A Noise-tolerant Differentiable Learning Approach for Single Occurrence Regular Expression with Interleaving

Rongzhen Ye [1], Tianqu Zhuang [1], Hai Wan [1,*], Jianfeng Du [2,*], Weilin Luo [1], Pingjia Liang [1]

[1] School of Computer Science and Engineering, Sun Yat-sen University, Guangzhou, China
[2] Guangzhou Key Laboratory of Multilingual Intelligent Processing, Guangdong University of Foreign Studies, China

Content

Content

Definition of Problem

Learning a regular expression (RE) from a set of text strings.

- focus on *single occurrence regular expression with interleaving* (SOIRE)
- full expressive power
- noisy data

Example 1

Given the set of strings as follows:

- positive: ab, abc, bac, bacc
- negative: ac, aac, bb, **ba** (noise)

Find a SOIRE that maximizes the accuary of matching. Here is $(a\&b)c^*$.

Significance and Challenge

Wide applications:

- XML database system [2,11,12]
- system verification [1,3]
- natural language processing [5,13]

Challenging tasks:

- heavy computation in searching to guarantee the full expressive power
- wrong search bias resulting from noisy data

Related Work

Learning approaches for different subclass of SOIREs:

- positive strings only
  - chain regular expression with interleaving (ICHARE) [14]
  - restricted SOIRE (RSOIRE) [8]
  - k-occurrence regular expression with interleaving (kOIRE) [6]
- both positive and negative strings
  - subclass of ICHARE, called SIRE [7]
- natural language descriptions with positive and negative strings (Different from ours)
  - domain specific language [10]

Developing a differentiable learning approach to learn arbitrary SOIREs from noisy data.

Content

Outline

The outline of the approach `SOIREDL`:

- train a neural network that simulates SOIRE matching
  - matching algorithm `SOIRETM`
  - convert `SOIRETM` to a neural network
- interpret the target SOIRE from the parameters
  - correspondence between parameters of the neural network and SOIREs
  - find the nearest faithful encoding

Filter Matching

Filter matching for a SOIRE r and a string s is to check if r matches $filter(s, \alpha(r))$, where $\alpha(r)$ denotes the set of symbol in a SOIRE r, and the filter function $filter(s, V)$ returns a string that only retains symbols in V, where $V \subseteq \Sigma$.
Let $g_{i,j}^t \in \{0, 1\}$ $(1 \leq t \leq |r|, 1 \leq i, j \leq |s|)$ denote whether r^t matches $filter(s_{i,j}, \alpha(r^t))$, where $s_{i,j}$ denotes the substring of s from i to j, and where $s_{1,0} = \epsilon$ specially.

Example 2

For Figure 1 and $s = dbac$,

- filter matching is to check if $(a\&b)c^*$ matches $filter(dbac, \{a, b, c\}) = bac$, as $\alpha((a\&b)c^*) = \{a, b, c\}$.
- $g_{1,2}^2$ denotes if $a\&b$ matches ba and $g_{1,2}^2 = 1$.

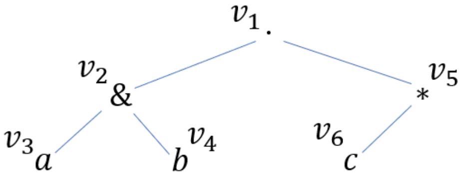

Figure 1: The syntax tree of SOIRE $(a\&b)c^*$. $v_1, \ldots, v_6$ represent $\cdot$, $\&$, a, b, $*$ and c, respectively.

r^t denotes the corresponding SOIRE of the subtree with root t. For example, $r^2 = a\&b$.

