_1) \) and passing it to the next transformation \( (T_2, \delta_2) \), where we then sample a new string \( z_2 \), and so on until we have exhausted all transformations. The objective function is the following:

\[
\arg\min_{\theta} \mathbb{E}_{(x,y) \sim D} \left( L(x,y,\theta) + \max_{z \in R(x)} L(z,y,\theta) \right) \tag{18}
\]

HotFlip augmentation performs adversarial training using the HotFlip (Ebrahimi et al., 2018) attack to find \( z \) and solve the inner maximization problem. The objective function is the same as Eq 18.

A3T adopts a curriculum-based training method (Huang et al., 2019; Gowal et al., 2019) that uses a hyperparameter \( \lambda \) to weigh between normal loss and maximization objective in Eq. (2). We linearly increase the hyperparameter \( \lambda \) during training.

\[
\arg\min_{\theta} \mathbb{E}_{(x,y) \sim D} \left( (1 - \lambda)L(x,y,\theta) + \lambda \max_{z \in \text{augment}_k(S_{\text{adv}}, x)} L(\text{abstract}(S_{\text{adv}}, z), y, \theta) \right). 
\]

Also, we set \( k \) in \( \text{augment}_k \) to 2, which means we select 2 perturbed samples to abstract.

A.3.3. Evaluation Results

RQ2: Effects of size of the perturbation space In Figure 4, we fix the word-level model A3T (search) trained on \{(TDup, 2), (TSubSyn, 2)\}. Then, we test this model’s exhaustive accuracy on \{(TDup, \delta_1), (TSubSyn, 2)\} (Figure 4(a)) and \{(TDup, 2), (TSubSyn, \delta_2)\} (Figure 4(b)), where we vary the parameters \( \delta_1 \) and \( \delta_2 \) between 1 and 4, increasing the size of the perturbation space. The exhaustive accuracy of A3T(HotFlip) and A3T(search) decreases by 17.4% and 11.4%, respectively, when increasing \( \delta_1 \) from 1 to 4, and decreases by 2.3% and 1.9%, respectively, when increasing \( \delta_2 \) from 1 to 4. All other techniques result in larger decreases in exhaustive accuracy (≥17.5% in \{(TDup, \delta_1), (TSubSyn, 2)\}) and ≥3.1% in \{(TDup, 2), (TSubSyn, \delta_2)\}).

In Figure 5, we fix the word-level model A3T (search) trained on \{(TDelStop, 2), (TDup, 2), (TSubSyn, 2)\}. Then, we test this model’s exhaustive accuracy on \{(TDelStop, \delta_1), (TDup, 2), (TSubSyn, 2)\} (Figure 5(a)), \{(TDelStop, 2), (TDup, \delta_2), (TSubSyn, 2)\} (Figure 5(b)), and \{(TDelStop, 2), (TDup, 2), (TSubSyn, \delta_3)\} (Figure 5(c)), where we vary the parameters \( \delta_1 \), \( \delta_2 \) and \( \delta_3 \) between 1 and 3, increasing the size of the perturbation space. The exhaustive accuracy of A3T(HotFlip) and A3T(search) decreases by 1.1% and 0.9%, respectively, when increasing \( \delta_1 \) from 1 to 3, decreases by 12.9% and 6.9%, respectively, when increasing \( \delta_2 \) from 1 to 3, and decreases by 1.4% and 0.9%, respectively, when increasing \( \delta_3 \) from 1 to 3. All other techniques result in larger decreases in exhaustive accuracy (≥2.2% in \{(TDelStop, \delta_1), (TDup, 2), (TSubSyn, 2)\}) ≥13.0% in \{(TDelStop, 2), (TDup, \delta_2), (TSubSyn, 2)\}), and ≥2.8% in \{(TDelStop, 2), (TDup, 2), (TSubSyn, \delta_3)\}).
Figure 4. The exhaustive accuracy of \( \{(T_{\text{Dup}}, \delta_1), (T_{\text{SubSyn}}, \delta_2)\} \), varying the parameters \( \delta_1 \) (left) and \( \delta_2 \) (right) between 1 and 4.
Figure 5. The exhaustive accuracy of \{ \{ T_{DelStop}, \delta_1 \}, \{ T_{Dup}, 2 \}, \{ T_{SubSyn}, 2 \} \}, varying the parameters \delta_1 \text{ (left)}, \delta_2 \text{ (middle), and } \delta_3 \text{ (right) between 1 and 3.}