*Acta Mathematica Academiae Paedagogicae Nyíregyháziensis*
**24** (2008), 249–256
www.emis.de/journals
ISSN 1786-0091

# ON EQUIVALENCE OF TWO TESTS FOR CODES

J. FALUCSKAI

Abstract. Our goal is to show the equivalence between two algorithms concerning uniquely decipherable codes. We show an automaton that solves this problem, and we define remainders for the states of automaton. Finally we show that remainders compose the sets of the algorithm of Sardinas–Patterson.

## 1. The algorithm of Sardinas – Patterson

The algorithm of Sardinas – Patterson is based on the following: Let us compute all the remainders in all attempts at a double factorization. It can recognize a double factorization by the fact that the empty word is one of the remainders.

Let $A$ be a set, which we call an *alphabet*. A *word* $w$ on the alphabet $A$ is a finite sequence of elements of $A$

$$w = (a_1, a_2, \ldots, a_n), \quad a_i \in A$$

The set of all words on the alphabet $A$ is denoted by $A^*$. If we omit the empty word from $A^*$ then we get $A^+$. Let $X$ and $Y$ be two subsets of $A^+$ and let $x \in X$ and $y \in Y$. Denote $X^{-1}Y$ the following set: $w$ is an element of $X^{-1}Y$ if $xw = y$.

Let $C$ be a subset of $A^+$, and let

$$\begin{aligned} U_1 &= C^{-1}C \setminus \{\varepsilon\} \\ U_2 &= C^{-1}U_1 \cup U_1^{-1}C \\ &\vdots \\ U_{n+1} &= C^{-1}U_n \cup U_n^{-1}C \end{aligned} \tag{1}$$

Thus we have:

2000 *Mathematics Subject Classification.* 94B35, 94A45, 68Q45.
*Key words and phrases.* Uniquely decipherable codes, automata, length-variable codes, Sardinas-Patterson algorithm.

**Theorem 1** ([2])**.** *Let $C \subset A^+$ and let $(U_n)_{n\geq 1}$ be defined as above. For all $n \geq 1$ and $k \in \{1,\dots,n\}$, we have $\varepsilon \in U_n$ if and only if there exists a word $u \in U_k$ and integers $i, j \geq 0$ such that*

$$uC^i \cap C^j \neq 0, \quad i+j+k=n \tag{2}$$

*Proof.* ([2]) We prove the statement for all $n$ by descending induction on $k$. Assume, first $k=n$. If $\varepsilon \in U_n$, taking $u=1, i=j=0$, the equation above is satisfied. Conversely, if (2) holds, then $i=j=0$. This implies $u=1$ and consequently $\varepsilon \in U_n$. Now let $n>k$, and suppose that the equivalence holds for $n, n-1, \dots, k+1$. If $\varepsilon \in U_n$, then by induction hypothesis, there exists $v \in U_{k+1}$ and two integers $i, j \geq 0$ such that

$$uC^i \cap C^j \neq 0, \qquad i+j+k+1=n$$

Thus there are words $x \in C^i; y \in C^j$ such that $vx=y$. Now $v \in U_{k+1}$, and there are two cases. Either there is a word $z \in C$ such that

$$zv = u \in U_k$$

or there exists $z \in C; u \in U_k$ such that

$$z = uv$$

In the first case, one has $ux = zy$, thus

$$uC^i \cap C^{j+1} \neq 0, u \in U_k$$

In the second case, one has $zx = uvx = uy$, thus

$$uC^j \cap C^{i+1} \neq 0, u \in C_k$$

In both cases, formula (2) is satisfied. Conversely, assume that there are $u \in U_k$ and $i, j \geq 0$ with

$$uC^i \cap C^j \neq 0, \qquad i+j+k+1=n$$

Then

$$ux_1x_2\dots x_i = y_1y_2\dots y_j$$

for some $x_r, y_s \in C$. If $j=0$, then $i=0$ and $k=n$. Thus $j \geq 1$. Once more, we distinguish two cases, according to the length of $u$ compared to the length of $y_1$. If $u = y_1v$ for some $v \in A^+$, then $v \in C^{-1}U_k \subset U_{k+1}$ and further

$$vx_1x_2\dots x_i = y_2\dots y_j$$

Thus $vC^i \cap C^{j-1} \neq 0$ and by the induction hypothesis $\varepsilon \in U_n$. If $y_1 = uv$ for some $v \in A^+$, then $v \in U_k^{-1}C \subset U_{k+1}$ and

$$x_1x_2\dots x_i = vy_1y_2\dots y_j$$

showing that $C^i \cap vC^{j-1} \neq 0$. Thus again $\varepsilon \in U_n$ by the induction hypothesis. This concludes the proof. □