SOIRE	Semantics of $r \models filter(s, \alpha(r))$
$r = a \in \Sigma$	1. $filter(s, a) = a$.
$r = r_1^? = \epsilon \vert r_1$	1. $filter(s, \alpha(r_1^?)) = \epsilon$. 2. $r_1 \models filter(s, \alpha(r_1))$.
$r = r_1^* = \epsilon \vert r_1 \vert r_1^* \cdot r_1$	1. $filter(s, \alpha(r_1^*)) = \epsilon$. 2. $r_1 \models filter(s, \alpha(r_1))$. 3. $r_1^* \models filter(s_1, \alpha(r_1^*))$ and $r_1 \models filter(s_2, \alpha(r_1))$, where $s = s_1 s_2 (s_1, s_2 \neq \epsilon)$.
$r = r_1^+ = r_1 \vert r_1^+ \cdot r_1$	1. $r_1 \models filter(s, \alpha(r_1))$. 2. $r_1^+ \models filter(s_1, \alpha(r_1^+))$ and $r_1 \models filter(s_2, \alpha(r_1))$, where $s = s_1 s_2 (s_1, s_2 \neq \epsilon)$.
$r = r_1 \cdot r_2$	1. $r_1 \models filter(s, \alpha(r_1 \cdot r_2))$ and $r_2 \models \epsilon$. 2. $r_2 \models filter(s, \alpha(r_1 \cdot r_2))$ and $r_1 \models \epsilon$. 3. $r_1 \models filter(s_1, \alpha(r_1 \cdot r_2))$ and $r_2 \models filter(s_2, \alpha(r_1 \cdot r_2))$, where $s = s_1 s_2 (s_1, s_2 \neq \epsilon)$.
$r = r_1 \& r_2$	$r_1 \models filter(s, \alpha(r_1))$ and $r_2 \models filter(s, \alpha(r_2))$.
$r = r_1 \vert r_2$	1. $r_1 \models filter(s, \alpha(r_1 \vert r_2))$. 2. $r_2 \models filter(s, \alpha(r_1 \vert r_2))$.

Table 1: The semantics of filter matching, where r, r_1, r_2 are SOIREs and s, s_1, s_2 are strings. $r \models filter(s, \alpha(r))$ if and only if at least one condition on the right is satisfied.

SOIRETM

Theorem 1

Given a SOIRE r and a string s, $r \models s$ iff $filter(s, \alpha(r)) = s$ and $r \models filter(s, \alpha(r))$.

The step of `SOIRETM`:

1. build the syntax tree of r
2. check if $filter(s, \alpha(r)) = s$
3. calculate $g^t_{i,j}$ from shorter substrings to longer ones and from bottom to top of the syntax tree (dynamic programming)
4. return $g^1_{1,|s|}$ (filter matching)

Theorem 2

Given a SOIRE r and a string s, $r \models s$ iff $\mathtt{SOIRETM}(r, s) = 1$.

Trainable Parameters

The trainable parameters $\theta = (w, u)$:

- $w \in [0,1]^{T\times|\mathbb{B}|}$, w_a^t denotes the probability of vertex t representing a symbol in Σ or an ordinary operator or the none operator.
- $u \in [0,1]^{T\times T}$, $u_{t'}^t$ denotes the probability of vertex t choosing vertex t' as its right child.

where $\mathbb{B} = \Sigma \cup \{?, *, +, \cdot, \&, |, \text{none}\}$, and T is the bounded size of the target SOIRE.

Example 3

When $T = 6$, $w_\cdot^1$, $w_\&^2$, w_a^3, w_b^4, w_*^5, w_c^6, u_5^1, u_4^2 are 1s, whereas other parameters are 0s.
When $T = 8$, w_{none}^7, w_{none}^8 are 1s in addition to the above parameters.

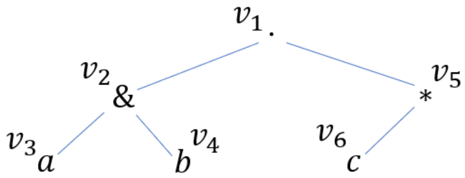

Figure 2: The syntax tree of SOIRE $(a\&b)c^*$. $v_1, \ldots, v_6$ represent $\cdot$, $\&$, a, b, $*$ and c, respectively.

