Theoretical and Implementational Aspects of the Formal Language Server (LaSer)

Stavros Konstantinidis(✉)

Mathematics and Computing Science, Saint Mary's University,
923 Robie St., Halifax, NS B3H 3C3, Canada
s.konstantinidis@smu.ca

Abstract. LaSer, the formal language server, allows a user to enter a question about an independent language and provides an answer either in real time or by generating a program that can be executed at the user's site. Typical examples of independent languages are codes in the classic sense, such as prefix codes and error-detecting languages, or DNA-computing related codes. Typical questions about independent languages are the satisfaction, maximality and construction questions. We present some theoretical and implementational aspects of LaSer, as well as some ongoing progress and research plans.

Keywords: Independent languages · Regular languages · Codes · DNA codes · Property satisfaction · Maximality · Implementation

1 Introduction

LaSer, [18], the formal language server, allows a user to enter a question about an independent language and provides an answer either in real time or by generating a program that can be executed at the user's site. Typical examples of independent languages are codes in the classic sense, such as prefix codes and error-detecting languages, or DNA-computing related codes. Typical questions about independent languages are the satisfaction, maximality and construction questions. For example, the satisfaction question is to decide, given an independence $\mathcal{I}$ and a regular language L, whether L is independent with respect to $\mathcal{I}$.

We present some theoretical and implementational aspects of LaSer, as well as some ongoing progress and research plans.

2 Transducer Independences Allowed in LaSer

Let R be a binary relation, that is, a subset of $\Sigma^* \times \Sigma^*$, where Σ is an alphabet. A language L is R-independent, [21,22], if

$$(x, y) \in R \text{ and } x, y \in L \text{ implies } x = y. \tag{1}$$

Research supported by NSERC, Canada.

© Springer Nature Switzerland AG 2020

Examples of R-independent languages are error-detecting languages for various error combinations, variable length codes, such as prefix codes and suffix codes, as well as DNA-related languages [2,5,6,10,12–14,23]. LaSer allows users to represent rational relations via transducers[1], and regular languages via NFAs[2]. The satisfaction question is whether $\mathtt{L}(\boldsymbol{a})$ is $\mathtt{R}(\boldsymbol{t})$-independent, given an NFA $\boldsymbol{a}$ and a transducer $\boldsymbol{t}$. If the answer is NO, a witness word pair (x, y) is computed such that $x \neq y$ and one of (x, y) and (y, x) is in $\mathtt{R}(\boldsymbol{t})$. The maximality question is whether $\mathtt{L}(\boldsymbol{a})$ is a maximal $\mathtt{R}(\boldsymbol{t})$-independent language knowing that $\mathtt{L}(\boldsymbol{a})$ is $\mathtt{R}(\boldsymbol{t})$-independent. If the answer is NO, a witness word $x \notin \mathtt{L}(\boldsymbol{a})$ is computed such that $\mathtt{L}(\boldsymbol{a}) \cup \{x\}$ is $\mathtt{R}(\boldsymbol{t})$-independent. The construction question is to make an n-element language (if possible) that is $\mathtt{R}(\boldsymbol{t})$-independent, given transducer $\boldsymbol{t}$, integer $n > 0$ and the size of the alphabet.

If $\boldsymbol{t}$ is a transducer then the following language class

$$\mathcal{P}_{\boldsymbol{t}} = \{L \mid L \text{ satisfies (1) for } R = \mathtt{R}(\boldsymbol{t})\}$$

is called the independence, or property, described by $\boldsymbol{t}$. In this case any language $L \in \mathcal{P}_{\boldsymbol{t}}$ is said to satisfy $\mathcal{P}_{\boldsymbol{t}}$. For example, the independence "prefix codes" is described by the transducer $\mathtt{px}$ and the independence "2-synchronization-error detecting languages" is described by the transducer $\mathtt{id}_2$ (see Fig. 1).

