Dyck Words, Pattern Avoidance, and Automatic Sequences

Lucas Mol, Thompson Rivers University
Narad Rampersad, University of Winnipeg
Jeffrey Shallit, University of Waterloo

WORDS 2023

Plan

Dyck Words

- A Dyck word is a string of balanced parentheses.
  - 0 – left paren
  - 1 – right paren

  E.g.,
  - $001011 = (()())$ is Dyck
  - $0110 = ())($ is not
- Formally, x is Dyck if
  - $x = \varepsilon$,
  - $x = 0y1$ for some Dyck word y, or
  - $x = yz$ for some Dyck words y and z.

Balance and Nesting Level

- The balance of x is defined by

$$B(x) = |x|_0 - |x|_1.$$

- The word x is Dyck iff

$$B(x) = 0 \text{ and } B(x') \geq 0 \text{ for all prefixes } x' \text{ of } x.$$

- The nesting level of a Dyck word x, denoted $N(x)$, is the deepest level of parenthesis nesting in x, e.g.,

$$N(001011) = 2.$$

- More generally,

$$N(x) = \max\{B(x') : \; x' \text{ is a prefix of } x\}.$$

QUESTIONS

- What repetitions must appear in long Dyck words? What repetitions can be avoided? What is the relationship between avoidable repetitions and nesting level?
- Can `Walnut` be used to prove statements about the Dyck factors of certain automatic sequences?

Plan

Introduction

Repetitions and Dyck words

Dyck factors of some automatic sequences

REPETITIONS

- The exponent of a word is its length divided by its smallest period, e.g.,
  - $\texttt{alfalfa} = (\texttt{alf})^{7/3}$ has exponent $7/3$
  - $\texttt{valtavalta} = (\texttt{valta})^2$ has exponent 2
- A word is α-power-free if it contains no factors of exponent greater than or equal to α.
  - E.g., the fixed point of $g = [012, 02, 1]$ is 2-power-free.
- A word is α^+-power-free if it contains no factors of exponent greater than α.
  - E.g., the Thue-Morse word is 2^+-power-free.

Theorem: A characterization of overlap-free Dyck words.

Corollary: There are arbitrarily long overlap-free Dyck words.

Sketch of Proof:

- Let $g = [012, 02, 1]$, and let $\mathbf{s} = g^{\omega}(0)$.
- Let $h = [01, 0011, 001011]$.
- Let x be a prefix of $\mathbf{s}$ ending in 10.
- Then $h(x)$ and $0h(x)1$ are overlap-free Dyck words.

Note: These words have nesting level at most 3.

Theorem: If w is a $\frac{7}{3}$-power-free Dyck word, then $N(w) \leq 3$.

Theorem: There are $\frac{7}{3}^+$-power-free Dyck words of every nesting level.

Idea of Proof:

- We sketch the simpler proof that there are *cube-free* Dyck words of every nesting level.
- Define $f = [001, 011]$.
- It is well-known that f is cube-free.
- Applying f preserves the Dyck property, and increases the nesting level by one.
- By induction, for all $t \geq 0$, the word $f^t(01)$ is a cube-free Dyck word of nesting level $t + 1$.

Summary: Repetitions and Dyck words

- There are arbitrarily long overlap-free Dyck words, but they have small nesting level.
- Dyck words of large nesting levels only become attainable when we allow $7/3$-powers.

Plan

Introduction

Repetitions and Dyck words

Dyck factors of some automatic sequences

Walnut

- `Walnut` can be used to prove statements, written in a certain first-order logic, about automatic sequences.
- But the language of Dyck words is not definable in this first-order logic!
- So `Walnut` cannot directly handle the Dyck factors of all automatic sequences...

Running-sum synchronized sequences

- For a binary k-automatic sequence $\mathbf{s} = (s(n))_{n\geq 0}$, define its running-sum sequence by

$$v(n) = \sum_{0\leq i<n} s(i).$$

- We say that $\mathbf{s}$ is running-sum synchronized if there is a DFA accepting, in parallel, the base-k representations of n and $v(n)$.

Theorem: `Walnut` can handle the Dyck factors of running-sum synchronized sequences!

An Example: Thue-Morse

The Thue-Morse sequence is running-sum synchronized.

$$0\ 1\ 1\ 0\ 1\ 0\ 0\ 1\ \cdots$$

An Example: Thue-Morse

The Thue-Morse sequence is running-sum synchronized.

$$0\ 1\ 1\ 0\ 1\ 0\ 0\ 1\ \cdots$$

$$0$$

An Example: Thue-Morse

The Thue-Morse sequence is running-sum synchronized.

$$0\ 1\ 1\ 0\ 1\ 0\ 0\ 1\ \cdots$$

$$0\ 1$$

An Example: Thue-Morse

The Thue-Morse sequence is running-sum synchronized.

$$0\ 1\ 1\ 0\ 1\ 0\ 0\ 1\ \cdots$$

$$0\ 1\ 2$$

An Example: Thue-Morse

The Thue-Morse sequence is running-sum synchronized.

$$0\ 1\ 1\ 0\ 1\ 0\ 0\ 1\ \cdots$$

$$0\ 1\ 2\ 2$$

An Example: Thue-Morse

The Thue-Morse sequence is running-sum synchronized.