Conversion of SOIRETM to Neural Network

There are four parts that should be considered:

- $\alpha(r^t)$, the set of symbols occurring in the subtree whose root is t
  Let ρ_a^t denote the probability of symbol $a \in \Sigma$ that occurs in the subtree whose root is t.
  Calculate ρ_a^t from bottom to top of the syntax tree by Equation 1, where
  $\sigma_{01}(x) = \min(\max(x, 0), 1)$.

$$\rho_a^t = \sigma_{01}(w_a^t + \sum_{o \in \{?, *, +, \cdot, \&, |\}} w_o^t \rho_a^{t+1} + \sum_{o \in \{\cdot, \&, |\}} w_o^t \sum_{t'=t+2}^{T} u_{t'}^t \rho_a^{t'}) \quad (1)$$

- $flag_{i,j}^{t,t'}$, $flag_{i,j}^{t,t'} = 1[filter(s_{i,j}, \alpha(r^t)) = filter(s_{i,j}, \alpha(r^{t'}))]$
  Treat it as the probability that there does not exist a symbol occurring in both $s_{i,j}$ and $\alpha(r^t)$ but not occurring in $\alpha(r^{t'})$, as defined in Equation 2.

$$flag_{i,j}^{t,t'} = 1 - \sigma_{01}(\sum_{a \in \Sigma} \sigma_{01}(1[a \in s_{i,j}] + (\rho_a^t - \rho_a^{t'}) - 1)) \quad (2)$$

- $g^t_{i,j}$, whether r^t matches $filter(s_{i,j}, \alpha(r^t))$
  - $p^t_{i,j}(?)$, $p^t_{i,j}(*)$ and $p^t_{i,j}(+)$: the probability that the right-hand side evaluates to 1.
  - $p^t_{i,j}(\cdot, t')$, $p^t_{i,j}(\&, t')$ and $p^t_{i,j}(|, t')$): the probability that the right-hand side with η^t substituted by t' evaluates to 1.
  - The probability that $g^t_{i,j}$ evaluates to 1 can be defined recursively by Equation 3.

$$g^t_{i,j} = \sum_{a \in \Sigma} w^t_a \cdot 1[filter(s_{i,j}, a) = a] + \sum_{o \in \{?,*,+\}} w^t_o p^t_{i,j}(o) + \sum_{o \in \{\cdot,\&,|\}} w^t_o \sum_{t'=t+2}^{T} u^t_{t'} p^t_{i,j}(o, t') \tag{3}$$

- return value of `SOIRETM`
  Combine $filter(s, \alpha(r)) = s$ and $g^1_{1,|s|}$.

$$\hat{y} = g^1_{1,|s|} - \max_{a \in \Sigma} \sigma_{01}(1[a \in s] - \rho^1_a) \tag{4}$$

The converted neural network is trained to minimize the objective function $\frac{1}{2}(\hat{y} - y)^2$, where $y \in \{0, 1\}$ is the ground-truth label for r matching s.

Faithful Encoding

Definition 1 (Faithful encoding)

An encoding $\theta = (w, u)$ of SOIREs with length T is said to be *faithful* if it satisfies all the following conditions:

1. $\forall 1 \leq t \leq T, w^t$ is a one-hot vector.
2. $\forall 1 \leq t \leq T, u^t$ is either a one-hot vector or an all-zero vector.
3. $\forall 1 \leq t \leq T, \sum_{t'=t+2}^{T} u_{t'}^t + \sum_{a \in \Sigma \cup \{?,*,+,\text{none}\}} w_a^t = 1.$
4. $\forall 1 \leq t \leq T-1, w_{\text{none}}^{t+1} - w_{\text{none}}^t \geq 0.$
5. $\forall 2 \leq t \leq T, \sum_{a \in \{?,+,*,\cdot,\&,|\}} w_a^{t-1} + \sum_{t'=1}^{t-2} u_t^{t'} + w_{\text{none}}^t = 1.$
6. $\forall 3 \leq t \leq T, \forall 1 \leq p \leq t-2, (t-1-p)u_t^p + \sum_{p'=p+1}^{t-1} \sum_{t'=t+1}^{T} u_{t'}^{p'} \leq t-1-p.$
7. $\forall a \in \Sigma, \sum_{t=1}^{T} w_a^t \leq 1.$

Example 4

When $T = 6$, $w^1_{\cdot}$, $w^2_{\&}$, w^3_a, w^4_b, w^5_*, w^6_c, u^1_5, u^2_4 are 1s, whereas other parameters are 0s.
When $T = 8$, w^7_{none}, w^8_{none} are 1s in addition to the above parameters.