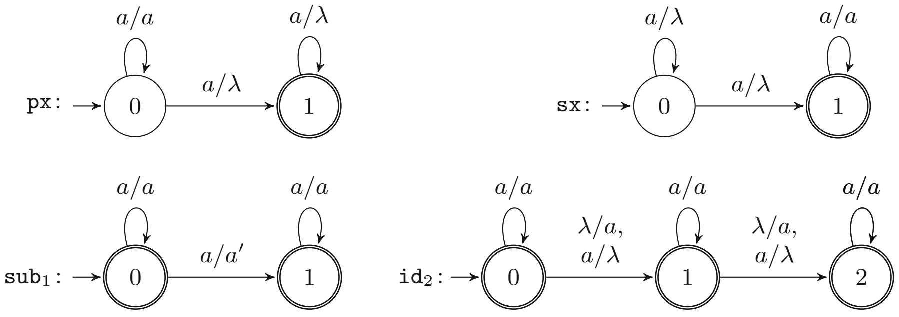

Fig. 1. Various transducers. An arrow with label a/a denotes multiple transitions: one with label a/a for each $a \in \Sigma$, and similarly for labels a/λ. An arrow with label a/a' denotes multiple transitions: one with label a/a' for all $a, a' \in \Sigma$ with $a \neq a'$. Let x be any word. We have: $\mathtt{px}(x)$ = the set of proper prefixes of x, equivalently, $\mathtt{R}(\mathtt{px})$ = the set of word pairs (x, y) such that y is a proper prefix of x; $\mathtt{sx}(x)$ = set of proper suffixes of x; $\mathtt{sub}_1(x)$ = set of words resulting by substituting at most one symbol in x with another one; $\mathtt{id}_2(x)$ = set of words resulting by inserting and/or deleting at most 2 symbols in x.

Transducer independences are closed under intersection, that is, if $\mathcal{P}_{\boldsymbol{t}_1}$ and $\mathcal{P}_{\boldsymbol{t}_2}$ are transducer independences, then also $\mathcal{P}_{\boldsymbol{t}_1} \cap \mathcal{P}_{\boldsymbol{t}_2}$ is a transducer independence.

[1] See [1,20], for instance, for transducer concepts.
[2] NFA = Nondeterministic Finite Automaton.

This is useful when we are interested in languages L satisfying two or more properties, such as the combined property of being a prefix and 1-substitution detecting code. As prefix codes are described by the transducer `px` and 1-substitution detecting codes are described by the transducer $\texttt{sub}_1$ then also the combined property is described by a transducer. This implies that LaSer can answer the satisfaction, maximality and construction questions for the combined property.

LaSer's backend is based on the Python package FAdo [8], which implements automata, transducers, and independences described by transducer objects in the module `codes.py` [17]. The choice to use FAdo is based on the facts that its installation is very simple, it contains a rich set of easy to use methods, and is written in Python which in turn provides a rich availability of high level methods.

3 Rational Independence Expressions

Three examples of independences that are not of the form $\mathcal{P}_{\boldsymbol{t}}$ are the following.

- The class of UD codes (uniquely decodable/decipherable codes). LaSer supports this class.
- The class of comma-free codes: that is, all languages L satisfying the equation

$$LL \cap \Sigma^+ L \Sigma^+ = \emptyset. \tag{2}$$

- The class of language pairs (L_1, L_2) satisfying the equation

$$L_1 \overset{\text{sdi}}{\leftarrow} L_2 = \emptyset. \tag{3}$$

  Here the site directed insertion operation $x \overset{\text{sdi}}{\leftarrow} y$ between words x, y, introduced in [3], is such that $z \in x \overset{\text{sdi}}{\leftarrow} y$ if $x = x_1uvx_2$, $y = uwv$, $z = x_1uwvx_2$, where u, v are nonempty. This operation models site-directed mutagenesis, an important technique for introducing a mutation into a DNA sequence.

