Consensual languages: a simple parallel machine model*

Stefano Crespi Reghizzi and Pierluigi San Pietro†

June 20, 2014

Abstract

This paper collects recent results on the consensually regular (C_{REG}) languages, a novel parallel abstract machine model. The model uses a finite automaton to define the sequential computations, and a simple matching rule to check that computations are in agreement. At any time, one computation acts as leader for the current transition, and the remaining computations must consent to the current character the leader has placed. Placing and consenting to the current character are encoded by doubling the alphabet with a marked copy. The recognizer of C_{REG} is a nondeterministic multi-set machine. After listing the main properties and language family comparisons, we show that regular languages coincide with the consensual languages based on strictly locally testable computations. A method for designing consensual languages so that their union is consensual is shown. Open problems and research directions are listed.

1 Introduction

After the introduction of the family of consensual languages in 2008 [2], the authors have investigated its properties in a series of works [3, 4, 7, 5, 9, 6, 15, 8, 10]. It is timely to present the main results in a more unified way, and, in particular, to revisit the properties singularly proved in earlier papers, as corollaries of the latest results.

Here, we introduce the consensual model by contrast with respect to the classical finite automata. The Deterministic Finite Automata (DFA) (and their extensions with unbounded memories) are the fundamental abstract model of sequential computations. On the other hand, parallel computations, though they are fundamental for modern computer applications, lack a real standard model, so that it is worthwhile to introduce and study new ideas that may lead to interesting models, to be added to the existing ones. Our consensual definitions enrich DFAs with parallel computations that "consensually" converge into a final configuration. Consider an input word, and recall how it is, respectively, recognized by a DFA, by a Nondeterministic FA (NFA), and by an Alternating FA (AFA). The DFA has just one accepting (i.e., reaching a final state) computation and zero rejecting ones, the NFA has one or more accepting and zero or more rejecting computations; and the AFA has one or more accepting and zero rejecting computations. In the model named Consensually Regular (C_{REG}), to recognize a word x of length n, the abstract machine has up to n accepting computations, such that, for each letter $x(1), \ldots, x(n)$, exactly one computation "places" the letter (i.e., asserts its validity) and the remaining computations consent to it. The dual behavior placing/consenting is implemented by a DFA that uses a double alphabet, made by the original alphabet and by a marked (dotted) copy. Such DFA defines a regular language over the double alphabet, called the base of the language consensually recognised. This device exceeds the power of finite-state computations, because the number of instantaneous configurations may grow linearly with the input length. Since a configuration is encoded by a multi-set of DFA states, the device is accordingly named a multi-set (consensual) machine and belongs to the class of so-called token machines, well-known through Petri Nets; but the firing rules of the two models are very different. Otherwise, we are not aware of any abstract machine model similar to consensual machines.

Thus a consensual computation involves tightly synchronized parallel processes, defined by the regular base language; at any time, one process leads the computation and the others must consent to the choice

*Work partially supported by *PRIN 2010LYA9RH-006* "Automi e linguaggi formali: Aspetti Matematici e Applicativi".

†The authors are with DEIB, Politecnico di Milano and CNR-IEIIT.

proposed by the leader. If leadership is restricted to a finite number of processes, the model degenerates to DFA. Otherwise the model is a logspace nondeterministic multi-counter machine, which is worth comparing with other types of automata.

2 The Consensual Model

We start with an intuitive example to show how the consensual model makes use of a FA to perform parallel coordinated computations which reinforce each other; then we formalize the model.

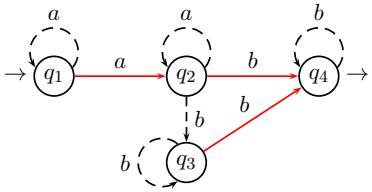
Consider the NFA A of Fig. 1, where some of the transitions are marked as dashed, and the remaining solid transitions are colored in red to highlight the distinction. There is a constraint on this marking: for each letter a and for each state q

at most one dashed a-labeled transition and at most one solid a-labeled transition exit from q. (1)

