*Journal of Universal Computer Science, vol. 16, no. 5 (2010), 821-832*
*submitted: 22/1/09, accepted: 12/2/10, appeared: 1/3/10 © J.UCS*

# Ordered Catenation Closures and Decompositions of Languages Related to a Language of Derick Wood

**Arto Salomaa**
(Turku Centre for Computer Science
Joukahaisenkatu 3–5 B, 20520 Turku, Finland
asalomaa@utu.fi)

**Abstract:** We investigate the problem of decomposing a language into a catenation of nontrivial languages, none of which can be decomposed further. In many cases this leads to the operation of an *ordered catenation closure*, introduced in this paper. We study properties of this operation, as well as its iterations. Special emphasis is on laid on ordered catenation closures of finite languages. It is also shown that if an infinite language is a code or a length code, then its ordered catenation closure does not possess a finite decomposition of indecomposable factors.

**Key Words:** decomposition of languages, indecomposable language, finite language, code, length code, ordered catenation closure
**Category:** F.4.3, F.1.1

## 1 Introduction

The *catenation* or *product* of languages is an operation widely studied but still many of the basic problems remain open. For instance, the conditions for commutativity $XY = YX$ are understood if $X$ and $Y$ are words but are still in many cases open when they are languages.

Languages can be represented as products in many different ways. Recently there has been much interest in representations, where the factors cannot be further decomposed, [Mateescu et al. 2002, Avgustinovich and Frid 2005] [Czyzowicz et al. 2003, Han et al., Han and Wood 2005, Salomaa and Yu 2000]. Such *prime decompositions* constitute the main topic of this paper.

Our starting point is the language introduced and investigated in [Han et al. 2006]. The language, denoted by $L_D$ in the sequel, turns out to be particularly interesting from the point of view of decomposition theory. The language $L_D$ cannot have a prime decomposition. By analyzing the properties causing this, we come to a class of languages with the same properties. This leads also to a special variant of the star operation, the notion of the *ordered catenation closure of a language* $L$, in symbols, $L^\diamond$. We believe that his notion is of interest also on its own right.

A brief outline about the contents of the paper follows. The next section gives the basic definitions and also some earlier results relevant to our considerations. It also discusses some fundamental facts about *infinite products*. Section

3 introduces the language $L_D$ and discusses its basic properties. It also introduces the *diamond* operation $L^\diamond$ and gives some arguments connecting it to $L_D$ and related languages. In Section 4 we prove, for some classes of languages $L$, the language $L^\diamond$ can never possess a prime decomposition. The main section of this paper, Section 5, is a study of the diamond operation and its compositions. The operations are compared with the star operation, with special focus on finite languages and state complexities. The final section describes some open problems.

## 2 Decompositions of languages and primality

We assume familiarity with the basics of formal languages. Small letters from the beginning of the English alphabet $a, b, c, d$, possibly with indices, denote letters of our formal alphabet $\Sigma$. Words are denoted by small letters from the end of the English alphabet. The empty word is denoted by $\lambda$. Following the regular expression notation, we sometimes denote the union by "+" and singleton sets $\{x\}$ simply by $x$. Thus, $\lambda + ab$ stands for the set $\{\lambda\} \cup \{ab\}$. The following definition, [Mateescu et al. 2002], contains the basic notions of this paper.

**Definition 1** *A nonempty language $L$ has a* nontrivial decomposition *if, for some $L_1 \neq \{\lambda\}$ and $L_2 \neq \{\lambda\}$, we have $L = L_1L_2$. A nonempty language $L \neq \{\lambda\}$ having no nontrivial decomposition is* prime*. A language $L$ has a* prime decomposition *if*

$$L = L_1 \dots L_m,\ m \geq 1,$$

*where each of the languages $L_i,\ 1 \leq i \leq m,$ is prime.*

Observe that by a *"prime decomposition"* without further specifications we always mean a *finite* decomposition, that is, the number of prime factors is finite. We consider below also products with infinitely many factors, prime or nonprime.

One can also visualize different ways of defining a "prime" language. For instance, in [Frid 2007] a language $L$ is termed *indecomposable* if the equation $L = L_1L_2$ implies that either $L = L_1$ or $L = L_2$. It is clear that if a language is prime in the sense of Definition 1, then it is also indecomposable. On the other hand, languages $\phi$ and $\{\varepsilon\}$ are indecomposable but not prime. These are the only exceptions, the two notions coincide otherwise, [Salomaa et al.].

