OPEN PROBLEMS IN DESIGNING CONVOLUTIONAL CODES BASED ON QUASIGROUPS

Dejan Spasov
Institute of Informatics
Skopje, Macedonia

ABSTRACT

Use of quasigroup transformations to build error-correcting codes was first proposed by Gligoroski, Markovski, and Kocarev [1]. In this paper, we observe that any quasigroup code designed so far, can be modelled as finite state machine. Thus we provide a natural generalization of convolutional codes and polynomial-time decoding of quasigroup codes.

I. INTRODUCTION

Let Q is a finite alphabet of $|Q|$ letters and let Q^n is the set of all strings of length n over Q. Then a *code* C is subset of Q^n of M elements. Elements of the code $c_i \in C$ are called *codewords*.

Let $d(x,y)$ denotes the *Hamming distance*, i.e. the number of coordinates in which two strings x and y differ, and let $wt(x)$ denotes the (*Hamming*) *weight*, i.e. the number of nonzero coordinates of x. Then we say that the code C has *(minimum) distance* d if

$$d = \min\left\{d\left(c_i, c_j\right)\right\}, \forall c_i, c_j \in C, i \neq j\,. \qquad (1)$$

Usually, the code C is gifted with the ability to efficiently find the nearest codeword, in terms of Hamming distance, for any string of length n. Due to their ability to find the nearest codeword, error-correcting codes are used for transmission of digital information over a noisy channel. A k-bit message is one-to-one encoded into a n-bit codeword and sent over a noisy channel. The channel can flip few bits, but the receiver may still be able to deduce which codeword has been sent, thus recovering the original message.

In this paper we classify quasigroup codes (according to the construction in [1]), as convolutional codes in which the combinatorial logic is based on quasigroup operations. With this approach we hope we will establish a bridge between quasigroup designs and convolutional codes.

II. CONVOLUTIONAL CODES

A convolutional encoder is a circuit made of two parts: shift register and combinatorial logic [2] (implemented with XOR gates) (Figure 1).

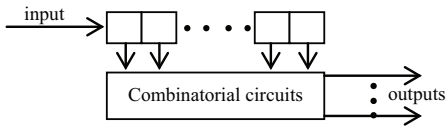

Figure 1 Convolutional Encoder

Given a information sequence of k letters, convolutional encoder produces N output sequences of k letters each, thus producing a code with rate $1/N$. The i-th letter in each of the N output sequences is obtained by combining the i-th letter from the information sequence and previous $j, j \leq i-1,$ letters from the information sequence. Figure 2 shows such a design of a convolutional encoder with rate ½.

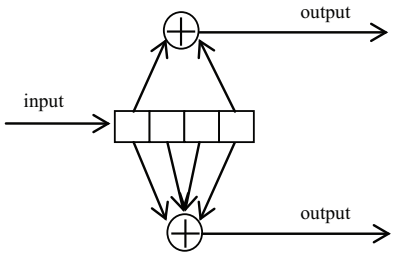

Figure 2 Rate 1/2 convolutional encoder

A. Recursive and Non-Recursive Convolutional Encoders

There are two main categories of convolutional encoders: *recursive* and *non-recursive* convolutional encoders. The convolutional encoder from figure 2 is an example of non-recursive convolutional encoder. More precisely, a convolutional encoder is non-recursive if the i-th letter in each of the N output sequences is obtained by combining the i-th letter from the information sequence and previous $j, j < i-1,$ letters the information sequence. If the i-th letter in the output sequences depends on all i letters from the information sequence, then the encoder is recursive.

We say that the encoder is systematic if one of the N output sequences is identical to the input information sequence. Figure 3 shows a simple rate ½ non-recursive systematic encoder, while figure 4 shows a recursive systematic encoder (RSC) with rate ½ .

