CIRM 2024 Research School in Discrete Mathematics and Computer Science `Walnut` exercises

Narad Rampersad[1] and Manon Stipulanti[2]

[1]*Department of Mathematics & Statistics, The University of Winnipeg, Canada,* `n.rampersad@uwinnipeg.ca`
[2] *Department of Mathematics, University of Liège, Belgium,* `m.stipulanti@uliege.be`

January 29–February 2, 2024

The material used in this list of exercises is taken from [5].

1 How to actually get your hands dirty into `Walnut`

To **start** `Walnut` under Linux, use a terminal window and the command-line interface. Go to the `bin` directory of `Walnut` and type:

```
java Main.Prover
```

You will then see a prompt that looks like

```
[Walnut]$
```

You can then enter `Walnut` commands sequentially, or paste them in.

- **Results** are stored in the directory called `Result`.
- **Word automata** (DFAO's) are stored in the directory called `Word Automata Library`.
- **Ordinary automata** (DFA's) are stored in the directory called `Automata Library`.

When **you are done** using `Walnut`, enter an EOF character (for example, CTRL-D under Linux) to stop the program. Or type `exit;` at the prompt.

2 A (not so short) summary on syntax

- **Words** made up of capital letters (other than those starting with `A` and `E`) refer to sequences (defined in terms of a DFAO) stored in the directory `Word Automata Library`. For example, `T` is the Thue-Morse sequence **t** and `F` refers to the Fibonacci word **f**.

- You index a **sequence** by natural numbers using square brackets: `T[n]` refers to the n'th symbol of **t**. It is possible for DFAO's to have multiple arguments. In this case you use multiple brackets: `SCA[i][j]` evaluates the two-dimensional sequence $\mathbf{sca}[i, j]$. In a `Walnut` command, natural numbers are always written in base 10, even though they might be represented in other ways internally.
- A complete list of those **DFAO's** provided with `Walnut` is given in [5, Section 7.3]. You can also add your own user-defined DFAO to this library by creating a text file in the right format, with the name of the DFAO being the name of the file.
- For the **logical syntax**, `E` is `Walnut`'s abbreviation for $\exists$ and `A` is the abbreviation for $\forall$. The symbol `=>` is logical implication, `&` is logical AND, `|` is logical OR, and `<=>` is logical IFF. Both the `E` and `A` can be followed by a single variable name, or a list of variables.
- **Natural numbers** can be used to index automatic sequences, and further can be part of algebraic expressions. So, for example, one can write `T[x+y]` or `x+y=2*z`. The allowed operations are addition, subtraction, multiplication by a natural number constant, and integer division by a nonzero constant (`x/c` represents $\lfloor x/c \rfloor$). When you write an algebraic expression, each variable must occur at most once. So, for example, instead of `x+y+x` write `2*x+y` in `Walnut`. Remember that the domain of variables in `Walnut` is $\mathbb{N}$, so that negative numbers cannot be used. However, subtraction can be used meaningfully with the ordinary minus symbol, provided the value of no subexpression is ever negative. Some care must be taken in this regard.
- **Sequences** take their values in $\mathbb{Z}$, the integers. While arithmetic cannot be done on sequence values directly, they can be compared for equality (e.g., `T[n]=RS[n+1]`) or inequality (e.g., `T[n]<=RS[n+1]`). Constant values of sequences can be specified by preceding the constant with the `@` sign, so that, for example, `T[n]=@0` specifies those n for which $\mathbf{t}[n] = 0$, where **t** is the Thue-Morse sequence.
- The use of `Walnut` highly depends on **numeration systems**. The phrase `?msd_3` preceding a formula specifies it should be evaluated in most-significant-digit-first format, in base 3. Here 3 can be replaced by any integer ≥ 2. One can also use `lsd` in place of `msd`, which stands for least-significant-digit-first format. (The default is `msd_2`.) Similarly, `?msd_fib` evaluates the formula in Fibonacci representation, and `?msd_trib` evaluates the formula in Tribonacci representation, and `?msd_pell` evaluates the formula in Pell representation. Unpredictable results, or errors, may occur if one attempts to mix multiple representations in the same expression. A command of the form

  ```
  ost name [0 1 2] [3 4]:
  ```

  defines an Ostrowski numeration system. This particular example defines an Ostrowski numeration system based on the continued fraction $[0, 1, 2, 3, 4, 3, 4, 3, 4, ...]$ and stores it under the name “name”. Once defined, to use it, preface a query with `?msd_name` or `?lsd_name`.
- **Formulas** can be defined and used in later work with the `def` command. For example, the formula

$$\textsc{FactorEq}(i, j, n) := \forall t < n, \mathbf{x}[i + t] = \mathbf{x}[j + t]$$

can be defined in `Walnut` for the Fibonacci word **f** as follows:

```
def ffactoreq "?msd_fib At t<n => F[i+t]=F[j+t]":
```