It is obvious that every finite language has a prime factorization. It is not necessarily unique, for instance,

$$(\lambda + a^2 + a^3 + a^4)(\lambda + a^3 + a^4) = (\lambda + a^2)(\lambda + a^3 + a^4 + a^5 + a^6),$$

where it is easy to verify that all four factors appearing are indeed prime.

Many languages have prime decompositions, although it might appear not to be the case at a first look. A typical example is the prime decomposition

$$\Sigma^* = (\lambda + \Sigma)(\lambda + \Sigma(\Sigma^2)^*),$$

where $\Sigma$ is an alphabet. This type of decompositions can be extended to concern any *length codes*. In what follows we define also the well-known notion of a *code*, [Rozenberg and Salomaa 1997].

**Definition 2** *A language $L$ is a* length code *(resp. a* code*) if every equation*

$$u_1 \dots u_k = v_1 \dots v_l,\ u_i, v_j \in L,\ 1 \le i \le k,\ 1 \le j \le l,$$

*implies that $k = l$ (resp. that $k = l$ and $u_i = v_i$, for $1 \le i \le k$).*

For instance, the language $\{a, ab, ba\}$ is not a code but it is a length code. Definition 2 follows [Salomaa et al.]. Length codes were called *numerically decipherable* in [Weber and Head 1996]. The following result is due to [Han et al. 2007, Salomaa et al.].

**Theorem 1** *If a regular language $L$ a length code, then $L^*$ has a prime decomposition consisting of two regular factors.*

In the sequel we consider also infinite products $\prod_{i=1}^{\infty} L_i$, where each $L_i$ is a language. Then we consider only finite words defined by the product. We also assume that each $L_i$ contains the empty word. Indeed, an infinite product of languages defines finite words only if all of these languages, with at most finitely many exceptions, contain the empty word. In this case there is a language $L'$ and an integer $m \ge 1$ such that the original product can be written as

$$\prod_{i=1}^{\infty} L_i = L' \prod_{i=m}^{\infty} L_i,$$

where each language in the product on the right side contains the empty word. Hence, our assumption that $\lambda \in L_i$ is natural from the point of view of decompositions of languages.

If every language in the product $\prod_{i=1}^{\infty} L_i$ contains the empty word, then each word in the product belongs to a finite prefix of the product.

## 3 The language $L_D$ and the operation of ordered catenation closure

We now come to the interesting language

$$L_D = \lambda + \{a^{i_1} b^{i_1} a^{i_2} b^{i_2} \dots a^{i_k} b^{i_k} | k \ge 1,\ 1 \le i_1 < i_2 < \dots < i_k\}$$

considered in [Han et al. 2006]. The language $L_D$ is not context-free but its complement is. By a careful analysis concerning the form of the possible factors, it is shown in [Han et al. 2006] that $L_D$ has no prime decomposition. As will be seen in the sequel, this fact is an example of a general result. In this context an operation related to catenation will be useful.

The first observation is that $L_D$ can be expressed as an infinite product

$$L_D = \prod_{i=1}^{\infty}(\lambda + a^i b^i),$$

where obviously all factors are prime. This product is obtained in a specific way from the "basic" language $\{a^i b^i | i \geq 1\}$. This "specific way" can be formalized as the notion of an *ordered catenation closure.*

Recall that the *lexicographic ordering* of a language $L$ is the total ordering of the words in $L$, where the words are first ordered according to increasing length, and alphabetically among the words of the same length. (Here it is assumed that the alphabet of $L$ is ordered.) Thus, the first few words in the lexicographic ordering of the language $\{a, b\}^*$ are

$$\lambda, a, b, aa, ab, ba, bb, aaa, aab, aba, \ldots$$

**Definition 3** *The* ordered catenation closure $L^\diamond$ *of a language $L$ is defined by*

$$L^\diamond = \prod_{i=1}^{n}(\lambda + x_i),$$

*where $x_1, x_2, \ldots$ is the lexicographic ordering of $L$, and $n$ is the cardinality of $L$. ($n$ is an integer or $\infty$.)*

If $\lambda \in L$, we omit the factor $(\lambda + \lambda)$ from the product.

Consider the language $L_d = \{a^i b^i | i \geq 1\}$. Obviously,

$$L_d^\diamond = \prod_{i=1}^{\infty}(\lambda + a^i b^i) = L_D.$$

Based on this representation, we will now give a fairly simple argument showing that $L_D$ does not possess a prime decomposition.

