UW Student Seminars: Automatic Sequences

Laindon Burnett

I am someone in two states at the same time, superimposed over one another. I am both a Computer Science and Pure Mathematics student here at the University of Waterloo. As such, the motivation for this talk is the interactions between mathematics and computer science: the bridge between my two worlds, so to speak. Most do not realise the extent of the relationship between areas of theoretical computer science and discrete mathematics. One such area of interest, which we will investigate today, is the theory of automatic sequences.

This talk will begin with a brief overview behind the theory of words in mathematics as well as the theory of finite automata in theoretical computer science. After this, we will define what an automatic sequence is, prove some fundamental theorems about them, and investigate some of their more intriguing properties. The majority of information presented comes from the text "Automatic Sequences: Theory, Applications, Generalisations" by Jean-Paul Allouche and the University of Waterloo's own Jeffrey Shallit, from the department of Computer Science.

The principle motivating sequence for this talk is the Thue-Morse sequence, defined as a sequence $\{a_n\}$ where $a_1 = 0$ and for $n \geq 2$, $a_n = a_{n-1}(\neg a_{n-1})$, where $\neg$ is the Boolean complement. So the first few terms are $0, 01, 0110, 01101001, ...$ This sequence has many fascinating properties to explore.

1 The Theory of Words

We presume the notion of a symbol and do not define it mathematically. For practical purposes, we will not be using any outlandish symbols over the course of this seminar.

Definition 1.1. An *alphabet* is a non-empty set of symbols, denoted Σ. In particular, Σ_k denotes the alphabet $\{0, 1, ..., k\}$.

Definition 1.2. A *word* is a finite or infinite collection of symbols from an alphabet, juxtaposed via concatenation. We typically use lowercase letter such as w to denote finite words. The set of finite words of an alphabet Σ is denoted Σ^* and the set of non-empty finite words is denoted Σ^+. Then *length* of a word w is denoted $|w|$. The empty word, denoted ϵ, is part of any language over any alphabet, and $|\epsilon| = 0$. Finally, if w is a word, then w^R denotes the *reverse* of w.

Definition 1.3. A *language* L over Σ is a finite or infinite set of words formed from the alphabet Σ, that is, a subset of Σ^*.

Definition 1.4. Let Σ and Δ be alphabets. A *morphism* h is a map $h : \Sigma^* \to \Delta^*$ which is such that $h(xy) = h(x)h(y)$ for all $x, y \in \Sigma^*$. Note that once a morphism is defined on Σ, it can be extended to one for Σ^*, so by convention when we define a morphism, we do so on Σ rather than Σ^*. Further, we can iterate any morphism h by letting $h^0(x) = x$ and for $i \geq 1$, $h^i = h(h^{i-1}(x))$.

Example 1.5. The Thue-Morse sequence is in fact a morphism. Specifically, it is the morphism μ on the alphabet $\Sigma = \{0,1\}$ such that $\mu(0) = 01$ and $\mu(1) = 10$. Then the nth term in the sequence is $\mu^n(0)$.

2 Finite State Automata

Much of theoretical computer science concerns the properties of the automaton: any mathematical object which takes input and applies transition functions to this input, producing a final state.

Definition 2.1. We now define a Deterministic Finite Automaton, or DFA. This is a rather terse definition, best explained first in words rather than in mathematics. A DFA takes a word as input, and either accepts or rejects this word. It begins at an initial state, and transitions between states based on the symbols in the input word and its own *transition function.* If after reading all symbols in the word, the DFA is in a state called an *accepting state*, then the input is accepted. Otherwise, the input is rejected.

