

Ensuring the Productivity of Infinite Structures¹

Alastair Telford² David Turner

September 1997
Revised March 1998

¹This work was supported by the UK Engineering and Physical Sciences Research Council grant number GR/L03279. We would also like to thank members of the Theoretical Computer Science group at the University of Kent at Canterbury for their discussions in connection with this work, particularly Andy King, Erik Poll and Simon Thompson. Eduardo Giménez, of INRIA, France, has also been most helpful in explaining his ideas and how they have been implemented within the Coq system.

²*E-Mail:* A.J.Telford@ukc.ac.uk. *Tel:* +44 1227 827590. *Fax:* +44 1227 762811.
ESFP webpage: <http://www.cs.ukc.ac.uk/people/staff/ajt/ESFP/>

$filter(x:y, 0, m) \rightarrow 0:filter(y, m, m)$

$filter(x:y, S(n), m) \rightarrow x:filter(y, n, m)$

$sieve(0:y) \rightarrow sieve(y)$

$sieve(S(n):y) \rightarrow S(n):sieve(filter(y, n, n))$

$nats(n) \rightarrow n:nats(S(n))$

$primes \rightarrow sieve(nats(S(S(0))))$

$$\mathit{ones} \rightarrow 1: \mathit{ones}$$
$$\mathit{zeros} \rightarrow 0: \mathit{zeros}$$
$$\mathit{alt} \rightarrow 0: 1: \mathit{alt}$$
$$\mathit{zip}(n:x, y) \rightarrow n: \mathit{zip}(y, x)$$

$\mathit{zip}(\mathit{zeros}, \mathit{ones})$ and alt rewrite in infinitely many steps to the same infinite normal form

1. $zip(zeros, ones) = alt$
2. $inv(inv(x)) = x$
3. $zip(odd(x), even(x)) = x$
4. $odd(zip(x,y)) = x$
5. $even(zip(x,y)) = y$



$morse = 1:0:zip(tail(morse), inv(tail(morse)))$

THE MORSE AND TOEPLITZ SEQUENCE

morse 1001011001101001
toeplitz 101110101011101110

toeplitz = diff morse

morse: DOL system $1 \rightarrow 10, 0 \rightarrow 01$, start 1

toeplitz: DOL system $1 \rightarrow 10, 0 \rightarrow 11$, start 1

toeplitz T is defined by

$T = 1 : \text{zip}(\text{inv}(T), \text{ones})$

101110101011101110
101110101

On the productivity of recursive definitions.

The purpose of this note is to summarize what we have learned at the last two sessions of the Tuesday Afternoon Club. We shall discuss conditions under which finite expressions can be said to represent infinite sequences. The simplest example of such a finite expression is

ones

where ones is defined recursively by

def ones = 1 : ones

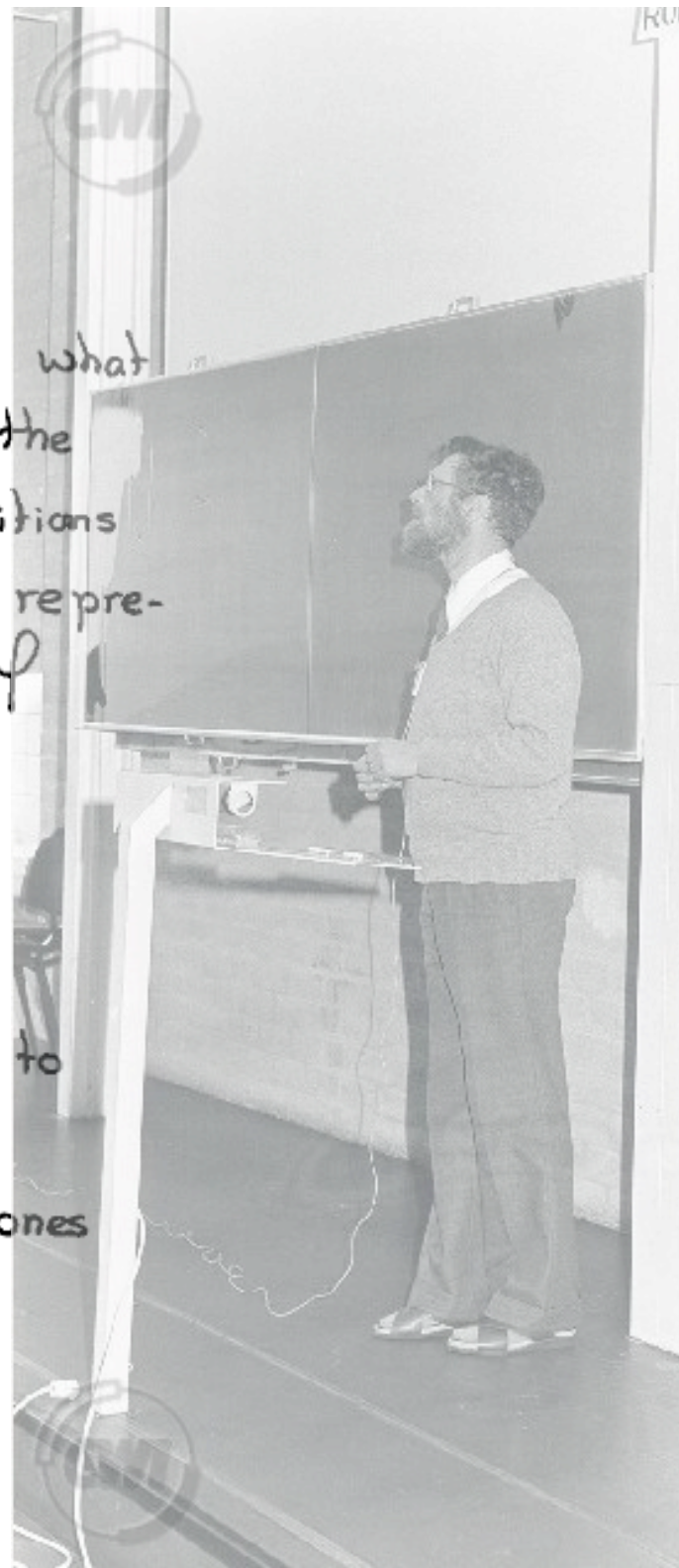
(Here we have borrowed from SASL the colon to denote concatenation.)