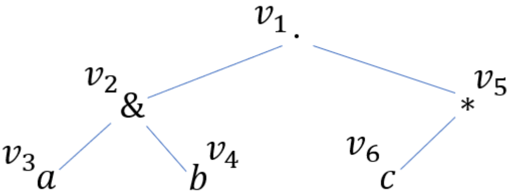

Figure 3: The syntax tree of SOIRE $(a\&b)c^*$. $v_1, \ldots, v_6$ represent $\cdot$, $\&$, a, b, $*$ and c, respectively.

- $\texttt{Enc2Pre}(\theta)$: the prefix notation of the SOIRE interpreted from a faithful encoding θ.
- $\texttt{PreForm}(r)$: the prefix notation of the SOIRE r.

Proposition 3

For any faithful encoding θ, there exists a SOIRE r such that $\mathit{Enc2Pre}(\theta) = \mathit{PreForm}(r)$.

Theorem 4

Given a bounded size $T \in \mathbb{Z}^+$, prefix notations of SOIREs r with $|r| \leq T$ and faithful encodings θ with length T have a one-to-one correspondence, i.e., $\mathit{Enc2Pre}(\theta) = \mathit{PreForm}(r)$.

Interpretation

We apply beam search to find a faithful encoding nearby the learnt encoding and then interpret it to the target SOIRE.

1. The interpretation steps are conducted from bottom to top of the syntax tree.
   - keep β candidate SOIREs for each subtree according to their score, which is defined as the geometric mean of the probabilities of all operators and symbols.
   - select different operators and candidate SOIREs from the left child and right child (if any) and merge them to generate new candidates.
2. calculate the accuracy of each SOIRE in the last step on the training set and pick out the SOIRE with the highest accuracy.

Content

Setting

Benchmarks:

- 30 SOIREs from the RE database built by Li et al. (2018)[9]
- 5 noise levels: $\delta = \{0, 0.05, 0.1, 0.15, 0.2\}$
- For each SOIRE r,
  - training sets and test sets: $|\Pi^+| = |\Pi^-| = 250$
  - validation sets: $|\Pi^+| = |\Pi^-| = 50$
- For each δ, reverse the labels for $|\Pi^+|\delta$ positive strings and $|\Pi^-|\delta$ negative strings in the training and validation sets.

Competitors:

- Positive strings only: `iSOIRE`[8] for RSOIREs, `GenICHARE`[14] for ICHAREs
- Both positive and negative strings: `iSIRE`[7] for SIREs, `RE2RNN`[4] for automatons, `SOIREDL` (Ours) for SOIREs

Task:

- All approaches first learn a SOIRE from the training and validation set and then are compared by evaluating the accuracy of matching of the learnt SOIREs on the test set.

Comparison on Noise-free Data

- `SOIREDL` outperforms `iSIRE` and `RE2RNN`, and achieves comparable performance with `iSOIRE` and `GenICHARE`.
- `SOIREDL` achieves the highest average accuracy among all approaches.
- Regarding the accuracy of the intermediate neural network, `SOIREDL` is also superior to `RE2RNN`.

These results show that `SOIREDL` achieves the SOTA performance on noise-free data.