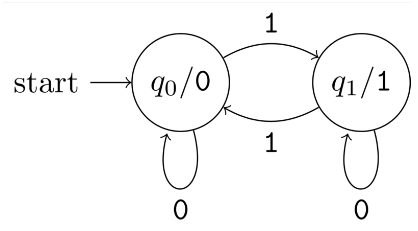
In terms of a formal mathematical definition, a DFA M is defined as a 5-tuple $M = (Q, \Sigma, \delta, q_0, F)$ where
Q is a finite set of states
Σ is a finite input alphabet
$\delta : Q \times \Sigma \to Q$ is the transition function
$q_0 \in Q$ is the initial state
$F \subseteq Q$ is the set of accepted states

Any DFA can be represented by a *transition diagram*, where states are represented using circles, accepted states by double circles, and the transition function by arrows indicating the change of state upon reading a symbol.

Definition 2.2. We can extended the principle of a DFA to a Deterministic Finite Automaton with Output, or DFAO, which is a DFA that also outputs a word from Δ for every accepted state according to the output function $\tau : Q \to \Delta$.

More formally, a DFAO M is a 6-tuple $M = (Q, \Sigma, \delta, q_0, F, \tau)$ with the first five elements defined as before. To draw a transition diagram, two circles are no longer needed for accepted states, but rather one circle and the symbol to be outputted. A k-DFAO is a DFAO which has $\Sigma = \Sigma_k$.

Example 2.3. Below is the transition diagram for a DFA representing the Thue-Morse sequence:



Definition 2.4. Let $M = (Q, \Sigma, \delta, q_0, \Delta, \tau)$ be a DFAO. If a function $f : \Sigma^* \to \Delta$ can be computed as $f_M(w) = \tau(\delta(q_0, w))$, it is called a *finite state function.*

Theorem 2.5. *Let $f : \Sigma^* \to \Delta$ be a finite state function. Then so is the function f^R defined by $f^R(w) = f(w^R)$.*

Proof: If f is a finite state function, then there is a DFAO $M = (Q, \Sigma, \delta, q_0, \Delta, \tau)$ such that $f = f_M$. Define a new DFAO $M' = (S, \Sigma, \delta', q_0', \Delta, \tau'$ where $S = \Delta^Q$, a set of functions from Q to Δ. Let q_0' be the function that maps q to $\tau(q)$, $\tau'(g) = g(q_0)$ for all functions $g \in S$, and for $g \in S$ and $a \in \Sigma$, $\delta'(g, a) = h$ where $h(q) = g(\delta(q, a))$. Quite the mouthful, but this is necessary for what comes next.

We now prove by induction on $|w|$ that $\delta'(q_0', w) = h$, where h is the function mapping q to $\tau(\delta(q, w^R))$. This is always true for $|w| = 0$, where we have $w = \epsilon$. Now suppose the result is true for $|w| = n$; we prove it for $|w| = n + 1$. Let $w = xa$, where $|x| = n$. Then $\delta'(q_0', xa) = \delta'(\delta'(q_0', x), a) = \delta'(g, a) = h$, where, by induction, g is the function that maps q to $\tau(\delta(q, x^R))$. Then $h(q) = g(\delta(q, a)) = \tau(\delta(\delta(q, a), x^R)) = \tau(\delta(q, ax^R)) = \tau(\delta(q, (xa)^R)) = \tau\delta(q, w^R)$. Since $\tau'(h) = h(q_0)$, it now follows that M' computes the function f^R.

3 Automatic Sequences

We are now ready to get into the main topic of the seminar, automatic sequences. The previous ideas are all things that are necessary prerequisites in order to understand such sequences.

Definition 3.1. A sequence $(a_n)_{n \geq 0}$ over some finite alphabet Δ is called *k-automatic* if there exists a k-DFAO $M = (Q, \Sigma_k, \delta, q_0, \Delta, \tau)$ for which $a_n = \tau(\delta(q_0, w))$ for all $n \geq 0$ and all $|w|_k = n$.

Theorem 3.2. *The Thue-Morse sequence is 2-automatic according to M defined in the earlier example.*

Proof: We can redefine the Thue-Morse sequence as being the number of ones in the binary representation of n, modulo 2. In fact, this was Thue's original definition for this sequence. We will proceed by induction. Consider $\tau(\delta(q_0, 0)) = 0$, and there are 0 ones in the binary representation of 0. Similarly, $\tau(\delta(q_0, 1)) = 1$, and there is 1 one in the binary representation of 1.

