The Thirty-Third International
FLAIRS Conference (FLAIRS-33)

Learning Picture Languages Represented as Strings *

David Kuboň, František Mráz
{dkubon, mraz}@ksvi.mff.cuni.cz
Faculty of Mathematics and Physics
Dept. of Software and Computer Science Education
Charles University, Prague, Czech Republic

Abstract

Analysis of two-dimensional (picture) formal languages is of similar importance as analysis of their one-dimensional (string) counterparts but is lacking state-of-the-art algorithms for their learning. In this paper, we introduce a new representation of picture languages based on mapping pictures to strings. The representation enables to learn picture languages by applying methods of grammatical inference for string languages. We propose a learning protocol and evaluate it on several picture languages.

Introduction

Formal two-dimensional languages are of significant theoretical and practical importance, but nevertheless do not share the extent of knowledge as their one-dimensional counterparts (Giammarresi and Restivo 1997), even though they can also be used as formal models of practical problems such as automatic detection of different shapes (e.g. road signs) or generally any problem on two-dimensional data which has some pattern regularity.

These languages can be defined as sets of two-dimensional pictures with formally exact description. Although they are referred to as *picture* languages they do not correspond to pictures in the general sense, as such cannot be defined mathematically – a set of photos containing dogs is not a formal picture language. Consequently, while deep neural networks have remarkable results on recognising objects in non-formal pictures, they do not perform well on formal languages. However, dedicated tools designed in particular for formal two-dimensional languages, such as powerful models of two-dimensional automata, are typically nondeterministic and thus inefficient for practical use.

The imbalance between knowledge about one- and two-dimensional formal languages can be seen also with regards to their learning. While there are known algorithms of grammatical inference – i.e. the process of learning a grammar for a language based on information about its words – for some classes of string languages, there is considerably less knowledge about grammatical inference for picture languages. Therefore, in this paper, we would like to investigate methods for learning two-dimensional languages from positive and negative samples.

Literature presents us with two types of representation of pictures and two-dimensional languages – generative and analytic. Generative representations of a picture (Freeman 1961; Maurer, Rozenberg, and Welzl 1982; Costagliola et al. 2003) describe how to draw a picture – how to move a pen over a plane. Analytic representation of a picture works with a rectangular array of symbols. Each symbol can be interpreted as the colour of a pixel in the image. The set of pictures, a language, is then represented by an automaton that decides if a given picture belongs to the set or not. Some automata models, e.g. non-deterministic online tessellation automaton (Giammarresi and Restivo 1997), more powerful sgraffito automaton (Průša, Mráz, and Otto 2014) or two-dimensional limited context restarting automaton (2LCRA) (Krtek 2014), are quite powerful but of high complexity – the problem of acceptance of a picture by such automaton is NP-complete.

A greater goal of designing automata for two-dimensional languages would be to find such classes of picture languages that would share similar properties and concepts – e.g. being context-free, closure properties regarding some operations – like some one-dimensional classes. In an ideal scenario, a reduction in dimensionality would lead to an analogical one-dimensional class.The main advantage then would be the possibility to relate two-dimensional classes to hierarchies – e.g. Chomsky hierarchy – as picture languages lack a similar concept and their taxonomy is more complicated and lacks a complete description. Following this concept, several of the mentioned models representing generalisations of regular languages into two dimensions when restricted to one dimension lead back to a regular language. However, the classes they accept are fairly restricted and the membership problem of a picture belonging to such language is still NP-complete.

Many attempts have been made with the learning of automata accepting picture languages, typically experimenting with supervised learning methods. For example (Krtek 2014) learned 2LCRA's by generating a list of all feasible reductions for every position of the picture and selecting sets of locally compatible reductions. However, this method is not very efficient and dedicated heuristics and enhancements need to be used to obtain results in a reasonable time.