Repeated application of the definition of ones gives

ones

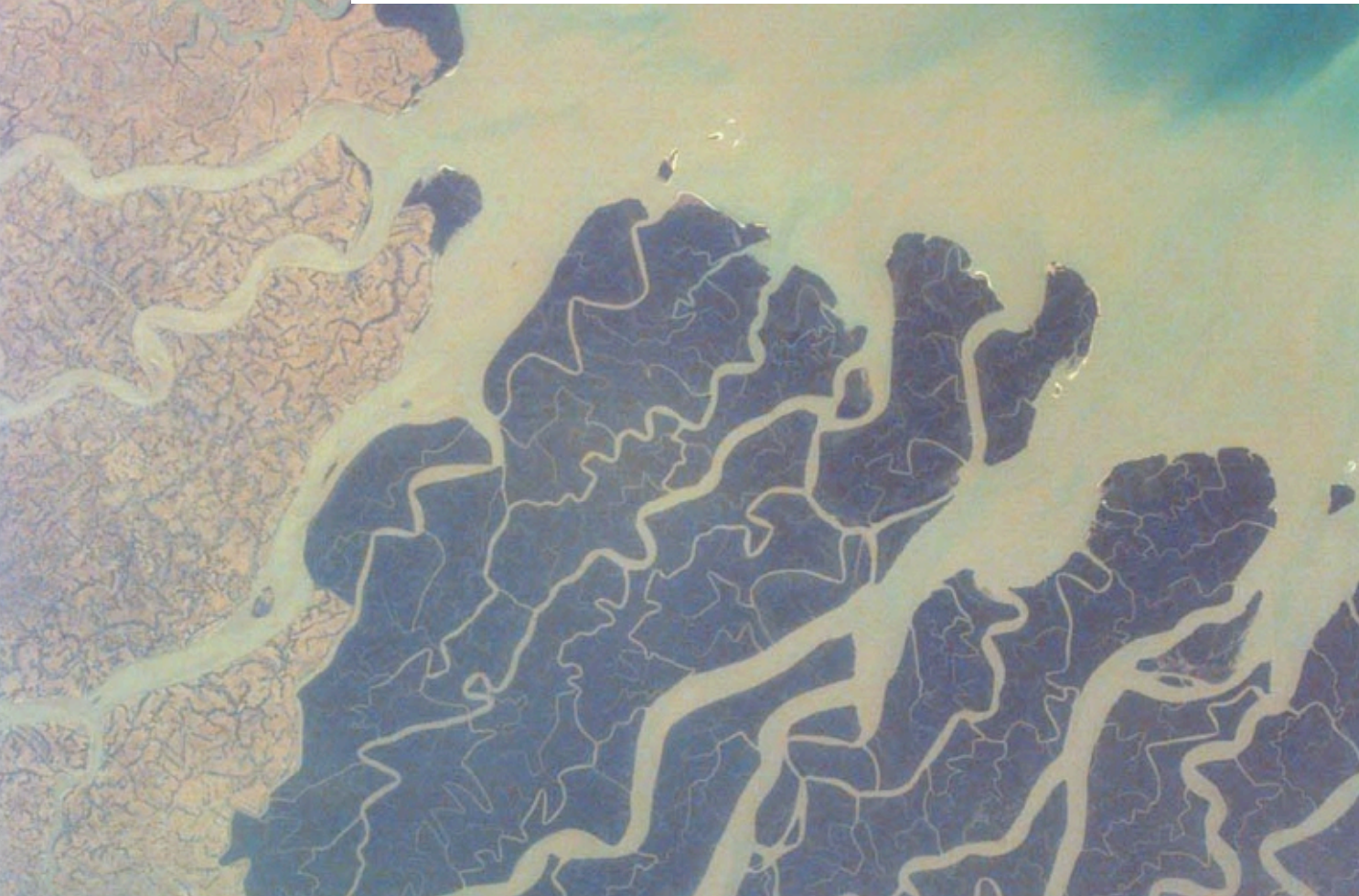
= 1 : ones

= 1 : 1 : ones, and so on.



I have hesitated for a long time

between "ongoing sequences" and "continuing sequences" — not being too pleased with either of the two—. After consultation of all my English and American dictionaries I concluded that "continued concatenation" should do the job.



Streams

Productivity =

*SN[∞] plus all normal forms must be
constructor normal forms*

Thue-Morse sequence:

$M = 1001 \ 0110 \ 01101001 \ 0110100110010110 \dots$

$M = 1001 \ 0110 \ 01101001 \ 0110100110010110 \dots$

$M = \text{zip } M \text{ inv}(M)$

NB. this does not define M !
Two solutions

$$X \rightarrow 0:\text{tail}(X)$$
$$\text{tail}(x:\sigma) \rightarrow \sigma$$
$$X \rightarrow 0: \text{tail } X$$
$$\rightarrow 0: \text{tail } (0:\text{tail } X)$$
$$\rightarrow 0: \text{tail } X)$$

JOERG'S EXAMPLE

$$\begin{aligned} J &\rightarrow 0:1: \text{ev}(J) \\ \text{ev}(x:\sigma) &\rightarrow x:\text{od}(\sigma) \\ \text{od}(x:\sigma) &\rightarrow \text{ev}(\sigma) \end{aligned}$$
$$\begin{aligned} J &\rightarrow 01 \text{ ev } J \\ \text{ev } x &\rightarrow x \text{ od} \\ \text{od } x &\rightarrow \text{ev} \end{aligned}$$
$$\begin{aligned} J &\rightarrow 01 \text{ ev } J \\ &\rightarrow 01 \text{ ev } 01 \text{ ev } J \\ &\rightarrow 010 \text{ od } 1 \text{ ev } J \\ &\rightarrow 010 \text{ ev ev } J \\ &\rightarrow 010 \text{ ev ev } 01 \text{ ev } J \\ &\rightarrow 010 \text{ ev } 0 \text{ od } 1 \text{ ev } J \\ &\rightarrow 010 \text{ ev } 0 \text{ ev ev } J \\ &\rightarrow 0100 \text{ od ev ev } J \end{aligned}$$
$$\begin{aligned} J &\twoheadrightarrow 0100 \text{ od ev ev } J \\ \text{od ev ev } J &\twoheadrightarrow_4 \\ \text{ev od ev ev } J & \\ &\twoheadrightarrow^\omega \text{ ev}^\omega \end{aligned}$$

INFINITARY VIEW ON JOERG'S EXAMPLE

$J \rightarrow 01 \text{ ev } J$ $\text{ev } x \rightarrow x \text{ od}$ $\text{od } x \rightarrow \text{ev}$

unwinding

$$\begin{aligned}
 J &\xrightarrow{\omega} (01 \text{ ev})^{\omega} \equiv 01 (\text{ev } 01)^{\omega} \\
 &\xrightarrow{\omega} 01 (0 \text{ od } 1)^{\omega} \\
 &\xrightarrow{\omega} (01 (0 \text{ ev})^{\omega}) \equiv 010 (\text{ev } 0)^{\omega} \\
 &\xrightarrow{\omega} 010 (0 \text{ od})^{\omega} \equiv (0100 (\text{od } 0)^{\omega}) \\
 &\xrightarrow{\omega} 0100 \text{ ev}^{\omega}
 \end{aligned}$$

$J \xrightarrow{\omega.5} 0100 \text{ ev}^{\omega}$

viewed as a process the system deadlocks after 4 proper steps, then is infinitely idling

$J \rightarrow abeJ$

$ea \rightarrow ao$

$eb \rightarrow bo$

$oa \rightarrow e$

$ob \rightarrow e$

abeabeabeabeabeabeabeabeabeabeabeabeabeabeabeabeabeabe

abaobaobaobaobaobaobaobaobaobaobaobaobaobaobaobaobaob

abae

abao

abae

$J \rightarrow abeJ$

$ea \rightarrow ao$

$eb \rightarrow bo$

$oa \rightarrow e$

$ob \rightarrow e$

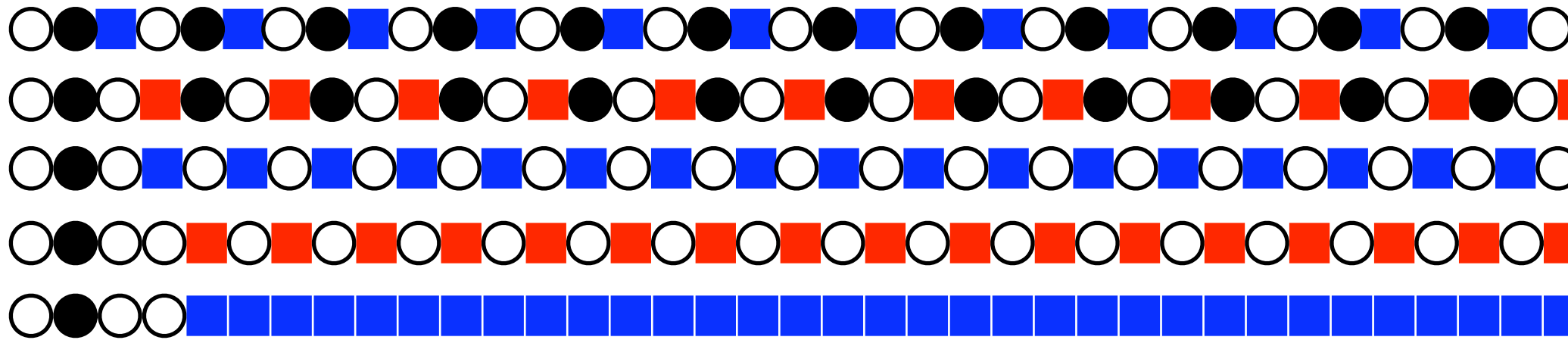
$J \rightarrow \text{O} \bullet \blacksquare J$

$\blacksquare \text{O} \rightarrow \text{O} \blacksquare$

$\blacksquare \bullet \rightarrow \bullet \blacksquare$

$\blacksquare \text{O} \rightarrow \blacksquare$

$\blacksquare \bullet \rightarrow \blacksquare$



guiding example 1.