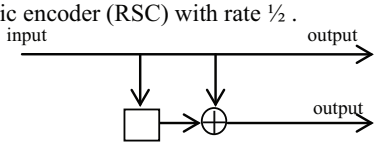

Figure 3 Rate 1/2 non-recursive systematic encoder

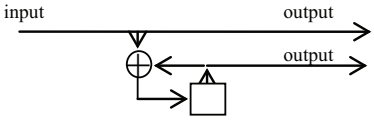

Figure 4 Rate 1/2 recursive systematic encoder

B. Trellis Diagram and decoding of convolutional codes

A convolutional encoder with m registers is finite state machine with 2^m states. *Trellis diagram* is labelled 2^m-partite graph, in which every path represents a valid codeword (Figure 5). Vertices of the graph represent all possible 2^m states and edges between two vertices represent possible state transitions. Edge labels represent the input letters to the encoder and the appropriate output letters. Trellis diagram of convolutional codes gives a hint about the decoding process; if the received sequence does not represent a valid path through the trellis diagram, errors have occurred. Then the decoding objective is to find the most probable valid path though the trellis. Several decoding algorithms exist for decoding convolutional codes. The most famous ones are the Viterbi algorithm [3,4] and the BCJR algorithm[5]. The Viterbi algorithm is universally used and is highly parallelizable. Viterbi decoders are easily implemented in hardware and software [6].

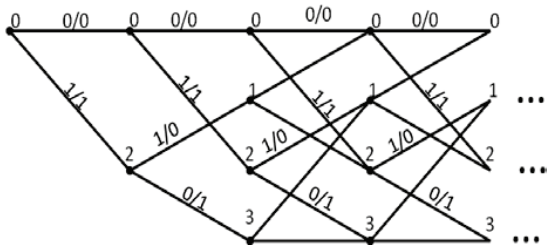

Figure 5 Trellis diagram of a convolutional code

III. QUASIGROUP-BASED CODES

A. Introduction to quasigroups

A *quasigroup* is an alphabet Q endowed with binary operation $*: Q \times Q \to Q$, such that for all $u, v \in Q$ there exist unique $x, y \in Q$ such that

$$\begin{cases} u * x = v \\ y * u = v \end{cases}.$$

From engineering perspective, quasigroup can be seen as circuit with two inputs and one output (Figure 6 a)). The circuit works the following way: let the input 2 on figure 6 a) maintains a constant letter. Then any random string ($\in Q^n$) that enters on the input 2 will change its representation according to the quasigroup transition diagram on figure 6 b.

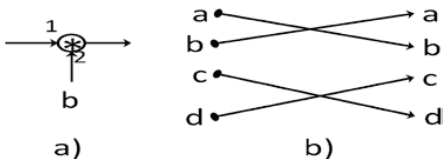

Figure 6 Circuit representation of quasigroups

B. Quasigroup codes based on E and D transforms

Given a string $S = (s_1, \ldots, s_n)$ over the alphabet Q and random letter $a \in Q$; E-transform is the string $E = (e_1, \ldots, e_n)$ such that

$$\begin{cases} e_1 \overset{def}{=} a * s_1 \\ e_i \overset{def}{=} s_i * e_{i-1} \end{cases}.$$

Circuit representation of the E-transform is given in figure 7. It is obious that we need memory register to store the previous letter.

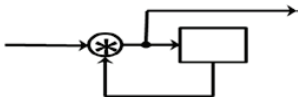

Figure 7 Circuit representation of E-transform

Given a string $S = (s_1, \ldots, s_n)$ over the alphabet Q and random letter $a \in Q$; D-transform is the string $D = (d_1, \ldots, d_n)$ such that

$$\begin{cases} d_1 \overset{def}{=} a * s_1 \\ d_i \overset{def}{=} s_i * s_{i-1} \end{cases}.$$

Circuit representation of the D-transform is given in figure 8.

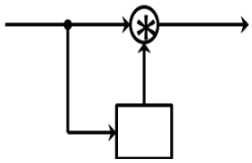

Figure 8 Circuit representation of D-transform

From these transforms we can build more complex transformations. For example, figure 9 shows four D-transform circuits connected in cascade line.