*This research was supported by the Charles University Grant Agency (GAUK) project no. 1198519.
Copyright © 2020, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

As learning one-dimensional languages has more successful results, we would like to employ it for learning picture languages, too. In this paper, we propose a new representation for two-dimensional languages. It combines the generative and analytic approach by using rectangular arrays of symbols to represent pictures and string languages to represent sets of pictures. Its core lies in a function R, which transforms any two-dimensional word p into a string and a one-dimensional language L. Then a picture language is defined as the set of all pictures p for which $R(p) \in L$. Moreover, by combining a classical algorithm for inferring regular languages from positive and negative samples with the proposed function R we get a protocol for learning picture languages, which we evaluate in some basic experiments.

Definitions

The first attempt to study sets of objects using methods for one-dimensional formal languages were chain codes (Maurer, Rozenberg, and Welzl 1982), where a picture is a set of unit length lines in the Cartesian plane and words are over an alphabet $\Pi = \{l, r, u, d\}$ representing left, right, up and down movements. An interpretation of a word is the drawing obtained by starting at any point with integer coordinates in the Cartesian plane and moving a pen in the directions indicated by the alphabet symbols – e.g. the letter 'L' could be drawn as $uuddr$. The set of such drawings is a regular language (Maurer, Rozenberg, and Welzl 1982).

A picture obtained in this way is clearly connected; a non-connected picture could be created by considering an extension of the alphabet Π by symbols $\uparrow$ and $\downarrow$ corresponding to lifting and lowering the pen above the drawing plane.

Another frequently used definition of a picture reflects more pictures in the common sense. A *picture* P over a finite alphabet Σ is a two-dimensional rectangular array of elements from Σ – see (Giammarresi and Restivo 1997). We say that P has dimensions (m, n), if it has m rows and n columns. The set of all rectangular pictures over Σ of dimensions (m, n) will be denoted as $\Sigma^{m,n}$ and the set of all rectangular pictures over Σ of any dimension will be denoted as $\Sigma^{*,*}$. A *picture language* is any subset of $\Sigma^{*,*}$.

Any automaton working on a picture P of dimensions (m, n) needs to know where is the border of the picture, therefore the picture is typically surrounded by sentinels $\#$, where $\# \notin \Sigma$. Delimited picture P is called a *boundary picture* $\widehat{P}$ over $\Sigma \cup \{\#\}$ of dimensions $(m + 2) \times (n + 2)$.

Algorithms For Learning Regular Languages

In this paper we experiment with learning from positive samples only and with learning from both positive and negative samples.

Let $k > 1$ be an integer. The class of k-testable languages is a subclass of regular languages that is learnable from positive samples (De la Higuera 2010). A (string) language $L \subseteq \Sigma^*$ is called *k-testable* , if there exist four finite sets of words: $I, F \subseteq \Sigma^{k-1}$, $C \subseteq \Sigma^{<k}$, and $T \subseteq \Sigma^k$ such that a word belongs to L if it is from C, or its prefix of length $k - 1$ is in I, its suffix of length $k - 1$ is in F, and all substrings of length k belong to T.

Given k and a sufficiently large set of positive samples S^+ from L, it is easy to learn L by collecting C as the set of all words in S^+ of length less than k, I as the set of prefixes of length $k - 1$ of words in S^+, F as the set of all suffixes of length $k - 1$ of words in S^+, and T as the set of all factors of length k of the words in S^+.

Regular Positive and Negative Inference (RPNI) (Oncina and García 1992) is a basic example of algorithms for learning regular languages from positive and negative samples. RPNI begins by separating the training set S into a set of positive samples S^+, and a set of negative samples S^-. Then a prefix tree automaton is built from all samples in S^+. The states of the automaton correspond to all prefixes of words from S^+, which is a finite language. Next, the algorithm goes through the states of the current automaton and attempts to merge pairs of states. If no negative sample from S^- is accepted by the merged automaton, it becomes the current automaton. Otherwise, the merge is canceled, the current automaton is not changed and the algorithm tries to merge the next pair in a predetermined order. This algorithm guarantees to produce a finite state automaton consistent with the samples in set S.