$X = 0^n: \text{tail}^m X$

for what n, m is this a good definition of a stream X ?

Try $X = 0: \text{tail} X$

guiding example 2.

Define a function f
from streams to
streams by

$$f(x:y:s) = x: f(f(s))$$

try on the stream $s = 00s$:
 $fs = f00s = 0ffs = 0ff00s =$
 $0f0ffs = 0f0ff00s = 0f0f0ffs$
 $= 0f0f0f0f0f0f0f0f0f...$

Joerg's example

$$f_{n,m}(x_1 : x_2 : \dots : x_n : s) = 0^m : f(s)$$

$$M_k = 0^k : f(M_k)$$

For what n, m, k does PR hold?

$f_{2,1}$ and M_3

$000f000f000f000f000f000f000f...$
 $0000f 00f 00f 00f 00f 00f 00f 0...$
 $00000f 0f 0f 0f 0f 0f 0f...$

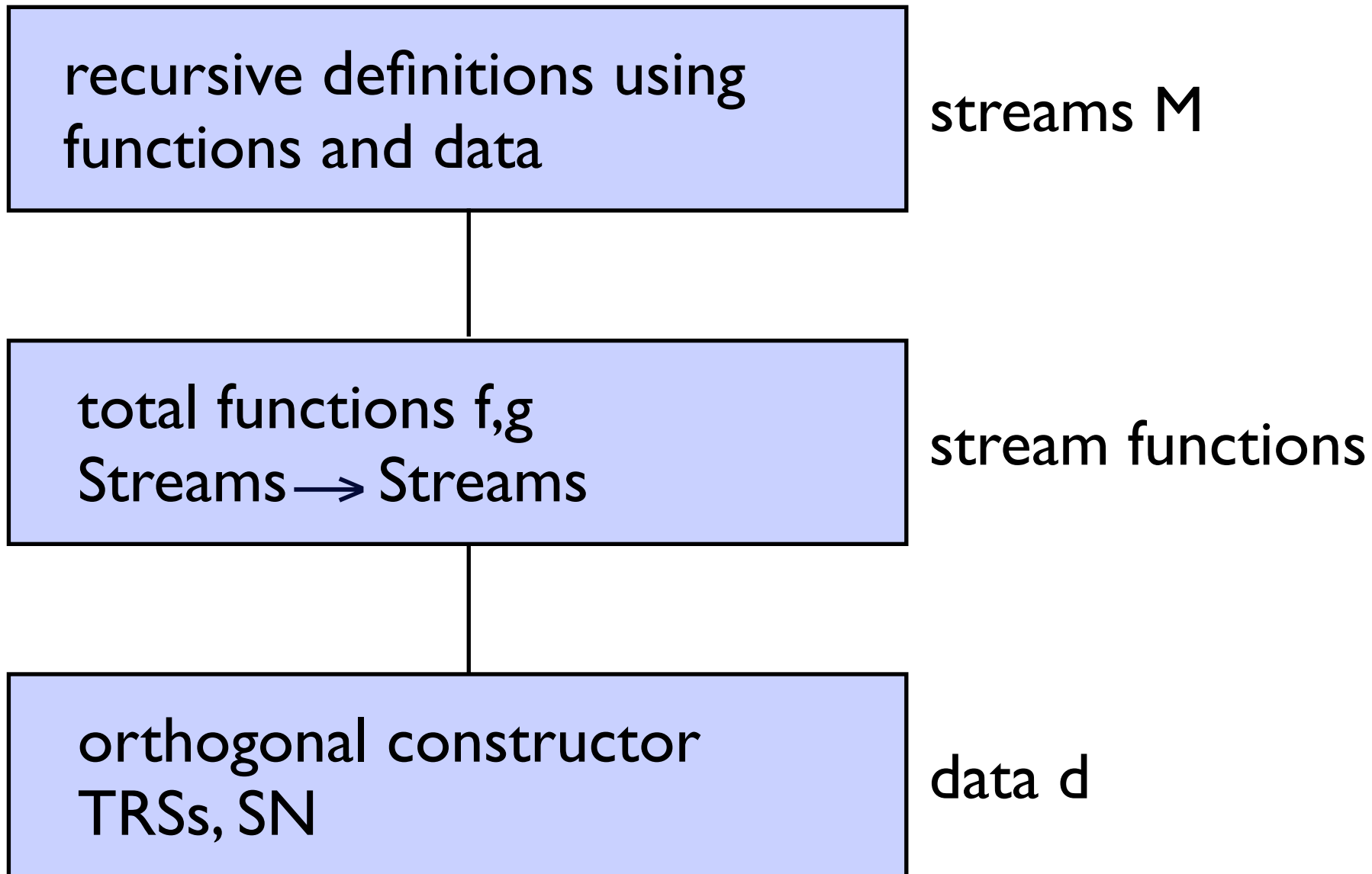
note the transfinite reductions

on zeros stream
 $f00 \rightarrow 0f$

SN^∞
 not PR

$k \geq n \ \& \ m \geq 1$

The format of stream definitions



Running example: Morse sequence

$$M \rightarrow 0 : \text{zip}(\text{inv}(\text{even}(M)), \text{tail}(M))$$

$$\text{zip}(x : \sigma, \tau) \rightarrow x : \text{zip}(\tau, \sigma)$$
$$\text{inv}(x : \sigma) \rightarrow i(x) : \text{inv}(\sigma)$$
$$\text{even}(x : \sigma) \rightarrow x : \text{odd}(\sigma)$$
$$\text{odd}(x : \sigma) \rightarrow \text{even}(\sigma)$$
$$\text{tail}(x : \sigma) \rightarrow \sigma$$

$$i(0) \rightarrow 1 \quad i(1) \rightarrow 0$$

stream

functions

data

productive

Format of total functions f, g
Streams \rightarrow Streams

stream functions

Each f has *one* rule, of one of the following two forms:

no pattern matching

$f(\vec{x}_1:\sigma_1, \dots, \vec{x}_n:\sigma_n, \vec{y}) \rightarrow \vec{t}:\sigma_i$ *collaps form*

$f(\vec{x}_1:\sigma_1, \dots, \vec{x}_n:\sigma_n, \vec{y}) \rightarrow \vec{t}:g(\vec{\sigma}, \vec{t}')$ *transfer form*

$\text{even}(x:\sigma) \rightarrow x:\text{odd}(\sigma)$

$\text{odd}(y:\sigma) \rightarrow \text{even}(\sigma)$

dependency cycle must pass at least one guard

Non-example 1

$\text{head}(x:\sigma) \rightarrow x$

is not an allowed stream function, since it leads to partiality:

refers to itself

$g(\sigma) \rightarrow 0: \text{head}(\text{tail}(g(\sigma))) : g(\sigma)$

applied on zeros $\rightarrow 0: \text{zeros}$ yields

$g(\text{zeros}) = 0: ? : 0: ? : 0: ? : 0: ? : 0: ? \dots$

so g is not a total function

Non-example 2: no nested functions

Define a function f from streams to streams by

$$f(x:y:\sigma) = x:f(f(\sigma))$$

applied on the stream $Z = 0:0:Z$:

$$f(Z) =$$

$$f(0:0:Z) =$$

$$0:f(f(Z)) =$$

$$0:f(f(0:0:Z)) =$$

$$0:f(0:f(f(Z))) =$$

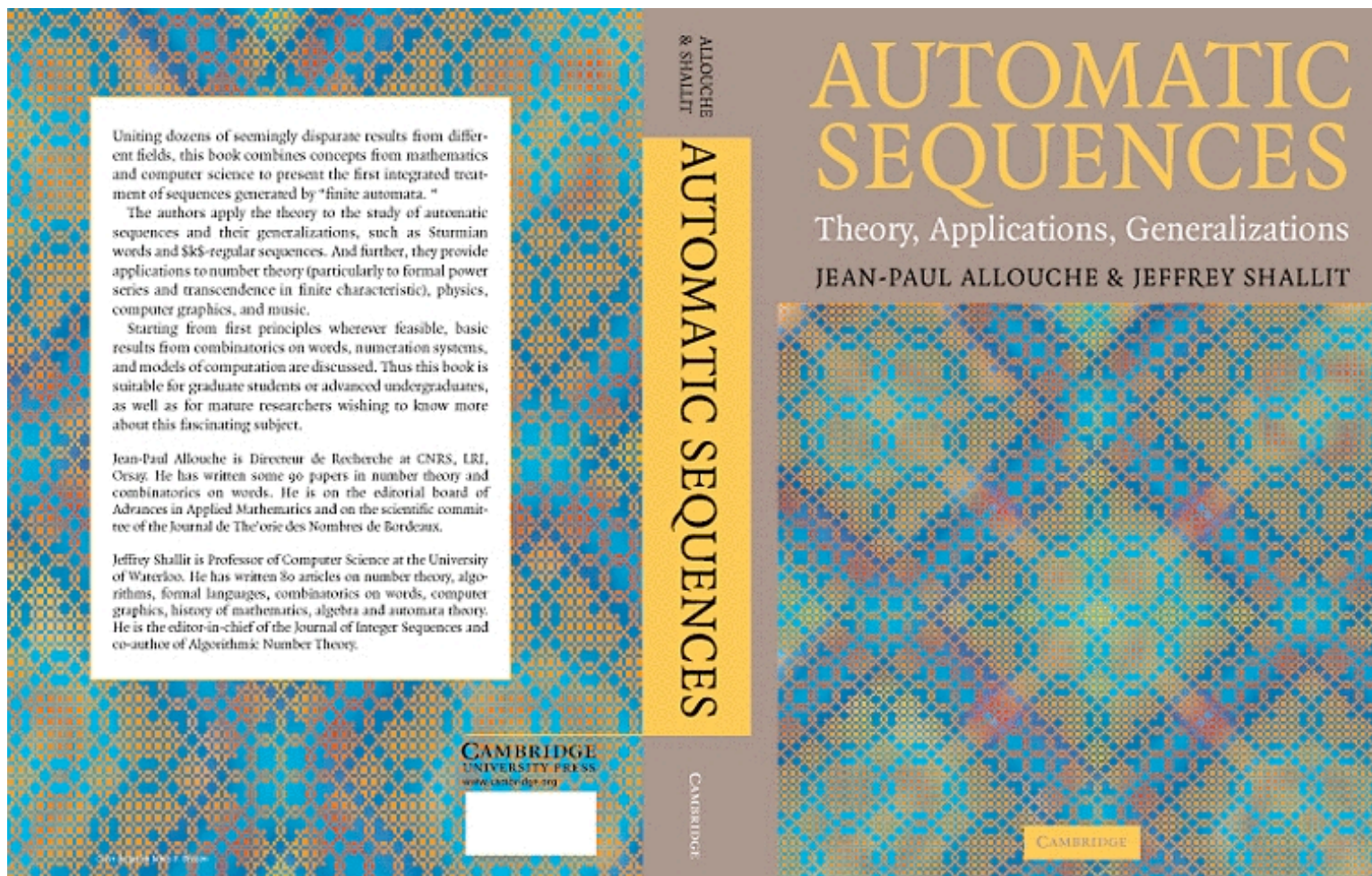
$$0:f(0:f(f(0:0:Z))) =$$

$$0:f(0:f(0:f(f(Z)))) =$$

$$0:f(0:f(0:f(0:f(0:f(0:f(0:f\dots$$

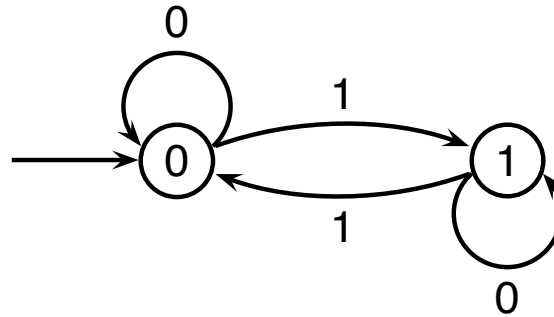
normal form, but not constructor normal form

This format defines all streams obtainable as D0L sequences, and also all automatic sequences (Allouche-Shallit)



Automatic Sequences

A DFAO A is a det., finite automaton with output $\in \Sigma$ in each state:



The output of $A(w)$ is the output of the final state when A reads w .

Definition

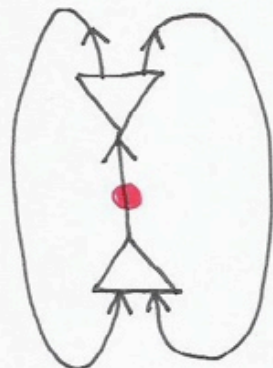
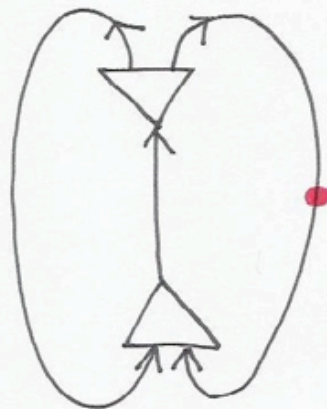
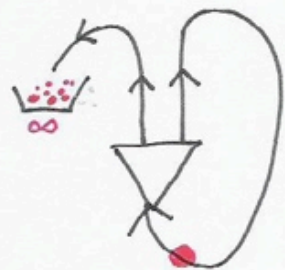
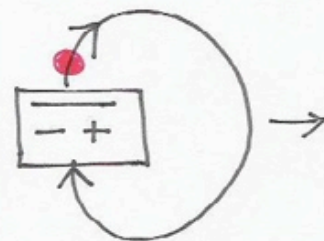
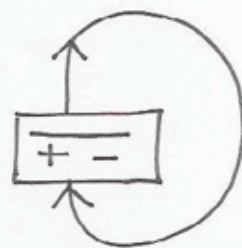
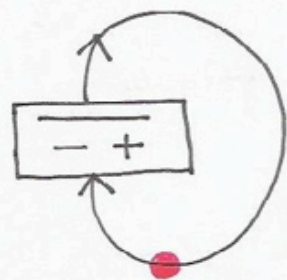
A sequence M is k -automatic if there exists DFAO A :

$$M(i) = \text{output of } A \text{ reading the representation of } i \text{ to the base } k$$

The above automaton defines Thue-Morse 0110...:

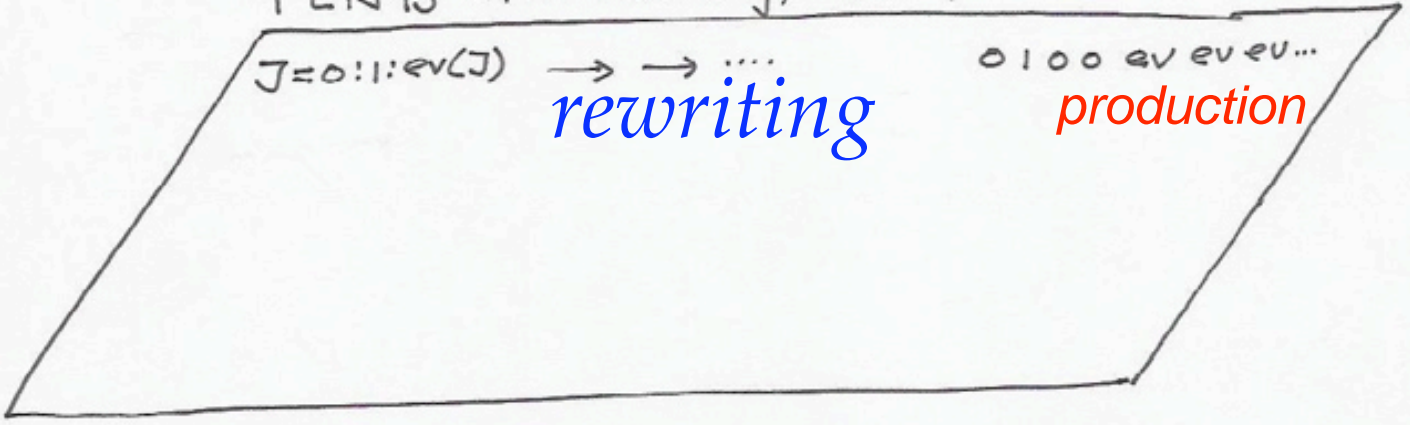
$$A(0) = 0, A(1) = 1, A(10) = 1, A(11) = 0, \dots$$