A solid transition $q_i \xrightarrow{a} q_j$ is called *placing* (it *places* letter a); a dashed transition $q_i \overset{a}{\dashrightarrow} q_j$ is called *consenting* (it *consents* to a). We describe the behavior of a non-deterministic, parallel machine M_A, featuring a set of synchronized sequential processes, each one being a finite-state computation of A. Thus, each computation starts in the initial state and is in one of the states of A. M_A is called a *multi-set machine*, since its configuration is a multi-set of states of A. Given an input $x \in \Sigma^+$, M_A starts up to $|x|$ parallel computations, each in the initial state q_1. At each step, M_A reads the current input symbol a and makes each sequential computation move (in parallel) to the next state, computed by the transition relation of A, for the present state and input symbol. In other words, each (sequential) computation nondeterministically chooses an a-labeled transition from its current state. This choice must meet at each step the following global "matching constraint":

one computation performs a placing transition, and **all others** perform consenting transitions. (2)

These computations are called *matching*. M_A accepts when x has been consumed and each computation is in a final state of A. The language accepted by the multi-set machine is called the *consensual language* defined by A.



(a) FA A with dashed (consenting) transitions and solid (placing) ones.

$c_1 : q_1 \xrightarrow{a} q_2 \overset{a}{\dashrightarrow} q_2 \overset{a}{\dashrightarrow} q_2 \xrightarrow{b} q_4 \overset{b}{\dashrightarrow} q_4 \overset{b}{\dashrightarrow} q_4$

$c_2 : q_1 \overset{a}{\dashrightarrow} q_1 \xrightarrow{a} q_2 \overset{a}{\dashrightarrow} q_2 \overset{b}{\dashrightarrow} q_3 \overset{b}{\dashrightarrow} q_3 \xrightarrow{b} q_4$

$c_3 : q_1 \overset{a}{\dashrightarrow} q_1 \overset{a}{\dashrightarrow} q_1 \xrightarrow{a} q_2 \overset{b}{\dashrightarrow} q_3 \xrightarrow{b} q_4 \overset{b}{\dashrightarrow} q_4$

(b) Three matching computations for input $aaabbb$.

Figure 1: A multi-set machine M_A. M_A accepts word a^3b^3 because (i) A admits the matching computations of Fig. 1(b), and (ii) at every step, there is exactly one solid (i.e., placing) transition. Therefore the computations are matching and define an accepting run of machine M_A.

Rather than using dashed and solid or colored transitions, it is preferable to identify a consenting transition by marking with a dot its letter, using the"dotted" copy $\mathring{a}, \mathring{b}, \ldots$ of the alphabet. This allows a more abstract definition based on the language recognized by A, instead of the graph of A. Given a finite alphabet Σ, let $\mathring{\Sigma}$ be the *dotted* (or marked) copy of Σ. The *double alphabet* is $\widetilde{\Sigma} = \mathring{\Sigma} \cup \Sigma$.

We reformulate the preceding example using the double alphabet. Let $B_A = \mathring{a}^* a \mathring{a}^* \mathring{b}^* b \mathring{b}^*$. This kind of expression is called a "consensual regular expression". It is easy to see that B_A is the language recognized by the above automaton A, modified by dotting the labels of dashed transitions. The modified version of A is now a DFA, since A meets constraint (1). Therefore, B_A can be regarded as the set of all possible (sequential) computations of M_A, going from the initial to the final state. For instance, computation c_1 above is represented by the word $a\mathring{a}\mathring{a}b\mathring{b}\mathring{b}$, c_2 by $\mathring{a}a\mathring{a}\mathring{b}\mathring{b}b$, and c_3 by $\mathring{a}\mathring{a}a\mathring{b}\mathring{b}b$.

To formalize the matching computations of a multi-set machine, we introduce a partial, symmetrical, and associative binary operator $@ : \widetilde{\Sigma} \times \widetilde{\Sigma} \to \widetilde{\Sigma}$, called *match*: defined as follows, for all $a \in \Sigma$:

$$a@\mathring{a} = \mathring{a}@a = a, \qquad \mathring{a}@\mathring{a} = \mathring{a}, \qquad @ \text{ is undefined in every other case.}$$

The match is extended to words of equal length as a letter-by-letter application, by assuming $\epsilon@\epsilon = \epsilon$: for every $n > 1$, for all $w, w' \in \widetilde{\Sigma}^n$, if, for all positions $i, 1 \leq i \leq n$, $w(i)@w'(i)$ is defined, then:

$$w \mathbin{@} w' = \big(w(1)@w'(1)\big) \cdot \ldots \cdot \big(w(n)@w'(n)\big). \quad \text{In every other case, } w@w' \text{ is undefined.}$$