Dataset	Positive and negative strings			Positive strings only				
	iSIRE	RE2RNN	SOIREDL	iSOIRE	GenICHARE	iSIRE	RE2RNN	SOIREDL
1	86.0	52.4 (94.4)	**100.0** (100.0)	**89.4**	**89.4**	83.8	48.0 (50.0)	57.0 (57.0)
2	77.4	48.8 (91.4)	**100.0** (100.0)	**100.0**	**100.0**	**100.0**	50.4 (50.0)	**100.0** (100.0)
3	90.8	50.6 (77.4)	**100.0** (100.0)	**100.0**	97.2	95.6	50.6 (49.6)	65.4 (65.4)
4	72.4	49.0 (80.2)	**99.6** (73.4)	**73.8**	72.6	73.4	49.6 (49.8)	61.4 (61.4)
5	**90.8**	52.6 (91.4)	58.2 (87.2)	**100.0**	**100.0**	86.2	49.6 (50.2)	52.8 (52.8)
6	77.8	51.6 (62.2)	**93.4** (93.0)	**100.0**	**100.0**	70.4	50.2 (50.0)	52.6 (52.6)
7	81.2	52.2 (95.8)	**99.2** (99.2)	**100.0**	96.4	89.0	49.2 (49.8)	69.4 (69.4)
8	76.2	49.8 (88.0)	**100.0** (100.0)	**100.0**	**100.0**	**100.0**	49.8 (50.0)	**100.0** (100.0)
9	93.8	44.2 (48.4)	**98.4** (98.4)	**99.8**	98.8	98.0	47.2 (49.6)	81.6 (81.6)
10	94.8	50.2 (89.8)	**99.8** (100.0)	**99.8**	**99.8**	82.8	44.4 (50.2)	61.2 (61.2)
11	**91.0**	52.4 (88.6)	89.2 (91.0)	**100.0**	**100.0**	91.4	47.6 (50.2)	57.4 (57.4)
12	78.6	51.2 (98.4)	**100.0** (100.0)	**100.0**	**100.0**	**100.0**	50.8 (49.4)	**100.0** (100.0)
13	96.6	52.8 (50.2)	**100.0** (100.0)	**100.0**	**100.0**	83.6	51.2 (49.6)	67.0 (67.0)
14	74.4	50.0 (79.0)	**84.8** (71.6)	70.0	**74.4**	68.8	49.0 (50.0)	54.4 (54.4)
15	95.2	54.0 (49.8)	**96.2** (100.0)	**100.0**	**100.0**	**100.0**	47.8 (50.2)	66.2 (66.2)
16	**96.8**	49.0 (71.4)	94.8 (100.0)	**100.0**	**100.0**	75.4	49.4 (50.0)	75.4 (75.4)
17	91.0	46.2 (87.2)	**100.0** (100.0)	**100.0**	**100.0**	**100.0**	50.6 (50.0)	**100.0** (100.0)
18	81.0	42.8 (78.8)	**87.4** (99.2)	87.4	**100.0**	82.0	40.2 (50.2)	53.0 (53.0)
19	88.2	50.0 (63.8)	**100.0** (93.4)	**100.0**	**100.0**	88.0	50.4 (50.0)	54.6 (54.6)
20	**93.4**	45.2 (54.4)	83.4 (90.0)	**100.0**	99.2	95.8	47.4 (50.0)	56.0 (51.2)
21	69.8	48.8 (96.2)	**100.0** (100.0)	**71.2**	**71.2**	**71.2**	49.8 (50.4)	**71.2** (71.2)
22	90.6	47.6 (50.6)	**100.0** (100.0)	**100.0**	**100.0**	91.4	50.0 (50.2)	58.2 (58.2)
23	85.6	32.8 (84.4)	**86.8** (65.2)	**90.0**	**90.0**	88.0	50.0 (48.8)	56.8 (56.8)
24	69.4	49.0 (57.2)	**74.6** (76.8)	**77.6**	76.0	71.4	47.6 (50.0)	54.2 (54.2)
25	67.8	54.2 (93.4)	**99.8** (74.6)	**70.0**	**70.0**	**70.0**	50.2 (50.0)	69.6 (69.6)
26	**80.2**	50.4 (50.4)	63.8 (98.2)	**100.0**	**100.0**	85.2	51.0 (49.6)	63.8 (63.8)
27	92.0	52.0 (91.6)	**96.4** (96.4)	**100.0**	**100.0**	97.4	52.2 (50.2)	67.6 (67.6)
28	65.0	54.8 (95.2)	**100.0** (98.6)	**65.6**	**65.6**	**65.6**	50.0 (50.0)	**65.6** (65.6)
29	93.4	56.2 (75.8)	**97.6** (97.2)	**100.0**	99.8	81.2	49.2 (49.8)	54.2 (54.2)
30	60.4	41.4 (61.6)	**78.8** (79.6)	**61.0**	**61.0**	60.4	49.8 (49.8)	54.8 (54.8)
Avg.	83.4	49.4 (76.6)	**92.7** (92.8)	91.9	**92.0**	84.9	49.1 (49.9)	66.7 (66.6)

Table 2: Accuracy (%) on noise-free data with best results in bold. For X(Y), X denotes the accuracy of the learnt SOIRE or automaton, and Y the accuracy of the neural network.