PEBBLEFLOW

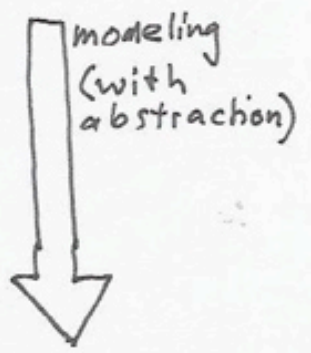


OVERVIEW

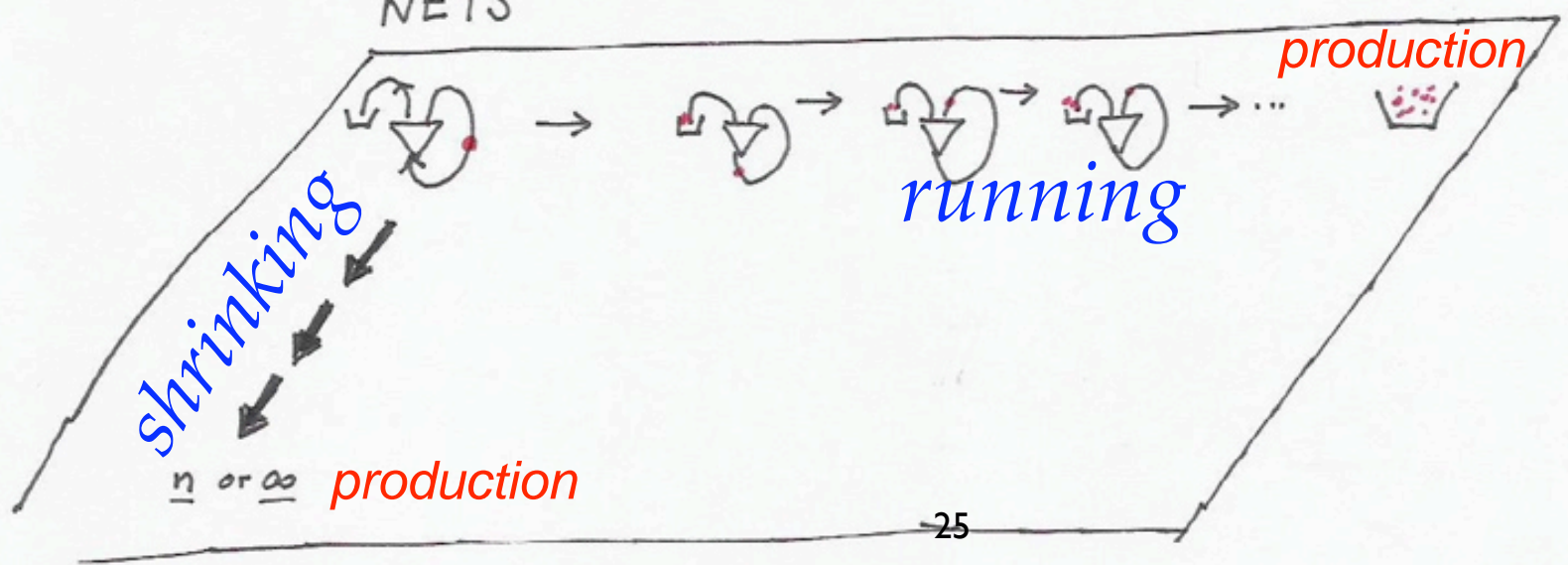
TERMS with rewriting, in finite



modeling



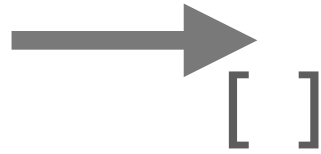
NETS



Terms
 $S = 0: \text{tail}(S)$

orthogonal TRS

abstraction
translation



productivity
preserving

*rational pebble
net with boxes*
 $\mu S. \bullet(\text{--}+(S))$

orthogonal CRS

*box composition and fixed point
defined coinductively or with
infinitary rewriting*

productivity
preserving

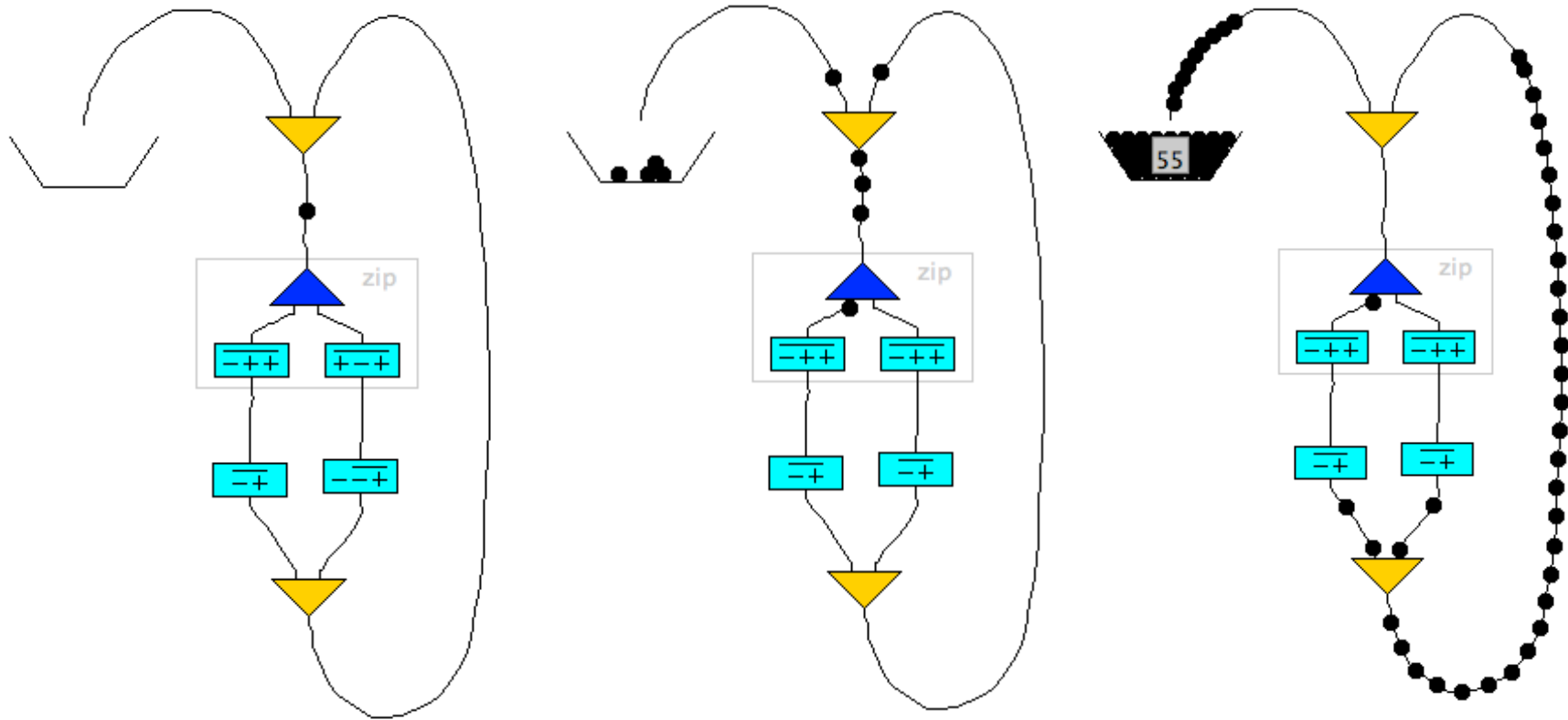
simplification to
source nets
 k or ∞

not orthogonal TRS,
but SN and CR

*lazy natural number
nets* $\mathbb{N}^\infty = \{ \underline{0}, \underline{1}, \underline{2}, \dots, \underline{\infty} \}$