Hence, the match is undefined on words w, w' of unequal lengths, which cannot in fact represent two parallel computations in a multi-set machine. Moreover, $w@w'$ is undefined even when $|w| = |w'|$, if there exists a position j such that $w(j)@w'(j)$ is undefined, which occurs in three cases: when both letters are in Σ, corresponding to the situation when two computations try to place a letter at the same time; when both characters are in $\mathring{\Sigma}$ and differ, i.e., two computations consent to different letters at the same time; when either character is dotted but it is not the dotted copy of the other, i.e., one computation is placing a letter and the other is not consenting to it.
The match is extended to two languages B', B'' on the double alphabet, as $B' \mathbin{@} B'' = \{w' \mathbin{@} w'' \mid w' \in B', w'' \in B''\}$. The iterated match $B^{i@}$ is defined for all $i \geq 0$, as $B^{0@} = B$, $B^{i@} = B^{(i-1)@}@B$, if $i > 0$. The *closure under match*, or *@-closure*, of a language $B \subseteq \widetilde{\Sigma}^*$ is $B^{@} = \bigcup_{i \geq 0} B^{i@}$.
By definition, if $w \in B_A^{@}$ and $w \in \Sigma^*$, then $w = w_1@w_2@\ldots@w_m$ for $m \geq 1$ words $w_1, \ldots, w_m \in B$, such that in each position $1 \leq i \leq |w|$, exactly one word, say w_h, is undotted, i.e., $w_h(i) \in \Sigma$. Therefore, the corresponding computation places $w_h(i)$, and $w_j(i) \in \mathring{\Sigma}$ for all $j \neq h$, i.e., the corresponding computations consent to $w_h(i)$. Given input $w \in \Sigma^*$, a multi-set machine, defined on a FA recognizing a language $B \subseteq \widetilde{\Sigma}^*$, accepts w if, and only if, there exist $m \leq |w|$ computations $w_1, \ldots, w_h \in B$ such that $w = w_1@w_2@\ldots@w_m$. We say that m is the match *degree*. These considerations lead to the following definition of consensual languages.

Definition 1. The *consensual language with base* B is: $\mathcal{C}(B) = B^{@} \cap \Sigma^*$. The family of *consensually regular* languages, denoted by C_{REG}, is the collection of all languages $\mathcal{C}(B)$, such that B is regular.

Hence, a C_{REG} language is *consensually specified* by a regular expression over $\widetilde{\Sigma}$. For instance, language $B_A = \mathring{a}^*a\mathring{a}^*\mathring{b}^*b\mathring{b}^*$ above defines the nonregular language $\mathcal{C}(B_A) = \{a^nb^n \mid n > 0\}$.

Example 1. The language $L_1 = \{a^nb^nc^n \mid n > 0\}$ is consensually specified by the base $B_1 = \mathring{a}^*a\,\mathring{a}^*\mathring{b}^*b\,\mathring{b}^*\mathring{c}^*c\,\mathring{c}^*$. For instance, $aabbcc$ is the (strong) match of $\mathring{a}\,a\,\mathring{b}\,b\,\mathring{c}\,c$ and $a\,\mathring{a}\,b\,\mathring{b}\,c\,\mathring{c}$.
The commutative language $L_2 = com\big((abb)^+\big)$, where com stands for the commutative closure, is specified by the base $B_2 = com\big(abb\big) \sqcup\!\sqcup \mathring{\Sigma}^*$, where $\sqcup\!\sqcup$ is the shuffle operator.
The non-semilinear language $L_3 = \{ba^1ba^2ba^3\ldots ba^k \mid k \geq 1\}$ is consensually specified by the base $\left(\mathring{a} \cup \mathring{b}\right)^*\ b\,\mathring{a}^*\,a\,\mathring{a}^*\left(\mathring{b}\,\mathring{a}^*\,a\,\mathring{a}^*\right)^+$.

Since the definition of match and match closure apply to any language, irrespectively of it being regular or not, it is meaningful to consider the consensual language defined by, say, a context-free base. For an arbitrary language family $\mathcal{F}$, $C_{\mathcal{F}}$ denotes the collection of all languages $\mathcal{C}(B)$, such that $B \in \mathcal{F}$. Some results for $C_{\textit{context-free}}$ and $C_{\textit{context-sensitive}}$ are in [9] but are not discussed here.