This creates a predicate `$ffactoreq`, that takes three arguments: i, j, n (in that order) that is true iff $\mathbf{f}[i..i+n-1] = \mathbf{f}[j..j+n-1]$, that is, if the length-n factor beginning at position i in the Fibonacci word is the same as the length-n factor beginning at position j. The predicate is represented by a DFA that takes the Fibonacci representation of (i, j, n) as input and accepts those inputs for which $FactorEq(i, j, n)$ evaluates to `TRUE`. When the predicate is used later in `Walnut`, it must be preceded by a dollar-sign. A crucial thing to remember is that the order of arguments when a predicate is invoked is alphabetical order of the free variables that appear in it when it is defined. If you do not intend to save the result for later use, you can use `eval` in place of `def`. This is particularly useful if your expression has no free variables, because then `eval` will print the result `TRUE` or `FALSE`.

- `Walnut` can also compute **linear representations** for formulas with at least two free variables. This is also done with the `def` command, but the syntax is somewhat different. If you write

  ```
  def name n "$func(i,n)":
  ```

  for example, `Walnut` produces and stores in the filename `name.mpl` a `Maple` file that defines the linear representation (v, γ, w) for the function $f(n)$ counting the number of i for which `$func(i,n)` is true. If there are infinitely many such i, the result cannot be relied upon.

- The command `inf` takes the name of an automaton as an argument, and returns `TRUE` if the automaton accepts infinitely many natural numbers, and `FALSE` otherwise.

- Formulas dealing with the representation of integers, expressed in terms of a **regular expression**, can be defined in `Walnut` using the `reg` command. The syntax is

  ```
  reg name msd_k regex:
  ```

  where `name` should be replaced by the predicate name, `k` should be replaced by the base of representation (or `fib` or `trib`) and `regex` is the regular expression. For regular expressions, the symbol `|` means union, `*` means Kleene closure, and the empy word is denoted by `()`. For example, if we are working in base 2, then the command

  ```
  reg power2 msd_2 "0*10*":
  ```

  defines a predicate `$power2` that evaluates to `TRUE` if its argument is a power of 2, and `FALSE` otherwise. If you wish to use bases larger than 10, then digits larger than 9 must be enclosed in square brackets. Regular expressions can work with tuples, by enclosing the tuple in square brackets and using commas as separators. If your regular expression is intended to represent a natural number, be sure that all representations having an arbitrary number of leading zeros (in msd- format) or trailing zeros (in lsd-format) are specified, otherwise anomalous results may occur.

- Some **automatic sequences** are built-in to Walnut. They are stored in the directory `Word Automata Library`. For instance, here is how `T` is represented in `Walnut`: it is stored in the file `T.txt`:

  ```
  msd_2
  0 0
  0 -> 0
  1 -> 1
  1 1
  0 -> 1
  1 -> 0
  ```

  The first line specifies the number system that the automaton is defined over, here base 2. Lines with two numbers and no arrow (`->`) give a state number (a natural number) first and an output (an integer) second. In this example, there are two states: a state numbered 0 with an output of 0 and a state numbered 1 with an output of 1. Finally, transitions for a state appear beneath that state and are of the form `a -> b`, where `a` is an input and `b` is a state. State numbers should begin with state 0 and consist of consecutive numbers $0, 1, 2, \ldots$.

- The `morphism` command, which allows you to create a k**-uniform morphism**. For example,

  ```
  morphism h "0->01 1->12 2->20":
  ```

  defines a morphism called h that maps $0 \mapsto 01$, $1 \mapsto 12$, and $2 \mapsto 20$. If you have defined a uniform morphism, you can "promote" it to a DFAO generating the fixed point starting with 0 of the morphism using the command `promote`. For example,

  ```
  promote T3 h:
  ```

  defines a DFAO generating $h^\omega(0)$. You can also apply a uniform morphism to a DFAO, getting a new DFAO as a result. If the morphism is h and the DFAO generates an infinite word $\mathbf{x}$ then the resulting DFAO generates $h(\mathbf{x})$. To do this, use the `image` command:

  ```
  image F2 h F:
  ```

  The resulting DFAO is defined over the same numeration system as the original DFAO. By combining these three commands, you can define an arbitrary k-uniform DFAO.