That "UD codes" and "comma-free codes" are not transducer independences can be shown using dependence theory: every transducer independence is a 2-independence, but the "comma-free codes" independence, for instance, is a 3-independence and not a 2-independence—see [12,15]. In general, intersecting (combining) the above independences with a transducer independence results into a new independence that is not of the form $\mathcal{P}_{\boldsymbol{t}}$ for some transducer $\boldsymbol{t}$. LaSer does not handle non-transducer independences, with the exception of "UD codes" for which specific algorithms are employed. More specifically, for the satisfaction question LaSer uses the quadratic-time elegant algorithm of [9]. For the maximality question LaSer uses Schützenberger's theorem that maximality of L is equivalent to the condition that every word in Σ^* is subword (part) of some word of L^* [2]. A specific algorithm is also used in [23] for the satisfaction question of the combination of the UD code and two other DNA-related independences.

We now discuss possible approaches of representing non-transducer independences as finite objects in a way that these objects can be manipulated by algorithms which can answer questions about the independences being represented. Some approaches are discussed in [15,19]. It is important to note that there is a distinction between the terms "property" and "independence". A (language) property is simply a class (set) of languages. A (language) independence is a property $\mathcal{P}$ for which the concept of maximality is defined, that is, $\mathcal{P}$ must satisfy

if $L \in \mathcal{P}$ then also $L' \in \mathcal{P}$ for all $L' \subseteq L$

(see [15] for details). A general type of independences can be defined via rational language equations $\varphi(L) = \emptyset$, where $\varphi(L)$ is an independence expression involving the variable L. An independence expression φ is defined inductively as follows: it is L or a language constant, or one of $\varphi_1\varphi_2, \varphi_1 \cup \varphi_2, \varphi_1 \cap \varphi_2, (\varphi_1)^*, \boldsymbol{t}(\varphi_1), \theta(\varphi_1)$, where φ_1, φ_2 are independence expressions, $\boldsymbol{t}$ is a transducer constant, and θ is an antimorphic permutation[3] constant. A rational language equation is an expression of the form $\varphi(L) = \emptyset$, where $\varphi(L)$ is an independence expression containing the variable L and the language constants occurring in $\varphi(L)$ represent regular languages. A language L is φ-independent if it satisfies the equation $\varphi(L) = \emptyset$. The independence $\mathcal{P}_\varphi$ described by φ is the set of φ-independent languages.

Example 1. Every transducer independence $\mathcal{P}_{\boldsymbol{t}}$ such that $\boldsymbol{t}$ is an input-altering transducer[4] is described by the rational language equation

$$\boldsymbol{t}(L) \cap L = \emptyset, \tag{4}$$

where we have used the standard notation $\boldsymbol{t}(L) = \{y \mid (x,y) \in \mathtt{R}(\boldsymbol{t}),\ x \in L\}$. The independence "comma-free codes" is described by the rational language equation (2). The independence "θ-free languages", [10], is described by the rational language equation

$$LL \cap \Sigma^+\theta(L)\Sigma^+ \ = \ \emptyset.$$

□

The satisfaction question for independences $\mathcal{P}_\varphi$ is implemented in [19], where parsing of expressions $\varphi(L)$ is implemented using Python's `lark` library, and evaluation of $\varphi(L)$ for given $L = \mathtt{L}(\boldsymbol{a})$ is implemented using the FAdo package.

4 Further Independences

What about independences described by equations like (3)? This is a non-transducer independence. Various general types of independences are defined

[3] An antimorphic permutation θ maps the alphabet Σ onto Σ and extends to words anti-morphically: $\theta(xy) = \theta(y)\theta(x)$. A typical example of this is the DNA involution on the alphabet $\{\mathtt{a}, \mathtt{c}, \mathtt{g}, \mathtt{t}\}$ such that $\theta(\mathtt{a}) = \mathtt{t}$, $\theta(\mathtt{c}) = \mathtt{g}$, $\theta(\mathtt{g}) = \mathtt{c}$, $\theta(\mathtt{t}) = \mathtt{a}$. In this case, $\theta(\mathtt{aac}) = \mathtt{gtt}$.