We *claim* that, whenever $L_D = L_1 L_2$, then $L_2$ is not prime. Consequently, $L_D$ cannot have a prime decomposition because it obviously is not itself a prime, and the last factor in any decomposition cannot be prime.

We may assume that neither of $L_1$ and $L_2$ consists of the empty word alone. On the other hand, we must have $\lambda \in L_1$ and $\lambda \in L_2$, so every word in $L_1$, as well as in $L_2$, is in $L_D$. This means that the words $a^i b^i \in L_D, i \geq 1$, must each belong to $L_1$ or $L_2$ because none of them is a product of two words in $L_D$. We cannot have $a^ib^i \in L_2$ and $a^{i+j}b^{i+j} \in L_1,\ j \geq 0$, because, otherwise, we would get the wrong order of the exponents in the product. Thus, there is a number $k$ such that $a^ib^i \in L_2$ whenever $i \geq k$. We choose the smallest among such numbers $k$. It is now easy to see that, for some $L_2'$, we have the equation

$$L_2 = (\lambda + a^kb^k)L_2',$$

which establishes our claim.

Indeed, let $L_2'$ be the left derivative of $L_2$ with respect to the word $a^kb^k$. Every word in $L_2'$ must belong to $L_D$ because, otherwise, $L_2$ would have words not in $L_D$. Moreover, no word in $L_2'$ has a factor $a^ib^i$ with $i \leq k$, and $a^kb^k$ is a prefix of every nonempty word in $L_2$. From these observations the required equation follows.

Clearly, the language $L_d$ is a code. The above argument will be generalized in the next section to concern all codes and length codes.

## 4 Nonexistence of prime decompositions

We are now ready to establish the following result.

**Theorem 2** *If $L$ is a code then $L^\diamond$ has a prime decomposition exactly in case $L$ is finite.*

*Proof.* If $L$ is finite, so is $L^\diamond$, and hence has a prime decomposition. Assume that $L = \{x_1, x_2, x_3, \ldots\}$ is an infinite code, where the ordering of the words is lexicographic. We now establish the following

*Assertion.* Whenever we have

$$L^\diamond = \prod_{i=1}^{n}(\lambda + x_i) = L_1L_2,$$

for some languages $L_1$ and $L_2$, then $L_2$ is not prime.

Clearly, Theorem 2 follows from this Assertion because any prime decomposition can be changed into $L_1L_2$ by defining $L_1$ as the product of all factors except the last one. (Observe that $L^\diamond$ is not prime itself.)

To prove the Assertion, note first that $\lambda \in L_1$ and $\lambda \in L_2$. This implies that both of the languages $L_1$ and $L_2$ are contained in $L^\diamond$. A further conclusion is that each of the words $x_i \in L$ belongs to either $L_1$ or $L_2$. (This follows because $L$ is a code and, hence, a length code. Thus a word in $L$ cannot be a product of two words in $L$.) It cannot be the case that

$$x_j \in L_1,\ x_i \in L_2,\ i < j.$$

This would imply that we could write $x_j x_i = x_k x_l$, for some $x_k, x_l,\ k < l$, which would contradict the fact that $L$ is a code. Therefore, there is an index $t$ such that $x_i \in L_2$ whenever $i \geq t$. We choose the smallest such index $t$.

Let $L_2'$ be the left derivative of $L_2$ with respect to the word $x_t$. To complete the proof the Assertion, we prove the validity of the equation

$$L_2 = (\lambda + x_t)(\lambda + L_2').$$

The right side of the equation is contained in the left side. As regards $x_t$ and the product $x_t L_2'$, this is obvious. To show that $L_2'$ is contained in $L_2$, we assume the contrary and let $w$ be a word in the difference $L_2' - L_2$. By the definition of $L_2'$ we have $x_t w \in L_2 \subseteq L^\diamond$. By the definition of $L^\diamond$, $w$ is in $L^\diamond$. It is neither in $L_1$, nor a product of a word in $L_1$ and a word in $L_2$, because in both cases we would contradict the fact that $L$ is a code. Hence, $w$ is in $L_2$, and our assumption to the contrary is wrong, and the right side of the equation is contained in the left side.