- The `Word Automata Library` of `Walnut` is the place to define your **own DFAO's**. Give your DFAO a name of capital letters (but not starting with `A` or `E`, which are reserved symbols for the universal and existential quantifier) such as `GHI` and then store it in the file `GHI.txt`. It can then be invoked inside `Walnut`. The first line should say what the numeration system of the automaton is. For example, a first line of `msd_3` states that inputs are over the alphabet $\{0, 1, 2\}$ and are read msd-first. Use `msd_fib` for the Fibonacci numeration system and `msd_trib` for the Tribonacci numeration system. If you do not wish to specify a numeration system, but merely the size of the alphabet, you can use notation like `{0,1}`, for example, to indicate a binary alphabet.

- The `Automata Library` directory is the place to define your **own DFA's**. Here each state should have an output of `1` (accepting) or `0` (non-accepting).
- It is possible to **combine two different numeration systems** in one `Walnut` expression, but it is not possible to perform arithmetic operations on integers specified in two different systems. To specify a numeration system, use an expression like `?msd_3`. The scope of such an expression is described as follows: if it appears outside parentheses, it modifies everything that follows. If it appears inside parentheses, the scope is restricted to the area between the command and the closing parenthesis. It is also not possible to mix `lsd` and `msd` representations in the same `Walnut` command.
- In `Walnut` you can define your **own numeration system**. To do so, you will have to provide two automata and store them in the `Custom Bases` directory: an automaton recognizing all valid representations in the numeration systems, and an adder. The file with the first automaton should be called `msd_name.txt`, where `name` is replaced by the name of the numeration system. The adder should be called `msd_name_addition.txt`, and check the relation $x + y = z$.

3 Basics

Exercise 1. Show that the sum of two even numbers is even.

Exercise 2. Show that the Rudin-Shapiro word (stored under `RS` in `Walnut`) is equal to the fixed point $\tau(h^\omega(0))$ where

$$h : 0 \mapsto 01, 1 \mapsto 02, 2 \mapsto 31, 3 \mapsto 32 \quad \text{and} \quad \tau : 0 \mapsto 0, 1 \mapsto 0, 2 \mapsto 1, 3 \mapsto 1.$$

Exercise 3. Consider the characteristic sequence of powers of 3, i.e.,

$$\mathbf{p_3} = 0101000001000000000000000000100 \cdots .$$

Upload a DFAO into `Walnut` generating this sequence (and check that your DFAO encodes the right sequence).

4 Combinatorics on words

Exercise 4. Find the starting positions of all occurrences of the factor 01 in the Thue–Morse word. In `Walnut`, the Thue–Morse word is stored under `T`.

Exercise 5. What are the possible distances between all occurrences of 1 in the Fibonacci word? In `Walnut`, the Fibonacci word is stored under `F`.

A *run* is a non-empty block consisting of a repeated single sympbol a. A run is *maximal* if it cannot be extended to the left or to the right.

Exercise 6. Show that the longest run of 0's in the Rudin-Shapiro sequence has length 4. In `Walnut`, the Rudin-Shapiro word is stored under `RS`. (*Hint*: First, in a word $\mathbf{x}$ and for a letter a, determine a formula $\textsc{IsRun}(i, n)$ asserting that $\mathbf{x}[i..i+n-1]$ is a maximal run of n letters a. Then, find a formula to determine the longest runs occurring in a sequence.)

Given a sequence $\mathbf{x}$, its *run-length encoding* is an expression of the form $\mathbf{x} = \prod_{i\geq 1} a_i^{e_i}$, where each $e_i \geq 1$ and $a_i \neq a_{i+1}$ for all $i \geq 1$. The set $\{e_1, e_2, e_3, \cdots\}$ is the set of *run lengths* of $\mathbf{x}$.

Exercise 7. The *second-bit sequence* $\mathbf{sb} = 00010011000011110\cdots$ consists of the second digit in the base-2 expansion of n starting with the most significant digit first (if it is not defined, it is set to be 0). Compute the set of maximal run lengths in $\mathbf{sb}$. Show that they are of length 3 and length 2^i for $i \geq 0$. In `Walnut`, the second-bit sequence is stored under `SB`.

Exercise 8. Check that the Thue–Morse is not ultimately periodic.

The *order* of a square xx is half its length $|x|$. We define the Fibonacci numbers $(F_n)_{n\geq 0}$ by $F_0 = 0$, $F_1 = 1$, and $F_n = F_{n-2} + F_{n-1}$ for all $n \geq 2$.