$$0\ 1\ 1\ 0\ 1\ 0\ 0\ 1\ \cdots$$

$$0\ 1\ 2\ 2\ 3$$

An Example: Thue-Morse

The Thue-Morse sequence is running-sum synchronized.

$$0\ 1\ 1\ 0\ 1\ 0\ 0\ 1\ \cdots$$

$$0\ 1\ 2\ 2\ 3\ 3$$

An Example: Thue-Morse

The Thue-Morse sequence is running-sum synchronized.

$$0\ 1\ 1\ 0\ 1\ 0\ 0\ 1\ \cdots$$

$$0\ 1\ 2\ 2\ 3\ 3\ 3$$

AN EXAMPLE: THUE-MORSE

The Thue-Morse sequence is running-sum synchronized.

$$0\ 1\ 1\ 0\ 1\ 0\ 0\ 1\ \cdots$$

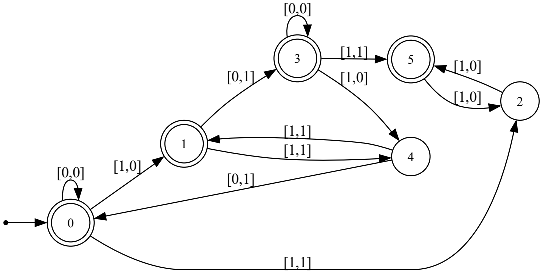
$$0\ 1\ 2\ 2\ 3\ 3\ 3\ 4\ \cdots$$

An Example: Thue-Morse

The Thue-Morse sequence is running-sum synchronized.

$$0\ 1\ 1\ 0\ 1\ 0\ 0\ 1\ \cdots$$

$$0\ 1\ 2\ 2\ 3\ 3\ 3\ 4\ \cdots$$



- $[1,1][1,0]$ is accepted, since $v(3) = 2$.
- $[1,0][1,0]$, $[1,0][1,1]$, and $[1,1][1,1]$ are not accepted!

An Example: Thue-Morse

The DFA on the previous slide was built in `Walnut` as follows:

```
def even "Ek n=2*k":     # accepts even numbers

def odd "Ek n=2*k+1":    # accepts odd numbers

def V "($even(n) & 2*x=n) |
       ($odd(n) & 2*x+1=n & T[n-1]=@0) |
       ($odd(n) & 2*x=n+1 & T[n-1]=@1)":
# accepts n and v(n) in parallel
```

An Example: Thue-Morse

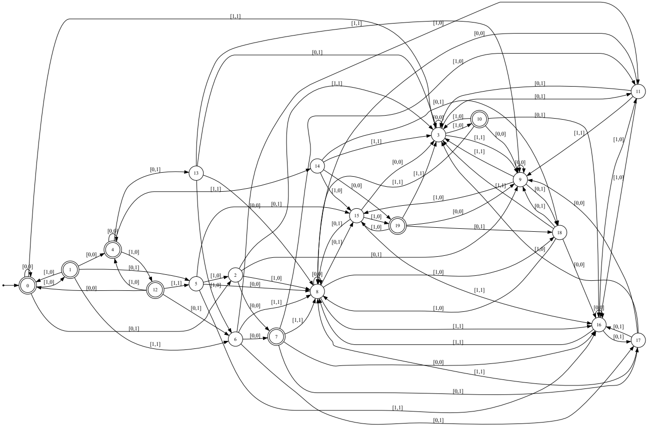
We can now build an automaton that identifies the Dyck factors of Thue-Morse:

```
def N1 "Ey,z $V(i,y) & $V(i+n,z) & x+y=z":
# accepts (i,n,x) if T[i..i+n−1] has x 1's

def N0 "Ey $N1(i,n,y) & n=x+y":
# accepts (i,n,x) if T[i..i+n−1] has x 0's

def Dyck "(Ew $N0(i,n,w) & $N1(i,n,w)) &
          At,y,z (t<n & $N0(i,t,y) & $N1(i,t,z)) => y>=z":
# accepts (i,n) if T[i..i+n−1] is Dyck
```

AN EXAMPLE: THUE-MORSE



The automaton recognizing Dyck factors of Thue-Morse!

An Example: Thue-Morse

Now we can prove statements about Dyck factors of TM.

- TM has Dyck factors of all even lengths.

  We run the command

  ```
  eval AllLengths "An $even(n) => Ei $Dyck(i,n)":
  ```

  and `Walnut` returns `TRUE`.

- Every Dyck factor of TM has nesting level at most 2.

  We run the commands

  ```
  def Bal "Ey,z $N0(i,n,y) & $N1(i,n,z) &
           ((y<z & x=0) | (y>=z & y=x+z))":
  def Nest "Em (m<n) & $Bal(i,m,x) &
            Ap,y (p<n & $Bal(i,p,y)) => y<=x":
  eval MaxNest "Ai,n,x ($Dyck(i,n) & $Nest(i,n,x)) => x<=2":
  ```

  and `Walnut` returns `TRUE`.

- We can also count Dyck factors of TM!

Summary: Automatic Sequences

- `Walnut` cannot directly handle the Dyck factors of all automatic sequences.
- `Walnut` can handle the Dyck factors of automatic sequences that are running-sum synchronized.

OUTLOOK

Some possible directions for future work:

- Extend to Dyck words with two or more types of parens.
- Develop techniques to recognize/characterize the Dyck factors of words that are not running-sum synchronized.


Thank you!