**Theorem 2** ([2])**.** *The set $C \subset A^+$ is a uniquely decipherable code if and only if none of the sets $U_n$ defined above contains the empty word.*

*Proof.* ([2]) The proof of theorem is based on the previous lemma. If $X$ is not a code, then there is a relation

$$x_1x_2\dots x_n = y_1y_2\dots y_m; x_i, y_j \in C; x_1 \neq y_1$$

Assume $|y_1| < |x_1|$. Then $x_1 = y_1u$ for some $u \in A^+$. But then

$$u \in U_1 \text{ and } uC^{n-1} \cap C^{m-1} \neq 0$$

According to the lemma, $\varepsilon \in U_{n+m-1}$. Conversely, if $\varepsilon \in U_n$, take $k = 1$ in the lemma. There exists $u \in U_1$ and integers $i, j \geq 0$, such that $uC^i \cap C^j \neq 0$. Now $u \in U_1$ implies that $xu = y$ for some $x, y \in C$. Furthermore $x \neq y$ since $u \neq \varepsilon$. It follows from $xuC^i \cap xC^j \neq 0$ that $yC^i \cap xC^j \neq 0$, showing that $C$ is not a code. This establishes the theorem. □

Our goal is to show the equivalence between the Sardinas - Patterson algorithm and the following algorithm:

## 2. An automaton to test codes

We construct an automaton for the code over $A$ by union of automata of codewords. If $w = x_1x_2\dots x_n$ is a codeword then the automaton $\mathcal{A}(w)$ of codeword $w$ is $\mathcal{A}(w) = (Q, q_i, Q_t, A, \delta)$ where $q_i$ is the initial state of $\mathcal{A}(w)$ and $Q_t$ is the set of terminal states. $Q$ is the set of states and $Q_t = \{q_i\}$; $q_i \in Q$; $card(Q) = length(w)$ since the rules of automaton $\mathcal{A}(w)$ are the following:

$$\begin{aligned}
\delta(q_i, x_1) &= q_{x_1} \\
\delta(q_{x_1}, x_2) &= q_{x_1x_2} \\
&\vdots \\
\delta(q_{x_1x_2\dots x_{n-2}}, x_{n-1}) &= q_{x_1x_2\dots x_{n-2}x_{n-1}} \\
\delta(q_{x_1x_2\dots x_{n-1}}, x_n) &= q_i
\end{aligned}$$

thus, $\mathcal{A}(w)$ can recognize $w^*$.

If we join the automata of codewords, then we get the automaton

$$\mathcal{A}(w_1, \dots, w_n)$$

of code $C = \{w_1, \dots, w_n\}$. We can use notation $\mathcal{A}(C)$, too. So

$$\mathcal{A}(C) = (Q = Q^{w_1} \cup \dots \cup Q^{w_n}, q_i, Q_t = \{q_i\}, A, \delta = \delta^{w_1} \cup \dots \cup \delta^{w_n})$$

Obviously, $\mathcal{A}(C)$ accepts $C^*$. An automaton is non deterministic if there is more then one rule for the same pair of state and symbol.

If a string $S$ decipherable on code $C$ then $\mathcal{A}(C)$ accepts $S$, namely $\mathcal{A}(C)$ read it and stay in $q_i$ state. If $S$ is not uniquely decipherable then we can follow different paths during reading. We join these different paths by the equivalent deterministic automaton. Formally

$$\overbrace{\underbrace{x_1 \dots x_{|w_{i_1}|}}_{w_{i_1}} \dots x_{|w_{j_1}|}}^{w_{j_1}} \dots \overbrace{\dots \dots \underbrace{\dots \dots x_{|S|}}_{w_{i_m}}}^{w_{jn}}$$