Productivity is decidable for pure stream specifications

pebbleflow of morse stream



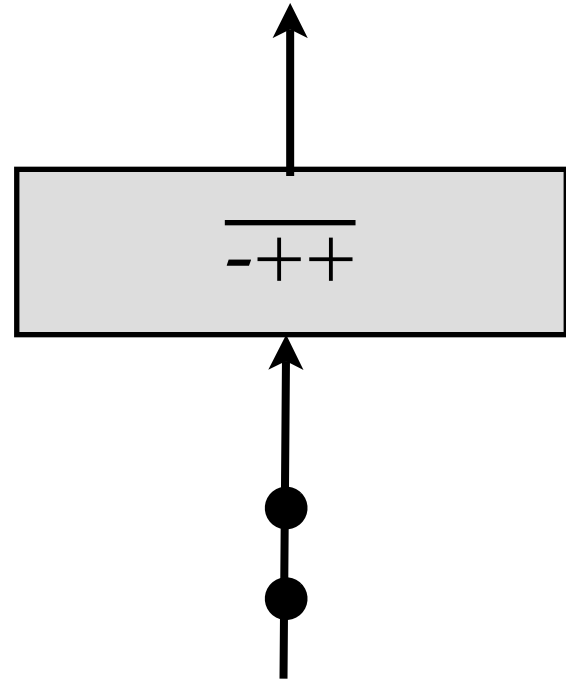
$$M = 0:\text{zip}(\text{inv}(M), \text{tail}(M))$$

Boxes: functions as I/O-sequences

$$\text{dup}(x:\sigma) \rightarrow x:x:\text{dup}(\sigma)$$

$$-++-++-++-++-++\dots = \overline{-++}$$

rational I/O-sequence



even(a:s) = a:odd(s)
odd(a:s) = even(s)
coarse_even(a:b:s) = a:coarse_even(s)

zip(a:s,t) = a:zip(t,s)
coarse_zip(a:s,b:t) = a:b:coarse_zip(s,t)

tail(a:s) = s

A = 0:zip(even(A),odd(A))

B1 = 0:zip(B1,even(tail(B1)))

B2 = 0:coarse_zip(B2,even(tail(B2)))

B3 = 0:zip(B3,coarse_even(tail(B3)))

We translate \mathbf{A} into a pebbleflow net and collapse it to a source:

$$\begin{aligned}
[\mathbf{A}] &= \mu A. \bullet (\Delta(\overline{-++}(\overline{-+-}(A)), \overline{+-+}(\overline{--+(A)}))) \\
&\rightarrow_c \mu A. \bullet (\Delta(\overline{-++-}(A), \overline{+---}(A))) \\
&\rightarrow_c \mu A. \overline{+-+}(\Delta(\overline{-++-}(A), \overline{+---}(A))) \\
&\rightarrow_c \mu A. \Delta(\overline{+-+}(\overline{-++-}(A)), \overline{+-+}(\overline{+---}(A))) \\
&\rightarrow_c \mu A. \Delta(\overline{+-++-}(A), \overline{++--}(A)) \\
&\rightarrow_c \Delta(\mu A. \overline{+-++-}(A), \mu A. \overline{++--}(A)) \\
&\rightarrow_c \Delta(\infty, \infty) \\
&\rightarrow_c \infty
\end{aligned}$$

Hence the definition of \mathbf{A} is productive.

We translate **B1** into a pebbleflow net and collapse it to a source:

$$\begin{aligned}
[\mathbf{B1}] &= \mu B1.\bullet(\Delta(\overline{-++}(B1), \overline{+-+}(\overline{-+-}(\overline{-+}(B1)))))) \\
&\rightarrow_c \mu B1.\bullet(\Delta(\overline{-++}(B1), \overline{+-++-}(\overline{-+}(B1)))) \\
&\rightarrow_c \mu B1.\bullet(\Delta(\overline{-++}(B1), \overline{+--+(B1)})) \\
&\rightarrow_c \mu B1.\overline{+-+}(\Delta(\overline{-++}(B1), \overline{+--+(B1)})) \\
&\rightarrow_c \mu B1.\Delta(\overline{+-+}(\overline{-++}(B1)), \overline{+-+}(\overline{+--+(B1)})) \\
&\rightarrow_c \mu B1.\Delta(\overline{+-+}(B1), \overline{++--}(B1)) \\
&\rightarrow_c \Delta(\mu B1.\overline{+-+}(B1), \mu B1.\overline{++--}(B1)) \\
&\rightarrow_c \Delta(\infty, \infty) \\
&\rightarrow_c \infty
\end{aligned}$$

We translate **B2** into a pebbleflow net and collapse it to a source:

$$\begin{aligned}
[\mathbf{B2}] &= \mu B2.\bullet(\Delta(\overline{-++}(B2), \overline{-++}(\overline{-+-}(\overline{-++}(B2)))))) \\
&\rightarrow_c \mu B2.\bullet(\Delta(\overline{-++}(B2), \overline{-++-}(\overline{-++}(B2)))) \\
&\rightarrow_c \mu B2.\bullet(\Delta(\overline{-++}(B2), \overline{-+--}(B2))) \\
&\rightarrow_c \mu B2.\overline{+-+}(\Delta(\overline{-++}(B2), \overline{-+--}(B2))) \\
&\rightarrow_c \mu B2.\Delta(\overline{+-+}(\overline{-++}(B2)), \overline{+-+}(\overline{-+--}(B2))) \\
&\rightarrow_c \mu B2.\Delta(\overline{+-+}(B2), \overline{+-+}(B2)) \\
&\rightarrow_c \Delta(\mu B2.\overline{+-+}(B2), \mu B2.\overline{+-+}(B2)) \\
&\rightarrow_c \Delta(\infty, 1) \\
&\rightarrow_c 1
\end{aligned}$$

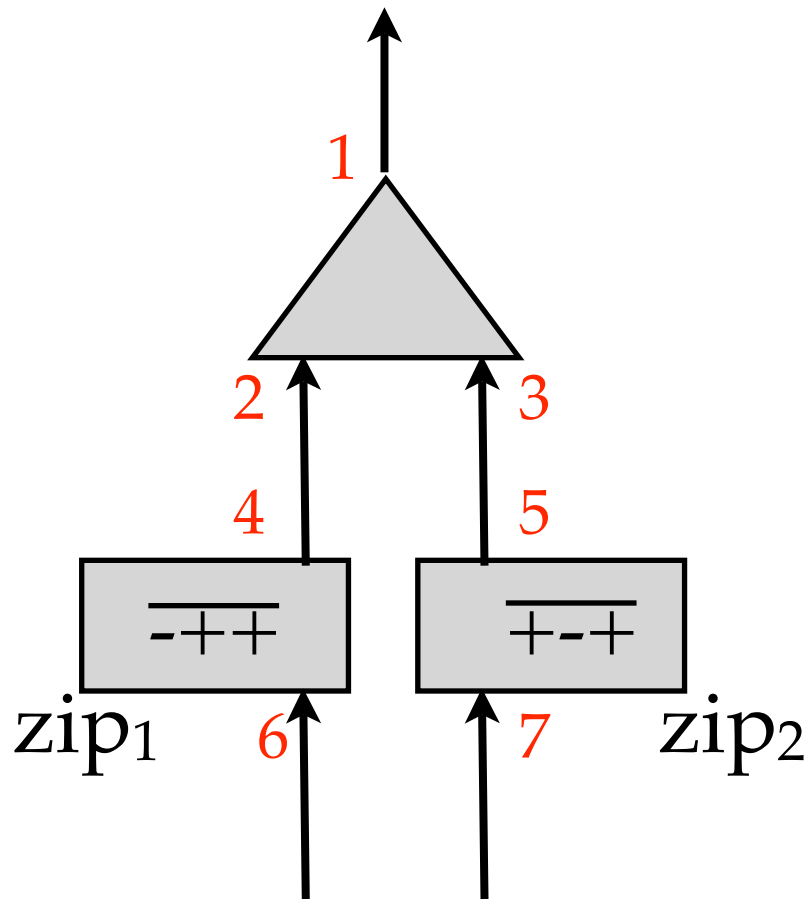
Hence the definition of **B2** is not productive (produces 1 element).