Summary of known C_{REG} properties. *Language family comparisons*: C_{REG} is incomparable with the context-free and deterministic context-free families, is included within the context-sensitive family and contains non-semilinear languages [4]. C_{REG} strictly includes the family of languages accepted by partially-blind multi-counter machines that are deterministic and quasi-real-time, as well as their union [8]; moreover, it includes the closure under union and concatenation of commutative semilinear languages [10].
Closure properties: C_{REG} is is closed under marked concatenation, marked iteration, inverse alphabetic

homomorphism, reversal, and intersection and union with regular languages [4]. The marked concatenation of two languages $L_1, L_2 \subseteq \Sigma^*$ is the language $L_1 \# L_2$, where $\# \notin \Sigma$, while the marked iteration of $L \subseteq \Sigma^*$ is the language $(L\#)^*$. A language family enjoying such properties is known as a *pre-Abstract Family of Languages* (see, e.g., [14]). A precise characterization of the bases that consensually specify regular languages is in [7]; an analysis of the reduction in descriptional complexity of the consensual base with respect to the specified regular language is in [4].
Complexity: C_{REG} is in NLOGSPACE, i.e., it can be recognized by a nondeterministic multitape Turing machine working in $\log n$ space. The recognizer of C_{REG} languages is the multi-set machine informally described above, which is a special kind of nondeterministic, real-time multi-counter machine [4].
More properties will be listed as we proceed.
An open problem is whether all C_{REG} languages over a unary alphabet are regular.

2.1 Regular Languages Coincide with Consensual Strictly-Locally-Testable Languages

When $\mathcal{C}(B)$ is regular? As said, if there exists a finite H such that every sentence of $\mathcal{C}(B)$ has match degree $h \leq H$, then $\mathcal{C}(B)$ is regular because the multi-sets in the consensual machine are bounded. A less evident condition for regularity is that the match degree of every sentence $w \in \mathcal{C}(B)$ is maximal, i.e., equal to $|w|$. In this case it is possible to construct a NFA simulating the multi-set machine [4].
A surprising deep result [9] is

Theorem 1. $REG = C_{SLT}$ *, where* SLT *is the family of* Strictly Locally Testable *languages [11].*

Family k-SLT is the subfamily of star-free (or noncounting) regular languages that is recognized by a DFA using a sliding window of width $k \geq 1$ (e.g., [1]); a language is SLT if it is k-SLT for some k. It is well-known that a word of length $\geq k$ belongs to a k-SLT language iff its k-prefix (resp. suffix) belongs to the set of legal k-prefixes (resp. suffixes), and every k-factor belongs to the set of legal k-factors.
The proofs of the two inclusions stated in Th. 1 are in [9] and we give just a short hint.
The proof of $REG \supseteq C_{SLT}$ considers the k-local DFA [1], A, recognizing B. Each state p of A is uniquely identified by the suffix of length k of all words leading into p (the initial state and its close neighbors are identified by ε and by shorter words). The transition relation of the multi-set machine for such a k-local DFA leads to multi-sets that are bounded, therefore the consensual language is regular.
In the other direction, the proof of

Lemma 1. $REG \subseteq C_{SLT}$

is more involved and required to extend the classical homomorphic characterization of regular languages (Medvedev Theor. [12] also in [13]), stating that every regular language over Σ is the alphabetic homomorphism of a 2-SLT language over a (much) larger alphabet Δ. We first generalized Medvedev Theor. [5, 6], in order to reduce the size of alphabet Δ by increasing the value of parameter k beyond 2. To prove Lm. 1 we only need the following simpler statement.

Lemma 2. *Let* $L \subseteq \Sigma^*$ *be recognized by an NFA having* n *states. Then* L *is the alphabetic homomorphism of a* k*-SLT language over an alphabet* Δ *of size* $|\Delta| = 2 \cdot |\Sigma|$ *where* k *is in* $O(\lg n)$. ([6] proves that the ratio $|\Delta|/|\Sigma|$ cannot go under 2.)