Consider an arbitrary word $u \in L_2$. Clearly, $x_i,\ i < t$, cannot be a prefix of $u$. If $x_j,\ j > t$, is a prefix of $u$, then $x_t u$ is in $L^\diamond$. Clearly $x_t u$ is not in $L_1$. If it would be a product of a word in $L_1$ and one in $L_2$, we would again get a contradiction with the fact the $L$ is a code. Consequently, $x_t u \in L_2$ and $u \in L_2'$. Hence, also the left side of our equation is contained in the right side. □

Theorem 2 can be extended to concern length codes. We omit the detailed proof of the following theorem. The argument follows the lines on the proof of Theorem 2. However, the proof concerning the impossibility of products $x_j x_i,\ i < j$, is more involved for length codes. For instance, consider the length code

$$\{b = x_1,\ bab = x_2,\ bba = x_3.\}$$

The product $x_3 x_1$ must be considered because its "proper order" $x_1 x_2$ does not contradict the length code property.

**Theorem 3** *If $L$ is a length code then $L^\diamond$ has a prime decomposition exactly in case $L$ is finite.*

## 5 Properties of ordered catenation closure. The operation $\diamond^\infty$

We now investigate basic properties of the operation $\diamond$. We also consider iterations of this operation. In general they lead to infinite hierarchies towards the operation $\diamond^\infty$.

We begin with an example showing that sometimes rather surprising prime decompositions can be found for languages $L^\diamond$. Consider the language $L_p = \{a^q | q \text{ odd prime}\}$. Then $L_p^\diamond$ consists of $\lambda$ and all words $a^i$, where $i$ is a sum of distinct odd primes. Using the ideas similar to [Han et al. 2006, Han et al. 2007], we get the prime decomposition

$$L_p^\diamond = (\lambda + a^3 + a^5 + a^7 + a^8 + a^{11} + a^{12} + a^{14} + a^{16} + a^{19})(\lambda + a^{10})(\lambda + a^{10}(a^{20})^*.$$

Observe also that we have now

$$(L_p^\diamond)^\diamond = L_p^\diamond.$$

The following result about closure properties is easy to obtain.

**Theorem 4** *There are regular languages $L$ such that $L^\diamond$ is not context-free. If $L$ is context-sensitive (resp. recursively enumerable], so is $L^\diamond$.*

*Proof.* The language $ab^+$ (that can be viewed as a simplification of the language $L_d$ considered in Section 3) can be used to prove the first sentence. Clearly, the language

$$\prod_{i=1}^{\infty}(\lambda + ab^i)$$

is not context-free. The claim concerning recursively enumerable languages is obvious, and the one about context-sensitive languages can be established using standard techniques. □

We already referred to compositions of the operation $\diamond$. In general, we define

$$L^{\diamond^1} = L^\diamond,\ L^{\diamond^{i+1}} = (L^{\diamond^i})^\diamond,\ i \geq 1,$$

and, furthermore,

$$L^{\diamond^\infty} = \bigcup_{i=1}^{\infty} L^{\diamond^i}.$$

The following chain of inclusions is obvious:

$$L \subseteq L^\diamond \subseteq L^{\diamond^2} \subseteq \ldots \subseteq L^{\diamond^\infty} \subseteq L^*.$$

There are many examples where each inclusion in the chain is strict. For instance, consider $L = \{a, b\}$. Then every inclusion the the chain is strict. We have, in particular,

$$L^{\diamond^\infty} = L^*ab + a + b + \lambda.$$

However, the chain of inclusions yields immediately the following result.

**Theorem 5** *If $L$ is a star language, then $L^\diamond = L^{\diamond^\infty} = L^*$.*

The *converse* if this theorem is not valid. We can have

$$L^{\diamond} = L^{\diamond^{\infty}} = L^{*}$$

although $L$ is not a star language. The language $(\{a,b\})^{*} - ab$ constitutes a simple example.

The language $L_p$ considered above satisfies $L_p^{\diamond} = L_p^{\diamond^{\infty}}$ and, consequently, $L_p^{\diamond^{\infty}}$ has a prime decomposition. Clearly, $L_p^{\diamond}$ differs from both $L_p$ and $L_p^{*}$.

In the remainder of this section we consider the operations $\diamond$ and $(\diamond)^{\infty}$ associated with finite languages $F$. We denote the lexicographic order by $<_l$.