$$\begin{aligned}
\delta(q_i, x_1) &= q_{x_1} \\
\delta(q_{x_1}, x_2) &= q_{x_1 x_2} \\
&\vdots \\
\delta(q_{x_1 x_2 \dots x_{|w_{i_1}|-1}}, x_{|w_{i_1}|}) &= \{q_{x_1 x_2 \dots x_{|w_{i_1}|}}, q_i\} \\
\delta(\{q_{x_1 x_2 \dots x_{|w_{i_1}|}}, q_i\}, x_{|w_{i_1}|+1}) &= \{q_{x_1 x_2 \dots x_{|w_{i_1}|+1}}, q_{x_{|w_{i_1}|+1}}\} \\
&\vdots \\
\delta(\{q_{x_1 x_2 \dots x_{|w_{j_1}|-1}}, q_{x_{|w_{i_1}|+1} \dots x_{|w_{j_1}|-1}}\}, x_{|w_{j_1}|}) &= \{q_i, q_{x_{|w_{i_1}|+1} \dots x_{|w_{j_1}|}}\} \\
&\vdots \\
\delta(\{q_{x_{|w_{j_n}|+1} \dots x_{|S|-1}}, q_{x_{|w_{i_m}|+1} \dots x_{|S|-1}}\}, x_{|S|}) &= \{q_i, q_i\} = q_i
\end{aligned}$$

Thus two (or more) factorizations of a string will be ended by using two (or more) rules with right side $q_i$.

**Theorem 3.** *A code is uniquely decipherable if and only if at the most one state equal to $q_i$ in right side of any rule of $\mathcal{A}_D(C)$ , namely*

$$\forall\, \delta(\{q_{i_1}, \dots, q_{i_n}\}, x) = \{q_{j_1}, \dots, q_{j_m}\} \in \mathcal{A}_D(C) : \nexists \quad l, k : q_{j_l} = q_{j_k} = q_i$$

## 3. The equivalence

As we wrote, the algorithm of Sardinas – Patterson is based on the remainders of double factorization.

By equation (1) it is in evidence that the set $U = \bigcup_{i=1}^{n} U_i$ contains every remainder. The definition is recursive, but in the case of infinity recursion $card(U) \leq k \in \mathbb{N}$, since the length of any remainder is less than that of the longest codeword. In non deterministic automaton $\mathcal{A}(C)$ a remainder appears in the following way: If $w_{i_1} \dots w_{i_n} \alpha = w_{j_1} \dots w_{j_m}$ i.e. $w_{i_1} \dots w_{i_n} = x_1 x_2 \dots x_n$; $w_{j_1} \dots w_{j_m} = x_1 x_2 \dots x_n x_{n+1} \dots x_m$; $\alpha = x_{n+1} \dots x_m$ then $q_i \xrightarrow{w_{i_1} \dots w_{i_n}} q_i$ and $q_i \xrightarrow{w_{j_1} \dots w_{j_m}} q_i$. There is no way such that $q_i \xrightarrow{\alpha} q_i$ ($\alpha \notin C$), so there have to be another way $q_i \xrightarrow{w_{i_1} \dots w_{i_n}} q_x; q_x \neq q_i$ and a way $q_x \xrightarrow{\alpha} q_i$. So in $\mathcal{A}_D(C)$ there is a state which contains both $q_x$ and $q_i$. If we link the touched symbols by the way from $q_x$ to $q_i$ in automaton $\mathcal{A}(C)$, then we obtain the remainder $\alpha$.

Generally we can say that: If a state of $\mathcal{A}_D(C)$ contains $q_i$ then mate states of $q_i$ will show remainders. If $q_i$ is accessible by different ways from a state in $\mathcal{A}(C)$, then the state will generate more remainders. It is advisable to indicate all possible remainders of all states of $\mathcal{A}(C)$. After this notation we can receive remainders directly from the names of states which appearing with $q_i$ in $\mathcal{A}_D(C)$. Let $R(q)$ denote the set of remainders of state $q$ of $\mathcal{A}(C)$, and let $U_i$ the sets of Sardinas – Patterson algorithm.