[4] This is a transducer $\boldsymbol{t}$ such that $w \notin \boldsymbol{t}(w)$ for all words w. For example, `px` and `sx` in Fig. 1 are input-altering transducers.

in [5,11,12,22]. Equation (3) however involves an independence expression with ***two*** language variables: L_1 and L_2. In analogy to transducer independences that are R-independences, for binary relations R, one can consider independences with respect to higher degree relations[5]. Consider the ***ternary relation*** $\texttt{SDI} = \{(x,y,z) \mid z \in x \overset{\text{sdi}}{\leftarrow} y\}$, which is realized by the transducer `sdi` in Fig. 2. A language pair (L_1, L_2) is `SDI`-independent if

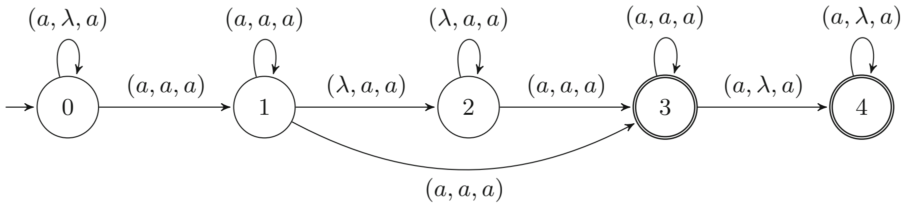

Fig. 2. The 3-tape transducer `sdi` realizing the set of all word triples (x,y,z) such that $x = x_1uvx_2$, $y = uwv$ and $z = x_1uwvx_2$, for some nonempty words u,v; that is, $z \in x \overset{\text{sdi}}{\leftarrow} y$.

$$\texttt{sdi}(L_1, L_2 : 3) \; = \; \emptyset.$$

Above we have made the following notation: for any k-tape transducer $\boldsymbol{t}$, for any $i \in \{1,\ldots,k\}$, and for any list of $k-1$ languages $L_1,\ldots,L_{k-1}$, the expression $\boldsymbol{t}(L_1,\ldots,L_{k-1} : i)$ denotes the set of all words w that result if we consider the i-th tape as output tape and the rest $k-1$ tapes as input tapes such that the input $k-1$ words are from the $k-1$ languages. Thus, we can talk about the independence described by the equation

$$\boldsymbol{t}(L_1,\ldots,L_{k-1} : i) \; = \; \emptyset.$$

Given a k-tape transducer $\boldsymbol{t}$ and $k-1$ NFAs accepting the languages $L_1,\ldots,L_{k-1}$, the satisfaction question can be decided if we construct the transducer resulting by intersecting $\boldsymbol{t}$ with the NFAs at the $k-1$ positions other than i, and then testing whether the resulting transducer has a path from an initial to a final state.

5 Looking Ahead

We propose to investigate further the rational language equations defined here as well as in [15]. Topics of interest are maximality, embedding and expressibility[6] as well as enhancement and implementation of algorithms involved. Some of

[5] See references [4,7], for instance, for higher degree relations.

[6] What independences are and are not describable by the independence-describing method.

these topics could be complex. For example, the maximality question for transducer independences can be answered by a simple algorithm, but the question is PSPACE hard. The embedding question is to construct a maximal independent language containing the given independent language $\mathtt{L}(\boldsymbol{a})$. For transducer independences described by Eq. (4), the embedding question is addressed in [16]. The maximality and embedding questions for non-transducer independences appear to be far more complex. For the satisfaction question of independences described by rational language equations, a useful question is to define and implement witnesses of non-satisfaction. For example, a witness of non-satisfaction for equation (2) would be a word triple (x, y, z) such that $x, y, z \in L$ and $xy = uzv$ for some nonempty words u and v.

Acknowledgement. We thank the anonymous referees for constructive comments.