Given a regular language R over Σ, by Lm. 2 we can find a a k-SLT language B' over $\widetilde{\Sigma}$ such that R is the homomorphic image of B'. Let *switch* : $\widetilde{\Sigma} \to \widetilde{\Sigma}$ be the mapping such that *switch*$(a) = \mathring{a}$, *switch*$(\mathring{a}) = a,$ for all $a \in \Sigma$. Then $B'' =$ *switch*(B') is clearly in k-SLT. Let $B = B' \cup B''$. A technical difficulty, addressed in [9], is that the three sets of legal k-prefixes—suffixes—factors for B' may be non-disjoint from the corresponding sets for B'', thus possibly causing over generalization when the sets are orderly united. By taking in Lm 2 a larger value of k and a so-called factor-decodable duality-free encoding of the NFA states, we ensure that the sets associated to B' and B'' are disjoint and their union thus defines exactly B. Since for every word w' in B the corresponding w'' in B'' strongly matches, $\mathcal{C}(B) = R$.
Open problem: if the base B is a star-free (non-counting) language, $\mathcal{C}(B)$ is generally non-regular, e.g., see Fig. 1a. Is the inclusion $C_{star-free} \subseteq C_{REG}$ strict?

2.2 Compositional Method for Union and Concatenation

As already discusses, few (non)closure properties of C_{REG} are known, and the status of the closure under the basic operations is uncertain. In particular, union and concatenation closures have neither proofs nor candidate counterexamples. To tackle the problem, we have investigated some "rich" subfamilies of C_{REG}, proving that their closure under union and concatenation remains in C_{REG}. This approach will hopefully give suggestions for the general case of C_{REG}. In the remainder, we illustrate our methods for the case of union; for concatenation, the similar but more complex discussion is in [10].

Example 2. The language $L'' = \{a^n b^{2n} \mid n > 0\}$ is consensually specified by the base $B'' = \mathring{a}^* a\, \mathring{a}^* \mathring{b}^* b\, \mathring{b}^* \mathring{b}^* b\, \mathring{b}^*$. For instance, $aabbbb$ is the match of $\mathring{a}\, a\, \mathring{b}\, b\, \mathring{b}\, b$ and $a\, \mathring{a}\, b\, \mathring{b}\, b\, \mathring{b}$.

Consider now again base $B_A = \mathring{a}^* a \mathring{a}^* \mathring{b}^* b \mathring{b}^*$ and let $L' = \mathcal{C}(B_A) = \{a^n b^n \mid n > 0\}$. The language $L' \cup L''$ is in C_{REG}, but, counter to a naive intuition, it is not specified by the base $B_A \cup B''$. In general, $\mathcal{C}(B_1 \cup B_2) \supset \mathcal{C}(B_1) \cup \mathcal{C}(B_2)$; in the example, $\mathcal{C}(B_A \cup B'')$ contains also undesirable "cross-matching" words, such as $aabbb = a\mathring{a}b\mathring{b}\mathring{b} \,@\, \mathring{a}a\mathring{b}bb$.

We first introduce a normal form, named decomposed, of the base languages. Second, we state a condition, named joinability, for decomposed forms, that guarantees closure under union. This results hold for every consensual language, but the difficulty remains to find a systematic method for constructing base languages that meet such conditions. Third, we introduce an implementation of decomposed joinable forms, relying on numerical congruences. Let B, B' be languages included in $\tilde{\Sigma}^+ - \mathring{\Sigma}^+$. We say that B is *unproductive* if $\mathcal{C}(B) = \emptyset$, and that the pair (B, B') is *unmatchable* if $B@B' = \emptyset$.

Definition 2. A base B is in *decomposed form* if it can be partitioned into two languages, named the *scaffold* sc and the *fill* fl of B, such that fl is unproductive, and the pair (sc, sc) is unmatchable.

The name "scaffold" conveys the idea of an arrangement superposed just once on each word of the base; the name "fill" suggests an optional repeatable component to fill-in the letters which are dotted in the scaffold. Three straightforward remarks follow. For every base B there exists a consensually equivalent decomposed base: it suffices to take as scaffold the language $\{a\, dot(y) \mid ay \in B, a \in \Sigma, y \in \tilde{\Sigma}^*\}$, and as fill the language $\{dot(x)y \mid x \in \widetilde{\Sigma}, y \in \widetilde{\Sigma}^*, xy \in B\}$, where $dot : \Sigma \to \mathring{\Sigma}$ is the mapping $dot(a) = \mathring{a}$ for all $a \in \Sigma$. For every $s \subseteq sc$, $f \subseteq fl$, the base $s \cup f$ is a decomposed form. The scaffold, but not the fill, may include words over Σ. Consider a word $w \in \mathcal{C}(B)$. Since the fill is unproductive, its match closure leaves some letters of w dotted, which must be *placed* by the scaffold. Since by definition the match closure of the scaffold alone is the scaffold itself, the following lemma holds.