Assume that

$$F = \{x_1, x_2, \ldots, x_n\},\ x_i <_l x_{i+1},\ \text{for } 1 \leq i \leq n-1.$$

We assume that $F$ does not contain the empty word; the latter is always contained in $F^{\diamond}$. Words $w$ are referred to as *proper F-words* if we can write $w = y_1 \ldots y_k,\ k \geq 2$, where each $y_i$ belongs to $F$ and, moreover, $y_i <_l y_{i+1} \ldots y_k$, for $1 \leq i \leq k-1$.

Observe that we do not assume any properties of $F$ such as $F$ being a code or a length code. This means that the same word can have two decompositions in terms of words of $F$. For instance, if $F$ consists of the words $x_1 = a,\ x_2 = b,\ x_3 = ab$, then $w = ababab$ is a proper $F$-word because it can be written as $w = x_3x_1x_2x_3$. We can also write $w = x_3x_3x_3$ but this decomposition does not satisfy the conditions of a proper $F$-word.

Theorem 4 shows that regular languages are not closed under the operation $\diamond$. Finite languages are obviously closed under this operation but not under the operation $(\diamond)^{\infty}$. However, we will show that $F^{\diamond^{\infty}}$ is always a very simple regular language properly contained in $F^{*}$.

**Lemma 1** *If $F$ is a finite language, then $F^{\diamond^{\infty}}$ consists of all proper $F$-words and, in addition, of the words in $F$ and of the empty word.*

*Proof.* Observe that if $F$ contains only one word, then there are no proper $F$-words, and $F^{\diamond^{\infty}} = F + \lambda$. We assume that $F$ contains at least two (nonempty) words.

Clearly, the words listed in our lemma belong to $F^{\diamond^{\infty}}$. An obvious inductive argument shows that these words exhaust all words in $F^{\diamond^{\infty}}$. $\square$

We are now ready to characterize the set $F^{\diamond^{\infty}}$.

**Theorem 6** *Let $F$ be a finite language. If $F$ contains at least two words, then*

$$F^{\diamond^{\infty}} = F^{*}F_2 + F_1,$$

*where $F_1$ and $F_2$ are effectively constructible finite languages. If $F$ consists of only one word $w$, then $F^{\diamond^{\infty}} = w + \lambda$.*

*Proof.* The second sentence is obvious. To prove the first sentence, we consider the language $F = \{x_1, x_2, \ldots, x_n\}$ as above, with $n \geq 2$. We define the languages $F_1$ and $F_2$ as follows.

The language $F_1$ consists of the empty word, the words in $F$ and of all proper $F$-words $w$ satisfying $w <_l x_n$. The language $F_2$ consists of all proper $F$-words $w = y_1 y_2 \ldots y_k$ satisfying

$$x_n <_l w \text{ and } y_2 \ldots y_k \leq_l x_n.$$

Clearly, both languages are finite and effectively constructible. (A word $w$ can sometimes be written as a proper $F$-word starting with two different prefixes $y_1$ and $y_1'$. In such cases we choose the representation satisfying the condition for $F_2$.)

By Lemma 1, the set $F_1$ exhausts all words $w$ in $F^{\diamond^\infty}$ satisfying $w \leq_l x_n$. The remaining words in $F^{\diamond^\infty}$ are exhausted by $F^* F_2$ because all words in this language are proper $F$-words. □

For instance, if $F = \{ab, ba, (ba)^2\}$, then we have $F_1 = \{\lambda, ab, ba, abba, (ba)^2\}$ and

$$F^{\diamond^\infty} = F^*((ab + ba)(ba)^2 + (ab + ba)abba) + F_1.$$

In this case $F^{\diamond^\infty} = F^{\diamond^\infty}(\lambda + (ba)^3)$, which shows that $F^{\diamond^\infty}$ is not prime. This result can be generalized. If $F$ is a finite language as above with $n \geq 2$, we have

$$F^{\diamond^\infty} = F^{\diamond^\infty}(\lambda + x_{n-1} x_n).$$

If $F = \{a, a^k\},\ k \geq 2$, then

$$F^{\diamond^\infty} = \lambda + a + a^k a^*.$$

Now $F^{\diamond^\infty}$, although not prime itself, has a prime decomposition. This concerns all finite languages over one letter. It is shown in [Han et al. 2007] that all regular languages over one letter possess a prime decomposition. On the other hand, if $F$ is finite, then $F^{\diamond^\infty}$ is regular by Theorem 6.

These results can be summarized as follows.