Comparison on Noisy Data

- The performance of `iSOIRE`, `GenICHARE` and `RE2RNN` suggest that they are not robust on noisy data.
- Both `SOIREDL` and `iSIRE` perform well on noisy data.
- The average accuracy of `SOIREDL` slightly decreases, but it still keeps higher than others at all noise levels

This suggests that `SOIREDL` is the most robust on noisy data.

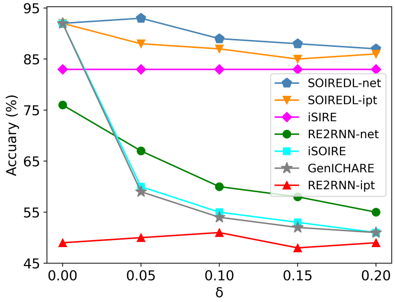

Figure 4: Average accuracy (%) on test sets at different noise levels δ with positive and negative strings. SOIREDL-ipt and RE2RNN-ipt represent the learnt SOIREs or automata, whereas SOIREDL-net and RE2RNN-net represent the neural networks.

Comparison in terms of Faithfulness

For SOIREDL and RE2RNN, We introduce faithfulness defined as $\frac{N_=}{|\Pi^+|+|\Pi^-|}$ to evaluate the consistency between the neural network and the interpreted SOIRE (SOIREDL) or automata (RE2RNN), where $N_=$ is the number of test strings that the neural network and the SOIRE or automata predicts the same label.

- The faithfulness of SOIREDL is much higher than that of RE2RNN and keeps over 80% at all noise levels.

The neural network of SOIREDL and its interpreted SOIRE are more consistent in performing SOIRE matching.

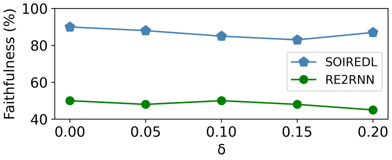

Figure 5: Average faithfulness (%) of SOIREDL and RE2RNN on test sets at different noise levels δ.

Case study

These results show that `SOIREDL` is able to learn different subclasses of SOIREs.

This may be due to the difficulty for a neural network to capture the long-distance dependency in SOIRE matching.

Subclass	Dataset	Ground-truth SOIREs	SOIREDL
SIRE	13	$a^?\&b^*\&c^?$	$a^?\&c^?\&b^*$
ICHARE	22	$(a\|b)^* c^* d^*$	$(a^*\&b^*) c^* d^*$
RSOIRE	3	$a^+\|(b\|c)^*\|d^+$	$(b^+\|c)^*\|a^*\|d^*$
SOIRE	1	$((a\|b) c^*)^+ d$	$((a\|b) c^*)^+ d$

Table 3: The ground-truth SOIREs and the SOIREs learnt by SOIREDL on different subclasses of SOIREs.

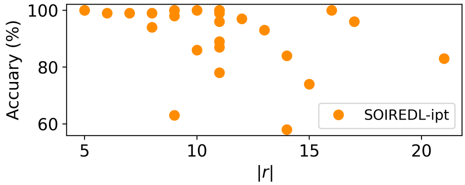

Figure 6: The relation of the accuracy (%) of a SOIRE learnt by `SOIREDL` and the size $|r|$ of the ground-truth SOIRE r.

The Necessity for Using Negative Strings

- Neural networks do not perform well because they prefer to classify unseen strings as positive ones when training on positive strings only.
- This suggests that negative strings are crucial in effectively learning with noisy data, possibly because they infer what kinds of strings that the target SOIRE cannot match.