RPNI does not perform well on sparse data, where it can almost never be sure that a compatible merge is truly valid and thus most of its actions are hopeful guesses (Lang, Pearlmutter, and Price 1998). The first approach to address this issue was using breadth-first search for state merging (Lang 1992) in the *traxbar* algorithm, with the idea that a valid merge involving the largest sub-trees in the prefix tree has a higher probability of being correct than other merges. Later (De La Higuera, Oncina, and Vidal 1996) suggested to find evidence and based on it select the best merge candidates, leading to *Evidence Driven State Merging* (EDSM) algorithm, which was improved in (Lang, Pearlmutter, and Price 1998).

Like with RPNI, the algorithm starts with a prefix tree automaton. In each step of the loop, for every pair of candidate nodes a merging score is computed and only the highest scoring one is performed.To compute the score, it is assumed the two nodes are equivalent and equivalence classes are computed for all the remaining nodes using the rule that all children of equivalent nodes are equivalent. Merge score is then the sum over equivalence classes.

Another version (Juillé and Pollack 1998) called *red-blue* extends a connected set of red nodes corresponding to unique states. Remaining nodes, blue, will either be merged with red nodes or re-coloured to red. Only heterochromatic merges are allowed.

A New Representation for Picture Languages

We propose a new representation for picture languages consisting of two parts: a function $R : \Sigma^{*,*} \to (\Gamma \cup \{\#\})^*$ which maps any two-dimensional picture over Σ into a string over $\Gamma \cup \{\#\}$ and a (string) language $L \subseteq (\Gamma \cup \{\#\})^*$. Then (R, L)-picture language is the set of all pictures $P \in \Sigma^{*,*}$ such that $R(P)$ is in L. We say that a string language L^{S} R-represents a picture language L^{P}, if for each picture P it holds that $P \in L^{\mathrm{P}}$ if and only if $R(P) \in L^{\mathrm{S}}$.

Clearly, there are multiple ways how to map a picture into a string. We can represent any rectangular picture $P \in \Sigma^{*,*}$ by arranging all symbols (i.e. colours of pixels) row-by-row into a string. However, in such representation, the information about the dimensions of the picture is lost. This can be addressed by using delimiter $\#$ to mark ends of rows, e.g. the language L_1^{P} of all non-empty pictures filled with b's will be represented by the (string) language $L_1^{\mathrm{S}} = \{(b^n\#)^{m-1}b^n \mid m, n > 0\}$, which is evidently a context-sensitive but not a context-free language. Nonetheless, not all words over $\{b, \#\}$ represent pictures and we would like to have a representation that enables us to represent simple picture languages (like L_1^{P}) by simple (e.g. low in the Chomsky hierarchy) string languages. Fortunately, the sample picture language L_1^{P} can be also represented by the string language defined by the regular expression $(b^+\#)^*b^+$.

However, regular representation of many elementary picture languages, like the language $L_{\mathrm{v1}}^{\mathrm{P}}$ of white pictures containing a single black column, is not possible with the above representation. Nevertheless, the language $L_{\mathrm{v1}}^{\mathrm{P}}$ could be represented using a function R rewriting the picture into a string column-by-column, as opposed to row-by-row. To generalise this idea instead of allowing multiple reading strategies, we use a reading window of size 3-by-3. The function R_{rw} scans a picture P with this window row-by-row and rewrites the window contents row-by-row into a string, intuitively preserving the information about neighbouring rows and columns of the picture. E.g. the picture of dimensions 2-by-3 containing a's in the first and last column and b's in the middle column will be rewritten into the string (the vertical lines were added to clearly separate the contents of the window at different positions and are not present in the actual string):

$$\#\#\#\#ab\#ab|\#\#\#abaaba|\#\#\#ba\#ba\#|$$
$$\#ab\#ab\#\#\#|abaaba\#\#\#|ba\#ba\#\#\#\#$$

Experimental Results

In order to verify the suitability of our new representation of picture languages for their learning, we conducted a series of experiments with 10 picture languages over the binary alphabet $\{\square, \blacksquare\}$ corresponding to white and black pixels, respectively (see Fig. 1 for sample pictures):

L_1 is the set of all white rectangles containing a black diagonal till the border of the picture.

L_2 is the set of all white pictures of dimensions at least $(3, 3)$ with a black border of one pixel width.

L_3 is the set of all pictures with a positive number of black rows followed by a positive number of white rows.