Lemma 3. *If $B = sc \cup fl$ is in decomposed form, as in Def. 2, then $\mathcal{C}(B) = \left(sc \cup (sc \,@\, fl^{@})\right) \cap \Sigma^*$.*

Example 3. The table shows the decomposed bases of languages L' and L'' of Ex. 2, restricted for brevity to the case that the number of a's is a multiple of 3. Let $L' = com\left(\{a^{3n}b^{3n} \mid n \geq 1\}\right)$, with scaffold sc' and fill fl', and $L'' = com\left(\{a^{3n}b^{6n} \mid n \geq 1\}\right)$, with scaffold sc'' and fill fl''.

	scaffold	*fill*	*a strong match*
L'	$(a\mathring{a}a)^+\ (b\mathring{b}b)^+$	$(\mathring{a}^3)^*\, \mathring{a}a\mathring{a}\, (\mathring{a}^3)^*\ (\mathring{b}^3)^*\, \mathring{b}b\mathring{b}\, (\mathring{b}^3)^*$	$a\ \mathring{a}\ a\ b\ \mathring{b}\ b \in sc'$ $@\ \mathring{a}\ a\ \mathring{a}\ \mathring{b}\ b\ \mathring{b} \in fl'$
L''	$(\mathring{a}aa)^+\ (\mathring{b}bb)^+$	$(\mathring{a}^3)^*\, a\mathring{a}\mathring{a}\, (\mathring{a}^3)^*\ (\mathring{b}^3)^*\, (b\mathring{b}\mathring{b})^2\, (\mathring{b}^3)^*$	$\mathring{a}\ a\ a\ \mathring{b}\ b\ b\ \mathring{b}\ b\ b \in sc'$ $@\ a\ \mathring{a}\ \mathring{a}\ b\ \mathring{b}\ \mathring{b}\ b\ \mathring{b}\ \mathring{b} \in fl'$

Clearly, every word in sc' is unmatchable with every other word in sc', hence $sc'@sc' = \emptyset$. Similarly, every fill is unproductive. Every word in L' is the match of exactly one word in the scaffold with one or more words in the fill. Analogous remarks hold for L''.

Next, consider two decomposed bases $B' = sc' \cup fl'$ and $B'' = sc'' \cup fl''$. By imposing further conditions on the bases, we obtain a very useful theorem about composition of the consensual languages by union.

Definition 3. Two bases B', B'' in decomposed form are *joinable* if their union $B' \cup B''$ is decomposed, with scaffold $sc' \cup sc''$ and fill $fl' \cup fl''$, and the pairs (sc', fl'') and (sc'', fl') are unmatchable.

Theorem 2. *If two bases B' and B'' are joinable then $\mathcal{C}(B') \cup \mathcal{C}(B'') = \mathcal{C}\,(B' \cup B'')$.*

Example 4. For Ex. 3, we check that both bases are joinable. The union of the bases is in decomposed form: $fl' \cup fl''$ is unproductive (because letters at positions 3, 6, ... cannot be placed); the pair (sc', sc'') is unmatchable, hence also $(sc' \cup sc'', sc' \cup sc'')$ is unmatchable. Moreover, (sc', fl''), and (sc'', fl') are unmatchable. Therefore $L' \cup L'' = \mathcal{C}(sc' \cup sc'' \cup fl' \cup fl'')$.

A Decomposed Form Relying on Congruences We now design a decomposed form, suitable for supporting joinability, that uses module arithmetic for assigning the positions to the dotted and undotted letters within a word w over $\widetilde{\Sigma}$; the preceding examples offered some intuition for this formal development. Loosely speaking, each decomposed base language is "personalized" by a sort of unique pattern of dotted/undotted letters, such that, when we want to unite or concatenate two languages, the match of two words with different patterns is undefined, thus ensuring that the union or catenation of the two decomposed bases specifies the intended language composition.