We translate **B3** into a pebbleflow net and collapse it to a source:

$$\begin{aligned}
[\mathbf{B3}] &= \mu B3.\bullet(\Delta(\overline{-++}(B3), \overline{+-+}(\overline{---+}(\overline{---+}(B3)))))) \\
&\rightarrow_c \mu B3.\bullet(\Delta(\overline{-++}(B3), \overline{+---+}(\overline{---+}(B3)))) \\
&\rightarrow_c \mu B3.\bullet(\Delta(\overline{-++}(B3), \overline{+----++}(B3))) \\
&\rightarrow_c \mu B3.\overline{+-+}(\Delta(\overline{-++}(B3), \overline{+----++}(B3))) \\
&\rightarrow_c \mu B3.\Delta(\overline{+-+}(\overline{-++}(B3)), \overline{+-+}(\overline{+----++}(B3))) \\
&\rightarrow_c \mu B3.\Delta(\overline{+-+}(B3), \overline{++----++}(B3)) \\
&\rightarrow_c \Delta(\mu B3.\overline{+-+}(B3), \mu B3.\overline{++----++}(B3)) \\
&\rightarrow_c \Delta(\infty, 2) \\
&\rightarrow_c 2
\end{aligned}$$

Hence the definition of **B3** is not productive (produces 2 elements).

how to treat binary functions like zip?



$$[\text{zip}] = [\text{zip}_1] \Delta [\text{zip}_2] =$$

$$-+-+ \Delta +-+$$

	[zip ₁]	Δ	[zip ₂]
-	r6		
			s5 +
+	s4		
		r23, s1	
			r7 -
+	s4		
			s5 +
		r23, s1	

timeline of one zip cycle

The translation of zip is computed as follows:

$$\begin{aligned}
 [\text{zip}] &= \Delta([\text{zip}]_{1,0}[\text{zip}]_{2,0}) \\
 [\text{zip}]_{1,0} &= \mu(\text{zip}_{1,0}) \cdot \wedge \left\{ -+ \wedge \left\{ \mu(\text{zip}_{2,0}) \cdot \wedge \left\{ + \wedge \left\{ \text{zip}_{1,0} \right. \right. \right. \right. \\
 &= \overline{-++} \\
 [\text{zip}]_{2,0} &= \mu(\text{zip}_{2,0}) \cdot \wedge \left\{ + \wedge \left\{ \mu(\text{zip}_{1,0}) \cdot \wedge \left\{ -+ \wedge \left\{ \text{zip}_{2,0} \right. \right. \right. \right. \\
 &= \overline{+-+}
 \end{aligned}$$

The translation of inv is computed as follows:

$$\begin{aligned}
 [\text{inv}] &= \Delta([\text{inv}]_{1,0}) \\
 [\text{inv}]_{1,0} &= \mu(\text{inv}_{1,0}) \cdot \wedge \left\{ \begin{array}{l} -+ \wedge \left\{ \text{inv}_{1,0} \right. \\ -+ \wedge \left\{ \text{inv}_{1,0} \right. \end{array} \right. \\
 &= \overline{-+}
 \end{aligned}$$

Pebble flow reduction rules

$$P_1 \quad \bullet(N_1) \Delta \bullet(N_2) \rightarrow \bullet(N_1 \Delta N_2)$$

$$P_2 \quad \mu x. \bullet(N(x)) \rightarrow \bullet(\mu x. N(\bullet(x)))$$

$$P_3 \quad +\sigma(N) \rightarrow \bullet(\sigma(N))$$

$$P_4 \quad -\sigma(\bullet(N)) \rightarrow \sigma(N)$$

$$P_5 \quad s(\underline{k}) \rightarrow \bullet(\underline{k})$$

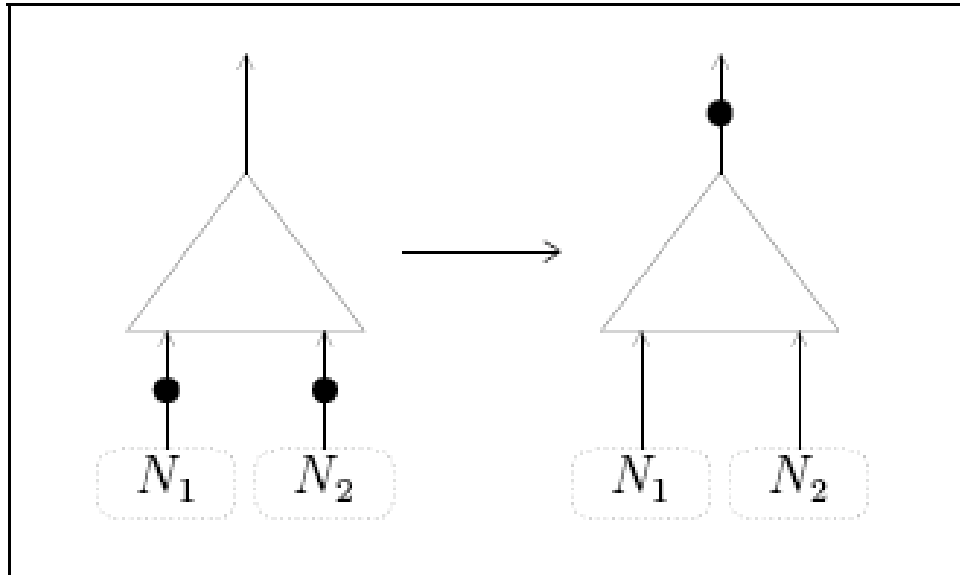
micro- μ -rule

$$\begin{aligned} \mu x. f(g(x)) &\rightarrow f(\mu x. g(f(x))) \\ &\rightarrow f(g(\mu x. f(g(x)))) \end{aligned}$$