Exercise 9. Show Séébold's result [4]: All squares in the Fibonacci word $\mathbf{f}$ are of order F_n for some $n \geq 2$. Furthermore, for all $n \geq 2$, there exists a square of order F_n in $\mathbf{f}$. Also show that $\mathbf{f}$ contains arbitrarily large squares beginning at each position.

A word is *primitive* if it is non-empty and a non-power. Another characterization that makes this property expressible in `Walnut` is the following: a word w is primitive if and only if no non-trivial rotation of w equals w.

Exercise 10. Show that every non-empty prefix of the Thue–Morse word is primitive. (*Hint*: Consider creating a formula that checks equality between the length-n factor $\mathbf{t}[i..i+n-1]$ and its rotation $\mathbf{t}[i+j..i+n-1]\mathbf{t}[i..i+j-1]$.)

Exercise 11. Stewart's choral sequence $\mathbf{sc} = 001001011\cdots$ is given by the fixed point of the morphism $0 \mapsto 001, 1 \mapsto 011$. Alternatively, we have $\mathbf{sc}[n] = 1$ if and only if the base-3 representation of n ends in a word of the form $21\cdots 1$ for some non-negative number of 1's. Show that Stewart's choral word $\mathbf{sc}$ avoids the pattern $xxyyxx$, where x is non-empty and y is allowed to be empty. (*Hint*: Set $|x| = m$ and $|y| = n$.)

For a rational number $p/q \geq 1$,

- a (p/q)*-power of period* n is a word of length $(p/q)n$ and period n;
- a $(p/q)^+$*-power of period* n is a word of length $> (p/q)n$ and period n.

Exercise 12. Show that the fixed point, starting with 0, of the morphism

$$0 \mapsto 0100110, \quad 1 \mapsto 1011001$$

is $(5/2)^+$-power-free.

Two words are *conjugates* if one is a rotation of the other. A factor y of $\mathbf{x}$ is *cyclically isolated* if no conjugate of y, except y itself, belongs to $\mathbf{x}$.

Exercise 13. Find the integers n for which there is a cyclically isolated factor of length n in the Thue–Morse word.

Exercise 14. Show that the set of lengths of palindromic factors of the Thue-Morse word is $\{1, 3\} \cup \{2n \mid n \geq 0\}$.

A sequence is *recurrent* if every factor that occurs, occurs infinitely often. A sequence $\mathbf{x}$ is *uniformly recurrent* if it is recurrent, and for every factor w there exists a constant c such that every occurrence of w in $\mathbf{x}$ is followed by another occurrence at distance at most c.

Exercise 15. The Cantor integers are those integers whose base-3 representation contains no letter 1. The Cantor sequence $\mathbf{ca} = 101000101000000000\cdots$ is the corresponding characteristic sequence. Show that the Cantor sequence (stored under `CA` in `Walnut`) is recurrent but not uniformly recurrent.

A word is called *Lyndon* if it is primitive and also lexicographically less than any of its proper nonempty suffixes.

Exercise 16. Show that there is a Lyndon factor of length n in the Thue–Morse sequence if and only if $n = 2^k$ or $n = 3 \cdot 2^k$ or $n = 5 \cdot 2^k$ for $k \geq 0$.

A word x is *rich* if and only if it has $|x| + 1$ distinct palindromic factors. Another characterization of rich words is the following [2, Prop. 3]: a finite word x is rich if every prefix p of x has a palindromic suffix s that occurs only once in p.

Exercise 17. Show that every factor of the period-doubling sequence (stored under `PD` in `Walnut`) is rich.

A word x is called *balanced* if the inequality $||y|_a - |z|_a| \leq 1$ holds for all identical-length factors y, z of x and all letters a of the alphabet. (Here $|y|_a$ denotes the number of occurrences of the letter a in the word y.) A characterization in the case of the binary alphabet is the following [1]: a binary word w is balanced if and only if there exists a word v such that both $0v0$ and $1v1$ are factors of w.

Exercise 18. Show that every factor of the Fibonacci word is balanced.

Exercise 19. For which integers n do the Thue–Morse sequence and the Rudin-Shapiro sequence have common length-n factors?

5 Regular sequences and enumeration problems

Exercise 20. For all $n \geq 0$, let $f(n)$ be the number of (scattered) subsequences of the form $0x1y0$ (where x, y are binary words) in the length-n prefix of the Thue–Morse word $\mathbf{t}$. Give a linear representation of the sequence $(f(n))_{n\geq 0}$.