For every $a \in \Sigma$, consider the projection of w on $\widetilde{a} = \{a, \mathring{a}\}$, denoted by $\pi_{\widetilde{a}}(x)$ and, in there, the numbered positions of each a and $\mathring{a}$. Let m be an integer. By prescribing that for each base language, each undotted letter a may only occur in positions j characterized by a specified value of the congruence $j \mod m$, we make the bases decomposed.

Definition 4 (Slots and modules)**.** Let $m > 3$, called *module*, be an even number. Let $R \subseteq \{1, \ldots, (m/2-1)\}$ be a nonempty set, called a *set of slots of module* m. For every $a \in \Sigma$, define a finite language $R_m(a) \subset \tilde{a}^m$, where only positions 1 and $r+1$ are dotted: $R_m(a) = \{\mathring{a}\, a^{r-1} \mathring{a}\, a^{m-r-1} \mid r \in R\}$. The disjoint regular languages *sc*-R_m, *fl*-R_m are defined as:

$$\textit{sc}\text{-}R_m = \{x \mid \forall a \in \Sigma, \pi_{\widetilde{a}}(x) \in (R_m(a) \cup a)^*\} \qquad \textit{fl}\text{-}R_m = \textit{switch}(\textit{sc}\text{-}R_m) - \mathring{\Sigma}^*.$$

It is fairly obvious that $\mathcal{C}(B) = \Sigma^+$, since $\Sigma^+ \subseteq$ *sc*-R_m. Also, *sc*-R_m@*sc*-$R_m = \emptyset$ and *fl*-R_m is unproductive. The following lemma is also obvious.

Lemma 4. *For all even numbers $m > 3$ and non-empty sets R of slots of module m, every base $E \subseteq$ sc-$R_m \cup$ fl-R_m is in decomposed form, with scaffold: $E \cap$ sc-R_m and fill: $E \cap$ fl-R_m.*

Example 5. Let $m = 6, R = \{1, 2\}$ and $\Sigma = \{a, b\}$. Then $R_6(a) = \{\mathring{a}\mathring{a}aaaa, \mathring{a}a\mathring{a}aaa\}$

$$\begin{aligned} \textit{sc}\text{-}R_6 &= (\mathring{a}\mathring{a}aaaa \cup \mathring{a}a\mathring{a}aaa \cup a)^* \sqcup\!\sqcup (\mathring{b}\mathring{b}bbbb \cup \mathring{b}b\mathring{b}bbb \cup b)^* \\ \textit{fl}\text{-}R_6 &= \Big((aa\mathring{a}\mathring{a}\mathring{a}\mathring{a} \cup a\mathring{a}a\mathring{a}\mathring{a}\mathring{a} \cup \mathring{a})^* \sqcup\!\sqcup\ (bb\mathring{b}\mathring{b}\mathring{b}\mathring{b} \cup b\mathring{b}b\mathring{b}\mathring{b}\mathring{b} \cup \mathring{b})^*\Big) - \{\mathring{a}, \mathring{b}\}^* \end{aligned}$$

By taking disjoint sets of slots over the same module, we obtain two bases that are joinable:

Theorem 3. *Let $m > 3$ and let R', R'' be two disjoint sets of slots of module m, and let $E' \subseteq$ sc-$R'_m \cup$ fl-R'_m and $E'' \subseteq$ sc-$R''_m \cup$ fl-R''_m be two bases. Then E' and E'' are joinable.*

Th. 3 is used to show that the closure under union of a subfamily $\mathcal{F}$ of C_{REG} is still in C_{REG}. First, one needs to prove that, for every language $L \in \mathcal{F}$, there exists $m > 3$ such that, for all $m' \geq m, 1 < r < m'$, L can be specified by a decomposed base with module m' and set of slots $\{r\}$. Consider the union of two or more languages in $\mathcal{F}$. Select the largest module and a different value of r, i.e., disjoint sets of slots, for their decomposed bases. By Th. 3, the union of such bases is still a joinable base. By Th. 2, the result is still a consensual language.

The above compositional method has been so far applied to two C_{REG} subfamilies: the deterministic partially blind multi-counter machines [8] and the commutative languages having semilinear Parikh image [10], but more cases are under way. The closure of C_{REG} under marked union and concatenation, proved with an ad hoc method in [4], is an immediate corollary of the compositional method.

Research Directions As said, most closure properties are still unknown for C_{REG}. The questions of deterministic versus nondeterministic behavior, of ambiguity, and of computational complexity are almost unexplored. The precise relations between consensual multi-set machines and the several existing families of multi-counter machines have been investigated to a very limited extent.