**Theorem 7** *Let $F$ be a finite language containing at least two nonempty words. Then $F^{\diamond^\infty}$ is not prime. If, in addition, $F$ is a language over one letter, then $F^{\diamond^\infty}$ has a prime decomposition.*

It is an open problem whether $F^{\diamond^\infty}$ has always a prime decomposition. By Theorem 6, this problem is a special case of the more general problem: Does every regular language possess a prime decomposition?

An interesting problem about the complexity of finite languages is is to compare the state complexities of $F$ and $F^{\diamond}$. Here the technique of *representation* *forms*, [Salomaa 1969], is applicable. Essentially, a representation form is obtained by applying to the regular expression the distributive law to the right as long as possible, and then always crossing out one of two identical "end parts". The state complexity of the language is seen directly from the representation form. We hope to return to this in another context. Here we give an example.

Consider the language $L = \{ab, a^2b, a^3b\}$. Then the regular expression

$$L^\diamond = \lambda + ab + a^2b + a^3b + aba^2b + aba^3b + a^2ba^3b + aba^2ba^3b$$

gives rise to the representation form

$$\lambda + a(b(\lambda + a^2) + a(ab + b(\lambda + a^3b))).$$

(The end part $b+ab+ba^3b$ has been crossed out.)From this we infer immediately, [Salomaa 1969], that the minimal deterministic automaton for $L^\diamond$ has 12 states. We have included also the automata for $L$ and $L^*$ in the picture. In each case the sink and the transitions leading to it have been omitted.

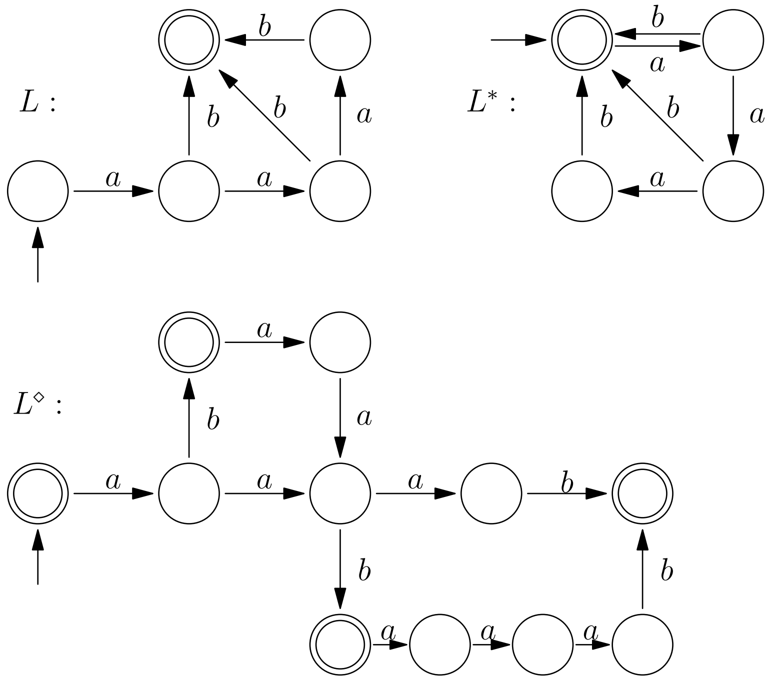


**Figure 1:** Variants of catenation closure

## 6 Open problems

We need the following definition from [Salomaa et al.]. We assume that each of the languages $L_i$ and $K_i$ properly contains the empty word.

**Definition 4** *A language $L$ has a* unique infinitary prime decomposition *if $L = \prod_{i=1}^{\infty} L_i$, where each $L_i$ is prime and, whenever $L = \prod_{i=1}^{\infty} K_i$, where each $K_i$ is prime, then $L_i = K_i$, for all $i$. If $L$ is over a one-letter alphabet, it is only required that the languages $K_i$ are the languages $L_i$ in some order.*

It is not difficult to see that the infinitary prime decomposition given above for the language $L_D$ is unique. A language can have both a prime decomposition in the sense of Definition 1 and an infinitary prime decomposition. For instance, as seen above, $\Sigma^*$ has a prime decomposition and also an infinitary prime decomposition

$$\Sigma^* = \prod_{w} (\lambda + w),$$

where $w$ runs through all nonempty words over $\Sigma$. But it is an *open problem* whether a language can have a prime decomposition and a *unique* infinitary one. As we have seen, $L_D$ is not such a language.