References

1. Berstel, J.: Transductions and Context-Free Languages. B.G. Teubner, Stuttgart (1979)
2. Berstel, J., Perrin, D., Reutenauer, C.: Codes and Automata. Cambridge University Press, Cambridge (2009)
3. Cho, D.-J., Han, Y.-S., Salomaa, K., Smith, T.J.: Site-directed insertion: decision problems, maximality and minimality. In: Konstantinidis, S., Pighizzini, G. (eds.) DCFS 2018. LNCS, vol. 10952, pp. 49–61. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-94631-3_5
4. Choffrut, C.: Relations over words and logic: a chronology. Bull. EATCS **89**, 159–163 (2006)
5. Domaratzki, M.: Trajectory-based codes. Acta Informatica **40**, 491–527 (2004). https://doi.org/10.1007/s00236-004-0140-4
6. Domaratzki, M.: Bond-free DNA language classes. Nat. Comput. **6**, 371–402 (2007). https://doi.org/10.1007/s11047-006-9022-8
7. Elgot, C.C., Mezei, J.E.: On relations defined by generalized finite automata. IBM J. Res. Dev. **9**(1), 47–68 (1965). https://doi.org/10.1147/rd.91.0047
8. FAdo: Tools for formal languages manipulation. http://fado.dcc.fc.up.pt/. Accessed Jan 2020
9. Head, T., Weber, A.: Deciding code related properties by means of finite transducers. In: Capocelli, R., de Santis, A., Vaccaro, U. (eds.) Sequences II, Methods in Communication, Security, and Computer Science, pp. 260–272. Springer, Berlin (1993). https://doi.org/10.1007/978-1-4613-9323-8_19
10. Hussini, S., Kari, L., Konstantinidis, S.: Coding properties of DNA languages. Theoret. Comput. Sci. **290**, 1557–1579 (2003). https://doi.org/10.1016/S0304-3975(02)00069-5
11. Jürgensen, H.: Syntactic monoids of codes. Acta Cybern. **14**, 117–133 (1999)
12. Jürgensen, H., Konstantinidis, S.: Codes. In: Rozenberg, G., Salomaa, A. (eds.) Handbook of Formal Languages, vol. 1, pp. 511–607. Springer, Berlin (1997). https://doi.org/10.1007/978-3-642-59136-5_8
13. Kari, L., Konstantinidis, S., Kopecki, S.: Transducer descriptions of DNA code properties and undecidability of antimorphic problems. Inf. Comput. **259**(2), 237–258 (2018). https://doi.org/10.1016/j.ic.2017.09.004

14. Konstantinidis, S.: An algebra of discrete channels that involve combinations of three basic error types. Inf. Comput. **167**(2), 120–131 (2001). https://doi.org/10.1006/inco.2001.3035
15. Konstantinidis, S.: Applications of transducers in independent languages, word distances, codes. In: Pighizzini, G., Câmpeanu, C. (eds.) DCFS 2017. LNCS, vol. 10316, pp. 45–62. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-60252-3_4
16. Konstantinidis, S., Mastnak, M.: Embedding rationally independent languages into maximal ones. J. Automata Lang. Comb. **21**(4), 311–338 (2017). https://doi.org/10.25596/jalc-2016-311
17. Konstantinidis, S., Meijer, C., Moreira, N., Reis, R.: Symbolic manipulation of code properties. J. Automata Lang. Comb. **23**(1–3), 243–269 (2018). https://doi.org/10.25596/jalc-2018-243. (This is the full journal version of the paper "Implementation of Code Properties via Transducers" in the Proceedings of CIAA 2016, LNCS 9705, pp 1–13, edited by Yo-Sub Han and Kai Salomaa)
18. LaSer: Independent LAnguage SERver. http://laser.cs.smu.ca/independence/. Accessed Apr 2020
19. Rafuse, M.: Deciding rational property definitions, Honours Undergraduate Thesis, Department of Mathematics and Computing Science, Saint Mary's University, Halifax, NS, Canada (2019)
20. Sakarovitch, J.: Elements of Automata Theory. Cambridge University Press, Berlin (2009)
21. Shyr, H.J., Thierrin, G.: Codes and binary relations. In: Malliavin, M.P. (ed.) Séminaire d'Algèbre Paul Dubreil, Paris 1975–1976 (29ème Année). Lecture Notes in Mathematics, vol. 586, pp. 180–188. Springer, Heidelberg (1977). https://doi.org/10.1007/BFb0087133
22. Yu, S.S.: Languages and Codes. Tsang Hai Book Publishing, Taichung (2005)
23. Zaccagnino, R., Zizza, R., Zottoli, C.: Testing DNA code words properties of regular languages. Theoret. Comput. Sci. **608**, 84–97 (2015). https://doi.org/10.1016/j.tcs.2015.08.034