Now assume that $\delta(q_0, w) = q_0$ for some word w with $|w| \geq 1$. Then w has 0 ones in its binary representation, modulo 2. Then we have $\tau(\delta(q_0, w0)) = 0$ and $w0$ has the same number of ones. We also have $\tau(\delta(q_0, w1)) = 1$ and $w1$ has one more one, so the number of ones is 1 modulo 2. The case for $\delta(q_0, w) = q_1$ is similar.

Thus by induction, we have that $\tau(\delta(q_0, w)) = a_n$ for all n and all $|w|_k = n$, and so the Thue-Morse sequence is indeed 2-automatic. □

There is a slight problem with our current definition of automatic sequence, namely that it requires the k-DFAO to produce the correct output for a word with any number of leading zeroes. This is extremely annoying, so we proceed to enhance our definition through a pair of nice theorems.

Theorem 3.3. *The sequence $(a_n)_{n\geq 0}$ is k-automatic if and only if there exists a k-DFAO $M = \tau(\delta(q_0,(n)_k))$ for all $n \geq 0$. Moreover, we may choose M such that $\delta(q_0, 0) = q_0$.*

Proof: The forward case follows directly from Definition 3.1. As for the converse, let $M = (Q, \Sigma_k, \delta, q_0, \Delta, \tau)$.
Define $M' = (Q', \Sigma_k, \delta', q_0', \Delta, \tau'$ by
$Q' = Q \cup \{q_0'\}$
$\delta'(q, a) = \delta(q, a)$ for all $q \in Q, a \in \Sigma_k$
$\delta'(q_0', a) = \delta(q_0, a)$ for $a \neq 0$
$\delta'(q_0', 0) = q_0'$
$\tau'(q_0') = \tau(q_0)$
$\tau'(q) = \tau(q)$ for all $q \in Q$
Then for all $i \geq 0$, $\tau'(\delta'(q_0', 0^i(n)_k)) = \tau'(\delta'(q_0', (n)_k)) = \tau'(\delta'(q_0, (n)_k))$. $\square$

We can also show that it is equally valid to pass in a word least-significant digit first rather than most significant digit first.

Theorem 3.4. *The following are equivalent: 1. $(a_n)_{n\geq 0}$ is a k-automatic sequence.*
2. There exists a k-DFAO $M = (Q, \Sigma_k, \delta, q_0, \Delta, \tau)$ such that $a_n = \tau(\delta(q_0, w^R))$ for all $n \geq 0$ and all $w \in \Sigma_k^*$ such that $|w|_k = n$.
3. There exists a k-DFAO $(Q', \Sigma_k^*, \delta', q_0', \Delta, \tau')$ such that $a_n = \tau'(\delta'(q_0', (n)_k^R))$ for all $n \geq 0$.

Proof: (1) $\implies$ (2): Let $(a_n)_{n\geq 0}$ be a k-automatic sequence. By Theorem 2.5, the function f^R that maps w^R to $a_{|w|_k}$ is a finite state function, so there does exist a DFAO that computes it.

(2) $\implies$ (3): Follows trivially.

(3) $\implies$ (1): Condition (3) implies there is a finite state function g that maps $(n)_k^R$ to a_n. Hence by Theorem 2.5, the function g^R is a finite state function, and it maps $(n)_k$ to a_n. Hence there exists a DFAO that computes g^R. By Theorem 3.3, this implies that $(a_n)_{n\geq 0}$ is k-automatic. $\square$.

From here, the reader is able to begin on a journey to explore all sorts of interesting automatic sequences. As a hint on how to explore, one such sequence can be found by folding a piece of paper in half repeatedly and looking at the hills and valleys. Enjoy the world of automatic sequences!