Exercise 21. For all $n \geq 0$, let $g(n)$ be the number of length-n palindromes in the period-doubling sequence $\mathbf{sp}$. Give a linear representation of the sequence $(g(n))_{n\geq 0}$. (*Hint*: Consider writing a function that checks whether a palindromic factor occurs for the first time.)

Exercise 22. For all $n \geq 0$, let $h(n)$ be the number of order-n squares in the Fibonacci word $\mathbf{f}$. Give a linear representation of the sequence $(h(n))_{n\geq 0}$.

6 Synchronized sequences

Let $\mathbf{t} = t_0 t_1 t_2 \cdots$ be the Thue–Morse word. The *odious* (resp., *evil*) *numbers* $1, 2, 4, 7, 8, 11, \ldots$ are those integers n for which $t_n = 1$ (resp, $t_n = 0$).

Exercise 23. Let o_n be the nth odious number, with $o_0 = 1$. Use the fact that $o_n = 2n + 1 - t_n$ for all $n \geq 0$ to show that the sequence $(o_n)_{n \geq 0}$ is 2-synchronized.

Exercise 24. For each $n \geq 0$, let s_n be the position of the first occurrence of two letters 1 exactly n symbols apart in the Thue–Morse sequence $\mathbf{t}$. Show that the sequence $(s_n)_{n \geq 0}$ is 2-synchronized.

Exercise 25. Consider the second-bit sequence $\mathbf{sb}$. For all $n \geq 0$, define $f_0(n)$ to be the starting position of the first run of 0's of length at least n in $\mathbf{sb}$. Show that $(f_0(n))_{n \geq 0}$ is 2-synchronized and verify that

$$f_0(n) = \begin{cases} 0, & \text{if } n \leq 3; \\ 2^{k+2}, & \text{if } 2^k < n \leq 2^{k+1}, k \geq 1, n \geq 4. \end{cases}$$

The *abelian complexity* of an infinite sequence $\mathbf{x}$ counts the number of distinct subwords, up to abelian equivalence of factors. For instance, the abelian complexity of the Thue–Morse sequence $\mathbf{t}$ at $n = 2$ is 3, because among the four factors 00, 01, 10, and 11, the factors 01 and 10 are (abelian) equivalent.

Exercise 26. Show that abelian complexity $\mathbf{p}_{\mathsf{ab}}$ of the Thue–Morse sequence $\mathbf{t}$ satisfies the following [3]:

$$\mathbf{p}_{\mathsf{ab}}(n) = \begin{cases} 1, & \text{if } n = 0; \\ 2, & \text{if } n \geq 1 \text{ and } n \text{ is odd}; \\ 3, & \text{if } n \geq 2 \text{ and } n \text{ is even}. \end{cases}$$

(*Hint*: Create a DFA that asserts that $\mathbf{t}[i..i + n - 1]$ is a novel occurrence of a length-n factor with that Parikh vector, and obtain the linear representation for the number of such i. Then, if (v, M_0, M_1, w) is the linear representation you get, show that the matrix products $v M_1 M_{x_1} M_{x_2} \cdots M_{x_\ell} w$, $x_i \in \{0, 1\}$, can only take two values.)

7 Additive number theory

Exercise 27. Show that a natural number is the sum of exactly two evil numbers if and only if it is neither 2, 4, nor of the form $2 \cdot 4^i$ for $i \geq 0$.

Exercise 28. Show that every natural number $n > 10$ is the sum of three distinct evil numbers.

References

[1] E. Cohen, G. A. Hedlund, Sequences with minimal block growth, *Math. Systems Theory* **7** (1973), 138–153.

[2] X. Droubay, J. Justin, G. Pirillo, Episturmian words and some constructions of de Luca and Rauzy, *Theoret. Comput. Sci.* **255** (2001), no. 1-2, 539–553.

[3] G. Richomme, K. Saari, L. Q. Zamboni, Abelian complexity of minimal subshifts, *J. Lond. Math. Soc. (2)* **83** (2011), no. 1, 79–95.

[4] P. Séébold, *Propriétés combinatoires des mots infinis engendrés par certains morphismes (Thèse de 3ème cycle)*, PhD thesis, Université P. et M. Cruie, Institut de Programmation, Paris, 1985.

[5] J. Shallit, *The logical approach to automatic sequences–exploring combinatorics on words with `Walnut`*, London Math. Soc. Lecture Note Ser., 482 Cambridge University Press, Cambridge, 2023, xv+358 pp.