L_4 is the set of all pictures with a regular chessboard pattern.

L_5 is the set of all white pictures containing one black vertical line, which can be shorter than the height of the picture.

L_6 is the set of all white pictures containing two black vertical lines, which must not touch and can be shorter than the height of the picture.

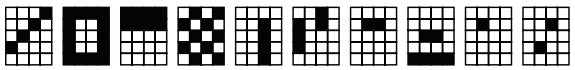

Figure 1: Sample pictures from languages $L_1, \ldots, L_{10}$.

L_7 is the set of all white pictures containing one horizontal line, which can be shorter than the width of the picture.

L_8 is the set of all white pictures containing two horizontal lines, which must not touch and can be shorter than the width of the picture.

L_9 is the set of all white pictures containing one black pixel.

L_{10} is the set of all white pictures containing two black pixels, which must not touch (even diagonally).

In our experiments, we have generated training and testing sets of positive and negative sample pictures of sizes 100, 200, 400, 800 and 1600 words, for each sample language. Samples contained pictures of dimensions between 3 and 8 such that each training set contained approximately the same number of positive and negative samples with one exception: for experiments using learning of k-locally testable (string) languages we have used only positive training samples.

In each experiment, we first rewrite each picture into a string by applying the function R_{rw} with window size 3-by-3. Then for each set S of sample strings, we learn a deterministic finite-state automaton consistent with S. For that we have used one algorithm learning from positive samples only and two versions of state merging algorithms:

a) Learning k-locally testable languages from positive examples using implementation in MATLAB from (Akram et al. 2010).

b) A program `traxbar` which implements a version of breadth-first Trakhtenbrot-Barzdin's state merging algorithm (Lang 1992).

c) An EDSM program `red-blue` by Juille (Lang, Pearlmutter, and Price 1998).

Afterwards, we tested each resulting automaton on an independent test set of pictures (rewritten into strings by the function R_{rw}). Each learned finite automaton correctly recognizes its training set of pictures, however, it need not accept/reject correctly all pictures from the test set.

Learning k-locally testable languages requires to set up the window size k. For that, we have conducted a series of experiments with different values for k between 2 and 18 (i.e. two times the area of the scanning window 3-by-3 used by R_{rw}). The obtained results differ for different languages. The worst accuracy – at most 0.9 – was obtained for language L_{10} even for the largest training sets. Based on these experiments, we have chosen 9 as the best value for k. Fig. 2 presents the accuracy of the learned automata for all sample picture languages and for all sizes of training sets.

Further, we have used the state merging algorithms for learning from both positive and negative samples implemented in programs `traxbar` and `red-blue`. The accuracy of the learned automata with respect to the size of

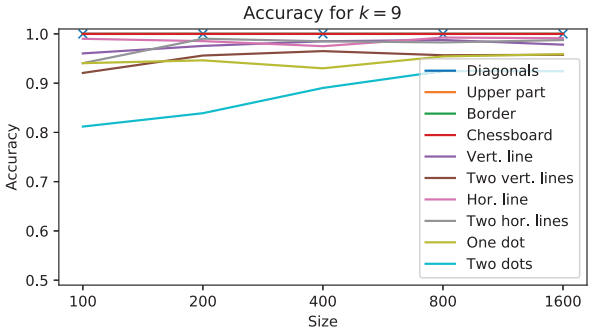

Figure 2: The accuracy of automata learned as automata for 9-locally testable languages $L_1, \ldots, L_{10}$ with respect to the size of the training set. Note that for L_1 to L_4 the accuracy was exactly 1.0 for all sizes of training sets in our tests.

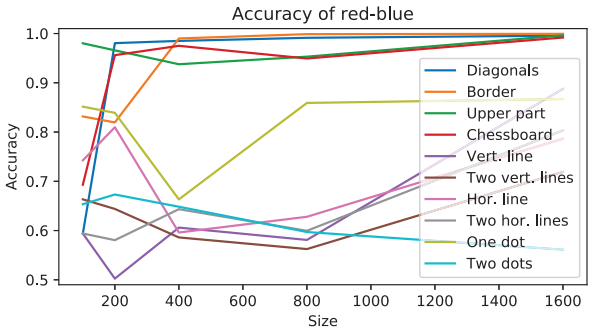

Figure 3: The accuracy of automata learned by `red-blue` state merging algorithm.

the training set when using the slightly better performing `red-blue` is depicted in Fig. 3.