Another open problem is: Does every regular language have a prime decomposition? This is true for regular languages over one letter, [Han et al. 2007].

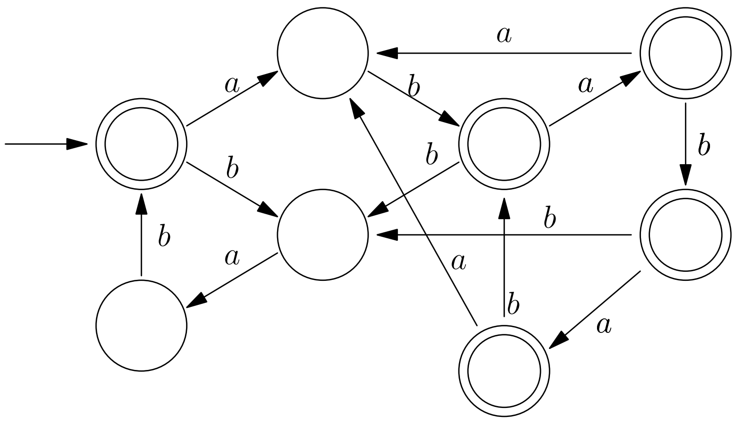


**Figure 2:** An automaton connected with the open problem

We noticed above that the language $K = \{ab, aba, bab\}$ is not a length code. The language $K^*$ has a prime decomposition into two nonregular factors, [Salomaa et al.]. Has it also a prime decomposition into regular factors?

The finite automaton accepting $K^*$ gives the impression that this is not the case, in view of the state constructions in [Mateescu et al. 2002]. But we have no conclusive proof. Again, we have omitted transitions to the garbage state from the picture. Observe that the automaton in the picture is not minimal: we have expanded the final states for $ab$ and $aba$ into two equivalent copies, to illustrate possibilities for decompositions.

## References


[Avgustinovich and Frid 2005] Avgustinovich, S.V. and Frid, A., A unique decomposition theorem for factorial languages. *Internat. J. Alg. Comp.* 15 (2005) 149–160.

[Czyzowicz et al. 2003] Czyzowicz, J., Fraczak, W., Pelc., A. and Rytter, W., Linear-time prime decompositions of regular prefix codes. *Internat. J. Found. Comp. Sci.* 14 (2003) 1019–1031.

[Frid 2007] Frid, A., Commutation in binary factorial languages. *Proc. DLT'2007*, Springer LNCS 4588 (2007) 193–204.

[Han et al. 2007] Han, Y.-S., Salomaa, A., Salomaa, K., Wood, D. and Yu, S., Prime decompositions of regular languages. *Theor. Comp. Sci.* 376 (2007) 60–69.

[Han et al. 2006] Han, Y.-S., Salomaa, K. and Wood, D., Prime decompositions of regular languages. In Ibarra, O. and Dang, Z. (eds.) *Developments in Language Theory 2006*, Springer LNCS 4036 (2006) 145–155.

[Han et al.] Han, Y.-S., Wang, Y. and Wood, D., Infix-free regular expressions and languages. *Internat. J. Found. Comp. Sci.*, to appear.

[Han and Wood 2005] Han, Y.-S. and Wood, D., The generalization of generalized automata: Expression automata. In *Implementation and Application of Automata, CIAA'04*, Springer LNCS 3317 (2005) 156–166.

[Mateescu et al. 2002] Mateescu, A., Salomaa, A. and Yu, S., Factorizations of languages and commutativity conditions. *Acta Cybernetica* 15 (2002) 339–351.

[Rozenberg and Salomaa 1997] Rozenberg, G. and Salomaa, A. (eds.), *Handbook of Formal Languages 1–3*. Springer-Verlag, Berlin, Heidelberg, New York (1997).

[Salomaa 1969] Salomaa, A., *Theory of Automata*. Pergamon Press, Oxford, London, New York (1969).

[Salomaa et al.] Salomaa, A., Salomaa, K. and Yu, S., Variants of codes and indecomposable languages. Submitted.

[Salomaa and Yu 2000] Salomaa, A. and Yu, S., On the decomposition of finite languages. In *Developments in Language Theory, DLT'99*, World Scientific Publ.Co. (2000) 22–31.

[Weber and Head 1996] Weber, A. and Head, T., The finest homophonic partition and related code concepts. *IEEE Trans. Inform. Theory* IT-42 (1996) 1569–1575.