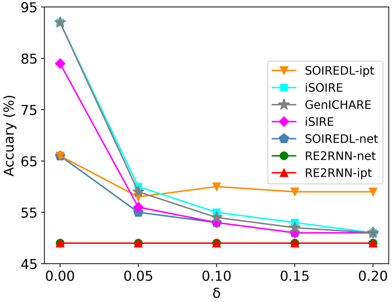

Figure 7: Average accuracy (%) on test sets at different noise levels δ with positive strings only. SOIREDL-ipt and RE2RNN-ipt represent the learnt SOIREs or automata, whereas SOIREDL-net and RE2RNN-net represent the neural networks.

Content

Conclusion and Future Work

Conclusion:

1. We have proposed a noise-tolerant differentiable learning approach `SOIREDL`. The neural network introduced in `SOIREDL` simulates SOIRE matching.
2. Theoretically, the faithful encodings learnt by `SOIREDL` one-to-one correspond to SOIREs for a bounded size.
3. Experimental results have demonstrated higher performance compared with the SOTA approaches.

Future work:

1. tackle the problem of long dependency in SOIRE matching.
2. extend our approach to other subclasses of REs.

References I

[1] Bojanczyk, M.; Muscholl, A.; Schwentick, T.; Segoufin, L.; and David, C. 2006. Two-Variable Logic on Words with Data. In *LICS*, 7–16.

[2] Colazzo, D.; Ghelli, G.; Pardini, L.; and Sartiani, C. 2013. Efficient asymmetric inclusion of regular expressions with interleaving and counting for XML type-checking. *Theor. Comput. Sci.*, 492: 88–116.

[3] Gischer, J. L. 1981. Shuffle Languages, Petri Nets, and Context-Sensitive Grammars. *Commun. ACM*, 24(9): 597–605.

[4] Jiang, C.; Zhao, Y.; Chu, S.; Shen, L.; and Tu, K. 2020. Cold-Start and Interpretability: Turning Regular Expressions into Trainable Recurrent Neural Networks. In *EMNLP*, 3193–3207.

[5] Kuhlmann, M.; and Satta, G. 2009. Treebank Grammar Techniques for Non-Projective Dependency Parsing. In *EACL*, 478–486.

[6] Li, Y.; Cao, J.; Chen, H.; Ge, T.; Xu, Z.; and Peng, Q. 2020. FlashSchema: Achieving High Quality XML Schemas with Powerful Inference Algorithms and Large-scale Schema Data. In *ICDE*, 1962–1965.

[7] Li, Y.; Chen, H.; Zhang, L.; Huang, B.; and Zhang, J. 2020. Inferring Restricted Regular Expressions with Interleaving from Positive and Negative Samples. In *PAKDD*, volume 12085, 769–781.

[8] Li, Y.; Chen, H.; Zhang, X.; and Zhang, L. 2019. An effective algorithm for learning single occurrence regular expressions with interleaving. In *IDEAS*, 24:1–24:10.

[9] Li, Y.; Chu, X.; Mou, X.; Dong, C.; and Chen, H. 2018. Practical Study of Deterministic Regular Expressions from Large-scale XML and Schema Data. In *IDEAS*, 45–53.

[10] Li, Y.; Li, S.; Xu, Z.; Cao, J.; Chen, Z.; Hu, Y.; Chen, H.; and Cheung, S. 2021. TRANSREGEX: Multi-modal Regular Expression Synthesis by Generate-and-Repair. In *ICSE*, 1210–1222.

[11] Makoto, M.; and Clark, J. 2003. RELAX NG. https://relaxng.org/.

References II

[12] Martens, W.; Neven, F.; Niewerth, M.; and Schwentick, T. 2017. BonXai: Combining the Simplicity of DTD with the Expressiveness of XML Schema. *ACM Trans. Database Syst.*, 42(3): 15:1–15:42.

[13] Nivre, J. 2009. Non-Projective Dependency Parsing in Expected Linear Time. In *ACL*, 351–359.

[14] Zhang, X.; Li, Y.; Cui, F.; Dong, C.; and Chen, H. 2018. Inference of a Concise Regular Expression Considering Interleaving from XML Documents. In *PAKDD*, volume 10938, 389–401.

Thank you for your listening!