Surprisingly, the best accuracy was achieved by using learning from positive samples only. This is caused by the fact, that the sample picture languages $L_1, \ldots, L_4$, are evidently local (Giammarresi and Restivo 1997), i.e. the language is fully characterized by the set of tiles (contents of the scanning windows used by R_{rw}) that can appear in the bordered picture. This can be clearly seen in Fig. 2 where the accuracy of the learned automata for $L_1, \ldots, L_4$ is 1.0. The remaining sample languages are not local, but often an inspection of the set of tiles in an input picture enables us to classify the picture correctly.

On the other hand, the automata learned from both positive and negative samples by state merging methods have worse accuracy. Evidently, each of our sample languages can be R_{rw}-represented by a regular (string) language. However, neither `traxbar` nor `red-blue` was able to learn such regular language. This can be caused by small training sets or the fact that the training sets are sparse. The lengths of all words in training sets were multiples of 9.

Conclusions

Our new representation of picture languages can be used for learning picture languages. As it is composed of transforming a picture into a string and then checking membership for a string language, any method of grammatical inference for string languages can be employed.

We experimented with one such transformation represented by the function R_{rw} that rewrites the picture by storing the contents of a scanning window of size 3-by-3 when scanning given picture row by row. For learning, we have tried learning 9-locally testable languages from positive samples only and two methods based on state merging from both positive and negative samples.

The obtained results seem to be promising, but we are not satisfied with the achieved accuracy. For better accuracy of the learned automata in our future research, we will modify the employed state merging algorithms and possibly also use different transformations from pictures to strings.

References

Akram, H. I.; De La Higuera, C.; Xiao, H.; and Eckert, C. 2010. Grammatical inference algorithms in matlab. In *ICGI 2020*, 262–266. Springer.

Costagliola, G.; Deufemia, V.; Ferrucci, F.; and Gravino, C. 2003. On regular drawn symbolic picture languages. *Information and Computation* 187(2):209–245.

De La Higuera, C.; Oncina, J.; and Vidal, E. 1996. Identification of DFA: Data-dependent versus data-independent algorithms. In *International Colloquium on Grammatical Inference*, 313–325. Springer.

De la Higuera, C. 2010. *Grammatical inference: learning automata and grammars*. Cambridge University Press.

Freeman, H. 1961. On the encoding of arbitrary geometric configurations. *IRE Transactions on Electronic Computers* EC-10(2):260–268.

Giammarresi, D., and Restivo, A. 1997. Two-dimensional languages. In *Handbook of Formal Languages: Volume 3 Beyond Words*. Berlin, Heidelberg: Springer. 215–267.

Juillé, H., and Pollack, J. B. 1998. A sampling-based heuristic for tree search applied to grammar induction. In *AAAI/IAAI*, 776–783.

Krtek, L. 2014. Learning picture languages using restarting automata. master thesis, Charles University, Faculty of Mathematics and Physics.

Lang, K. J.; Pearlmutter, B. A.; and Price, R. A. 1998. Results of the abbadingo one DFA learning competition and a new evidence-driven state merging algorithm. In *ICGI 1998*, volume 1433 of *LNCS*, 1–12. Springer.

Lang, K. J. 1992. Random DFA's can be approximately learned from sparse uniform examples. In *COLT 1992*, 45–52. ACM.

Maurer, H. A.; Rozenberg, G.; and Welzl, E. 1982. Using string languages to describe picture languages. *Information and Control* 54(3):155–185.

Oncina, J., and García, P. 1992. Inferring regular languages in polynomial updated time. In *Pattern recognition and image analysis: selected papers from the IVth Spanish Symposium*, 49–61. World Scientific.

Průša, D.; Mráz, F.; and Otto, F. 2014. Two-dimensional sgraffito automata. *RAIRO-Theoretical Informatics and Applications* 48(5):505–539.