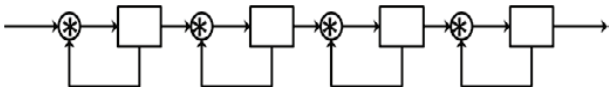

Figure 9 Complex quasigroup-based string transformation

Let $M = (m_1, \ldots, m_k)$ be a message that we wish to encode, i.e. a string over Q. Let v, $v \notin Q$, be a special letter, i.e. a blank space, and let $V = (v, \ldots, v)$ is a string of $n-k$ blank spaces. Let the string I consists of all letters from M and V randomly arranged. For example,

$$I = (m_1, m_2, v, v, m_3, m_4, v, v \ldots, m_{k-1}, m_k, v, v)$$

Then the codeword $C = (c_1, \ldots, c_n)$ that corresponds to the message M is obtained from a complex transformation (as in Figure 9) over the string I.

IV. Open Problems

We have observed the complex quasigroup transformations (as in fig. 9) as finite state machines. It is obvious that any encoder based on quasigroups will have trellis diagram simila to the example on figure 5. Hence quasigroup codes bear similarity to convolutional codes. The most important fact from this similarity is that quasigroup codes, too, can be decoded in polynomial time with the Viterbi algorithm.

However, in order to make these codes practically applicable, several questions must be resolved. Here we list three of them.

It is well known that the combinatorial logic of the convolutional codes is exclusively based on XOR operations, thus achieving low circuit complexity. With the introduction of quasigroup operations, increased complexity in the combinatorial logic is inevitable. Hence, the first important issue that need to be answered is:

Open problem #1: What is the (circuit) complexity of the combinatorial logic based on quasigroup operations?

Trivially, quasigroup circuit (Figure 6) can be made with look-up tables in quadratic complexity. Here we ask if good quasigroup circuit can be made with linear complexity.

Convolutional codes are known to coding theorists for 50 years. Over the time several designs of convolutional encoders has proved to be good. One such a design, known as RSC encoder is given in the following figure.

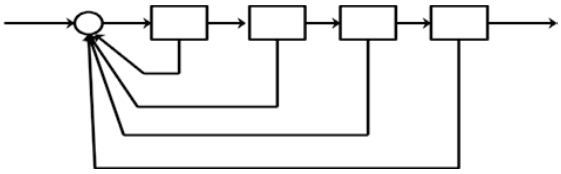

Figure 10 RSC encoder

By introducing quasigroups we want to obtain better codes in terms of signal-to-noise ratio. In the design process one can learn from these designs of convolutional codes and can use them in cryptographic designs based on quasigroups (figure 9). Hence:

Open problem #2: Establish guidelines for design of good convolutional codes based on quasigroups

Finally, we would like to address the issue of decoding. Quasigropup encoders, as well as convolutional encoders, are finite state machines whose operation can be described with the so-called trellis diagrams. Hence, the decoding of these codes can be successfully done with the Viterby algorithm or MAP algorithm. However, we expect that quasigroup encoders to have more states than the traditional convolutional encoders, thus increasing the decoding complexity which will make some quasigroup error-correcting systems to be impractical for implementation. Hence, it seems obvious to ask the following question:

Open problem #3: Design error-correcting system based on quasigroups with acceptable decoding complexity

References

[1] D. Gligoroski, S. Markovski, L. Kocarev, “Error-Correcting Codes Based on Quasigroups,” *Proceedings of 16th International Conference on Computer Communication and Networks*, pp. 16-172, August 13-16, 2007

[2] Lin CostelloP. R. Adry and M. A. H. Dempster, *Introduction to Optimisation Methods*. Chapman and Hall Ltd, London, 1974.

[3] A. J. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *IEEE Transactions on Information Theory* **13** (2): 260–269, April, 1967.

[4] G. D. Forney, "The Viterbi Algorithm," *IEEE Transactions on Information Theory* **61** (3): 268–278, March, 1973.

[5] L.Bahl, J.Cocke, F.Jelinek, and J.Raviv, "Optimal Decoding of Linear Codes for minimizing symbol error rate", IEEE Transactions on Information Theory, vol. IT-20(2), pp.284-287, March 1974.

[6] "Convolutional Codes" online: wikipedia.org/wiki/Convolutional_code. Accessed at: 27.05.2011.