**Corollary 1.** $\bigcup_{i=1}^{n} U_i = \{R(q_x) \mid \{q_x, q_i\} \subseteq q \text{ of } \mathcal{A}_D(C)\}$

*Example* 1. Let $C = \{010, 1101, 10, 11\}$. Thus we get $\mathcal{A}(010, 1101, 10, 11)$ in Figure 1. ($q_i = S$). Computing $R(q)$ for all states of $\mathcal{A}(C)$ we receive the following table. It is very easy to compute the remainders of a state $q$, since if we follow all the way from state $q$ to $q_i$ we will get them. Now $q_i = S$. In this case we do not write the remainder of $S$ because it is $\varepsilon$ for all time. We obtain $\varepsilon$ as a remainder if and only if there is a state of $\mathcal{A}_D(C)$ that contains two (or more) $S$ during the generation, as we mentioned above. (See the rule $\delta(\{B, C\}, 0) = \{S, S\}$) of $\mathcal{A}_D(C)$ for this case). If $S$ is accessible by different ways from a state $q$ in $\mathcal{A}(C)$, then the state $q$ generates more remainders. We can see an example for this in the case of state $C$. The table of remainders is the following:

| State | Remainders |
|---|---|
| A | 10 |
| B | 0 |
| C | 1, 0, 101 |
| D | 01 |
| E | 1 |

Table 1. Remainders of $\mathcal{A}$(010, 1101, 10, 11)

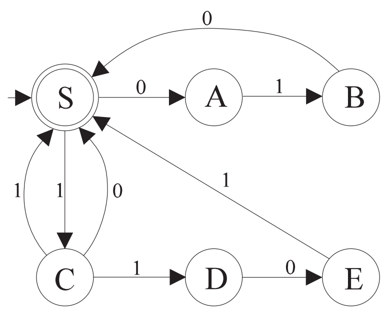


Figure 1. Automaton $\mathcal{A}$(010, 1101, 10, 11)

Let us construct $\mathcal{A}_D$(010, 1101, 10, 11). The result is given in Figure 2. We can see that

$$\delta(\{B, C\}, 0) = \{S, S\} = \{S\}$$

so $\varepsilon$ is one of the remainders (in that case the code is not uniquely decipherable).

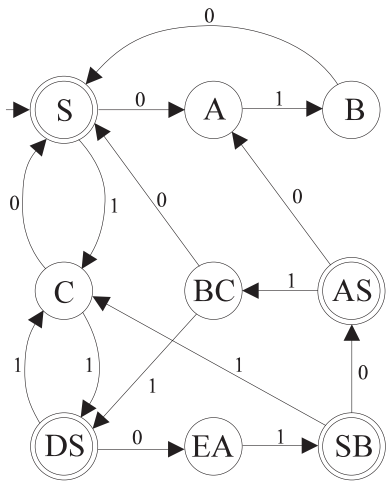


FIGURE 2. Automaton $\mathcal{A}_D$(010, 1101, 10, 11)

Finally we give those states, which appear together with $S$ in a common states of $\mathcal{A}_D(C)$: $\{A, B, D, S\}$. So the possible remainders are the following: $\{01, 0, 10, \varepsilon\}$. We get the same result by Sardinas – Patterson algorithm:

$$U_1 = \{01\} \quad U_2 = \{0\} \quad U_3 = \{10\} \quad U_4 = \{\varepsilon\}$$

so $\bigcup U_i = \{01, 0, 10, \varepsilon\}$.

We can show a closer relationship between the automaton and the Sardinas – Patterson algorithm. We can realize from the definition of the set $U_i$ that its elements derive from two strings, which contain whole codewords in all $i$ piece. (Except the case of $\varepsilon$, since the number of codewords is $i+1$.) In $\mathcal{A}_D(C)$ we can receive the number of codewords from the number of touched $S$. Unfortunately it is not sufficient, because of the following strings

$$w_{i_1} \dots w_{i_k} w_{i_{k+1}} \dots w_{i_n} \alpha = w_{j_1} \dots w_{j_k} w_{j_{k+1}} \dots w_{j_m}; \quad \forall \quad 1 \le l \le k: \quad i_l = j_l$$

If we ignore the prefix parts of strings, then we obtain the right subscript for $U$. Formally it looks like that

$$i = \begin{cases} n + m - 2k & \text{if } \alpha \neq \varepsilon \\ n + m - 2k - 1 & \text{if } \alpha = \varepsilon \end{cases} \tag{3}$$

So we have to regard the touched codewords for every state. Of course we can get a state by different ways, so more codeword sequences could belong to a state. Let the codewords be indicated by their last touched state and the symbol before $S$. For example in Figure 1 the notation of codeword 010 is $B_0$, since $S \xrightarrow{0} A \xrightarrow{1} B \xrightarrow{0} S$. If we give the possible touched sequences of codewords for all states which appear together with $S$ in common states of $\mathcal{A}_D(C)$, then we get index of $U_i$ by (3). We obtain the remainder by the table of remainders.