For each language family $\mathcal{F}$, it holds $C_{\mathcal{F}} \supseteq \mathcal{F}$, meaning that the consensual use boosts a language family (but $C_{\text{context-sensitive}} =$ context-sensitive). A generic question for any family $\mathcal{F}$ is: what family $\mathcal{B}$ exactly yields $\mathcal{F}$, i.e., $C_{\mathcal{B}} = \mathcal{F}$? For $\mathcal{F} = REG$ we know the answer is $\mathcal{B} = SLT$.

References

[1] P. Caron. Families of locally testable languages. *Theor. Comp. Sci.*, 242(1-2):361–376, 2000.

[2] Stefano Crespi Reghizzi and Pierluigi San Pietro. Consensual definition of languages by regular sets. In Carlos Martín-Vide, Friedrich Otto, and Henning Fernau, editors, *Language and Automata Theory and Applications, LATA 2008*, volume 5196 of *LNCS*, pages 196–208. Springer, 2008.

[3] Stefano Crespi Reghizzi and Pierluigi San Pietro. Languages defined by consensual computations. In Alessandra Cherubini, Mario Coppo, and Giuseppe Persiano, editors, *Theoretical Computer Science, 11th Italian Conference, ICTCS 2009, Cremona, Italy, September 28-30, 2009, Proceedings*, pages 82–85, 2009.

[4] Stefano Crespi Reghizzi and Pierluigi San Pietro. Consensual languages and matching finite-state computations. *RAIRO - Theor. Inf. and Applic*, 45(1):77–97, 2011.

[5] Stefano Crespi Reghizzi and Pierluigi San Pietro. From regular to strictly locally testable languages. In Petr Ambroz, Stepan Holub, and Zuzana Masáková, editors, *Proc. 8th Int. Conf. WORDS 2011*, volume 63 of *EPTCS*, pages 103–111, 2011.

[6] Stefano Crespi Reghizzi and Pierluigi San Pietro. From regular to strictly locally testable languages. *International Journal of Foundations of Computer Science*, 23(08):1711–1727, 2012.

[7] Stefano Crespi Reghizzi and Pierluigi San Pietro. Strict local testability with consensus equals regularity. In Nelma Moreira and Rogério Reis, editors, *17th International Conference CIAA 2012, Porto, Portugal, July 17-20, 2012. Proceedings*, volume 7381 of *Lecture Notes in Computer Science*, pages 113–124. Springer, 2012.

[8] Stefano Crespi Reghizzi and Pierluigi San Pietro. Deterministic counter machines and parallel matching computations. In Stavros Konstantinidis, editor, *Impl. and Appl. of Automata - 18th Int. Conf., CIAA 2013, Halifax, Nova Scotia, Canada, July 16-19, 2013.*, volume 7982 of *Lecture Notes in Computer Science*, pages 280–291. Springer, 2013.

[9] Stefano Crespi Reghizzi and Pierluigi San Pietro. Strict local testability with consensus equals regularity, and other properties. *Int. J. Found. Comput. Sci*, 24(6):747–764, 2013.

[10] Stefano Crespi Reghizzi and Pierluigi San Pietro. Commutative languages and their composition by consensual methods. In *Proc. 14th Int. Conference on Automata and Formal Languages, AFL 2014, Szeged, Hungary, May 27-29, 2014*, volume 151 of *Electronic Proceedings in Theoretical Computer Science*, pages 216–230. Open Publishing Association, 2014.

[11] R. McNaughton and S. Papert. *Counter-free Automata*. MIT Press, Cambridge, USA, 1971.

[12] Y. T. Medvedev. On the class of events representable in a finite automaton. In E. F. Moore, editor, *Sequential machines – Selected papers (translated from Russian)*, pages 215–227. Addison-Wesley, New York, NY, USA, 1964.

[13] S. Eilenberg. *Automata, Languages, and Machines*. Academic Press, 1974.

[14] Arto Salomaa. *Formal languages*. Academic Press, San Diego, CA, USA, 1987.

[15] Stefano Crespi Reghizzi and Pierluigi San Pietro. Commutative consensual counter languages. *Talk given at ICTCS 2013,*, 14th Italian Conference on Theoretical Computer Science, Palermo, Italia, Sept. 9-11, 2013., 2013.