*Example* 2. If $C = \{010,\ 1101,\ 10,\ 11\}$ then we receive the following table from $\mathcal{A}_D(C)$. We do not have to write all possible touched sequences of codewords for the states, since it is in evidence by (3) that the index of $U_i$ is the same for all sequences for a state. For the state pair $(S, A)$ the remainder is 10

| State | Touched codewords | | |
|---|---|---|---|
| $S$ | $C_1B_0$ | $C_1C_1B_0$ | $E_1C_1B_0$ |
| $A$ | $E_1$ | $C_1E_1$ | $E_1E_1$ |
| $S$ | $E_1$ | $C_1E_1$ | $E_1E_1$ |
| $B$ | $C_1$ | $C_1C_1$ | $E_1C_1$ |
| $S$ | $C_1$ | $C_1C_1$ | $C_1E_1C_1$ |
| $D$ | $-$ | $C_1$ | $C_1E_1$ |
| $S$ | $E_1B_0$ | $C_1E_1B_0$ | $E_1E_1B_0$ |
| $S$ | $C_1B_0C_0$ | $C_1C_1B_0C_0$ | $E_1C_1B_0C_0$ |

Table 2. Some touched sequences of codewords in $\mathcal{A}_D(010,\ 1101,\ 10,\ 11)$

(see the table of remainders, where 10 belongs to $A$), and the index is 3, since $length(C_1B_0) + length(E_1) = 2 + 1 - 2 * 0 = 3$ (see (3)). We get 3 for $C_1C_1B_0, C_1E_1$ as well: $length(C_1C_1B_0) + length(C_1E_1) = 3 + 2 - 2 * 1 = 3$, since they have common prefix part, whose length is 1. We have to mention the state pair $(S, S)$. The remainder is $\varepsilon$, (we did not give the remainder of $S$ in the table of remainders, because it is $\varepsilon$ for all time), the index is $length(E_1B_0) + length(C_1B_0C_0) = 2 + 3 - 2 * 0 - 1 = 4$. We use the second part of (3), since the remainder is $\varepsilon$. The following table shows the summary of results: Thus

| State pair | Remainder | Index of $U_i$ |
|---|---|---|
| $S, A$ | 10 | 3 |
| $S, B$ | 0 | 2 |
| $S, D$ | 01 | 1 |
| $S, S$ | $\varepsilon$ | 4 |

Table 3. The sets $U_i$ by $\mathcal{A}_D(010,\ 1101,\ 10,\ 11)$

$$U_1 = \{01\} \quad U_2 = \{0\} \quad U_3 = \{10\} \quad U_4 = \{\varepsilon\}$$

are equal to $U_i$ of Sardinas–Patterson algorithm.

We have to mention here that it was a simple example, we usually get more index for a state pairs, and number of $U_i$ can be infinite, too.

## References

[1] X. Augros and I. Litovsky. Algorithms to test rational $\omega$ code. In *Mathematical Foundations of Informatics'99 Conference*, Hanoi, 1999.

[2] J. Berstel and D. Perrin. *Theory of codes*, volume 117 of *Pure and Applied Mathematics*. Academic Press Inc., Orlando, FL, 1985.
[3] R. König. Lectures on codes. Internal Reports of the IMMD ILectures on codes.
[4] A. A. Sardinas and C. W. Patterson. A necessary and sufficient condition for the unique decomposition of coded messages. *IRE Internat. Conv. Rec.*, 8(104):108, 1953.
[5] K. Tsuji. An automaton for deciding whether a given set of words is a code. *Sūrikaisekikenkyūsho Kōkyūroku*, 1222:123–127, 2001. Algebraic semigroups, formal languages and computation (Japanese) (Kyoto, 2001).

*Received September 30, 2007.*

Institute of Mathematics and Computer Science,
College of Nyíregyháza,
Nyíregyháza, Pf. 166,
4401 Hungary
*E-mail address*: falu@nyf.hu