orthogonal CRS, hence CR

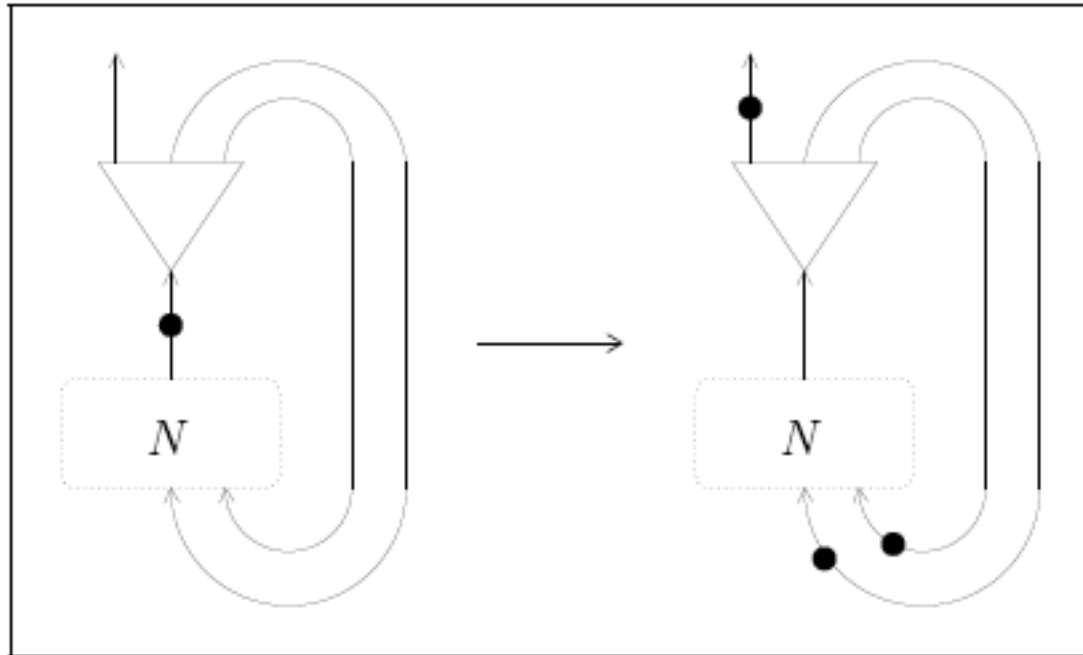
P_1

$$\bullet(N1) \Delta \bullet(N2) \rightarrow \bullet(N1 \Delta N2)$$



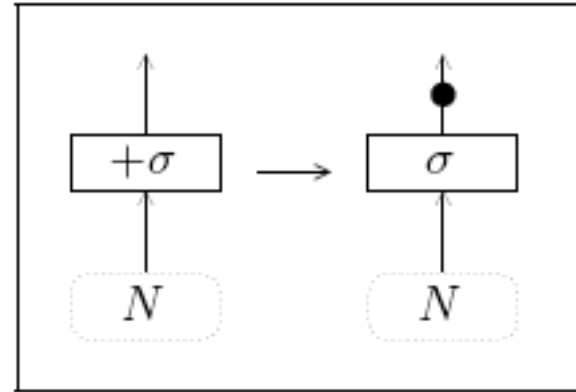
P_2

$$\mu x. \bullet(N(x)) \rightarrow \bullet(\mu x. N(\bullet(x)))$$



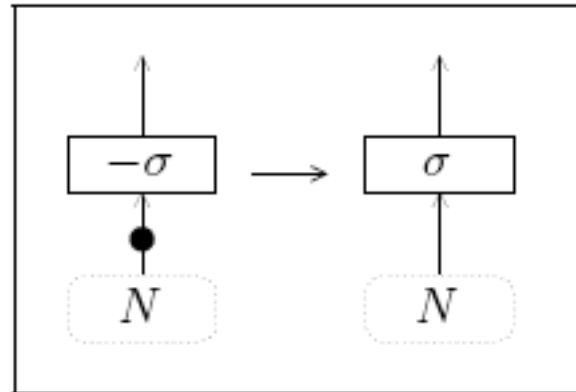
P_3

$$+\sigma(N) \rightarrow \bullet(\sigma(N))$$



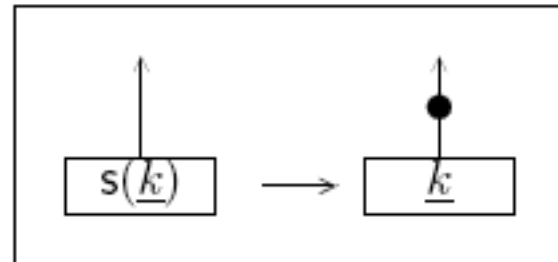
P_4

$$-\sigma(\bullet(N)) \rightarrow \sigma(N)$$



P_5

$$s(\underline{k}) \rightarrow \bullet(\underline{k})$$



Productivity of Stream Definitions

Productivity

Productivity Tool

Examples

Our Examples

Telford and Turner

Buchholz

Pebbleflow Applet

Contact

Our Examples

- Thue-Morse Sequence
- Alternative Definition of the Thue-Morse Sequence
- (Non-pure) Definition of the Ternary Thue-Morse Sequence
- (Pure) Definition of the Ternary Thue-Morse Sequence
- Hamming Numbers
- Fibonacci Numbers

Thue-Morse Sequence

The (strongly) cube-free Thue-Morse sequence M :

```
Signature(  
  M : stream(bit),  
  0,1 : bit,  
  tail : stream(x) -> stream(x),  
  zip : stream(x) -> stream(x) -> stream(x),  
  inv : stream(bit) -> stream(bit)  
)  
  
M = 0:zip(inv(M),tail(M))  
  
zip(a:s,t) = a:zip(t,s)  
tail(a:s) = s  
inv(0:s) = 1:inv(s)  
inv(1:s) = 0:inv(s)
```

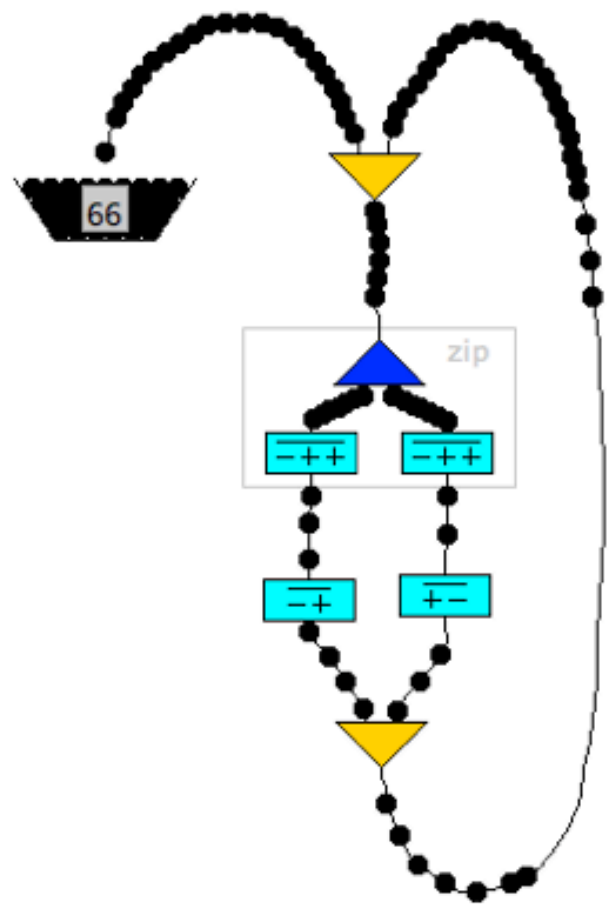
--- GO ---

Alternative Definition of the Thue-Morse Sequence

An alternative definition of the Thue-Morse sequence M (derived from the D0L sequence definition):

```
Signature(  
  M : stream(bit),  
  0,1 : bit,  
  tail : stream(x) -> stream(x),
```


[How to use\(another window\)](#)



Thue-Morse Sequence

The (strongly) cube-free Thue-Morse sequence M :

```
Signature(  
  M : stream(bit),  
  0,1 : bit,  
  tail : stream(x) -> stream(x),  
  zip : stream(x) -> stream(x) -> stream(x),  
  inv : stream(bit) -> stream(bit)  
)
```

```
M = 0:zip(inv(M),tail(M))
```

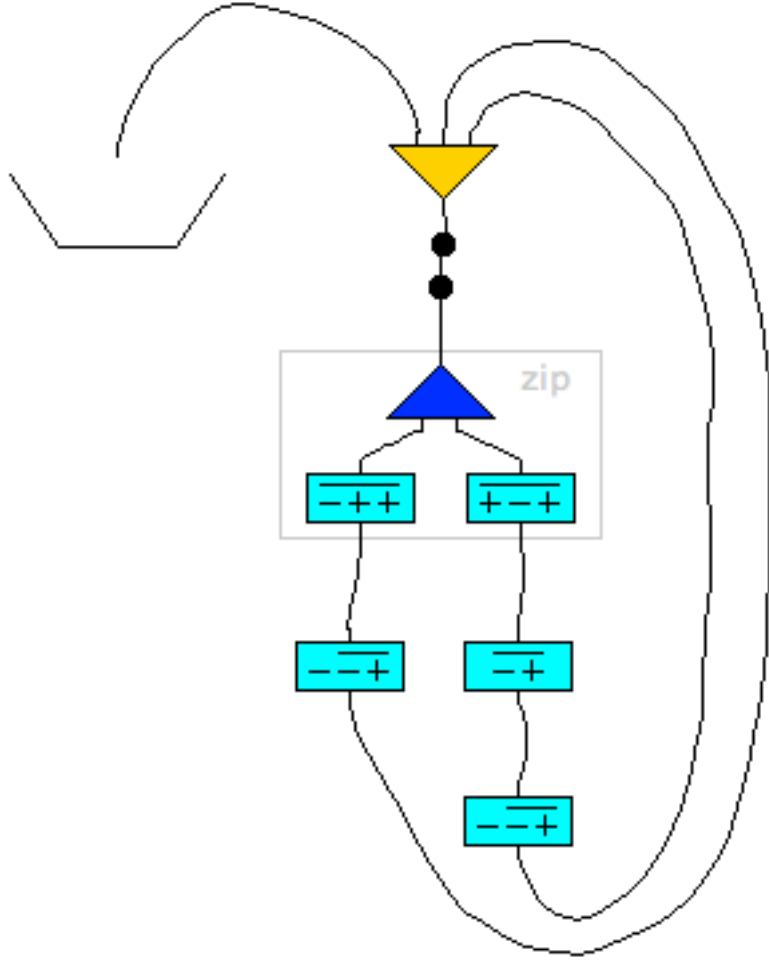
```
zip(a:s,t) = a:zip(t,s)
```

```
tail(a:s) = s
```

```
inv(0:s) = 1:inv(s)
```

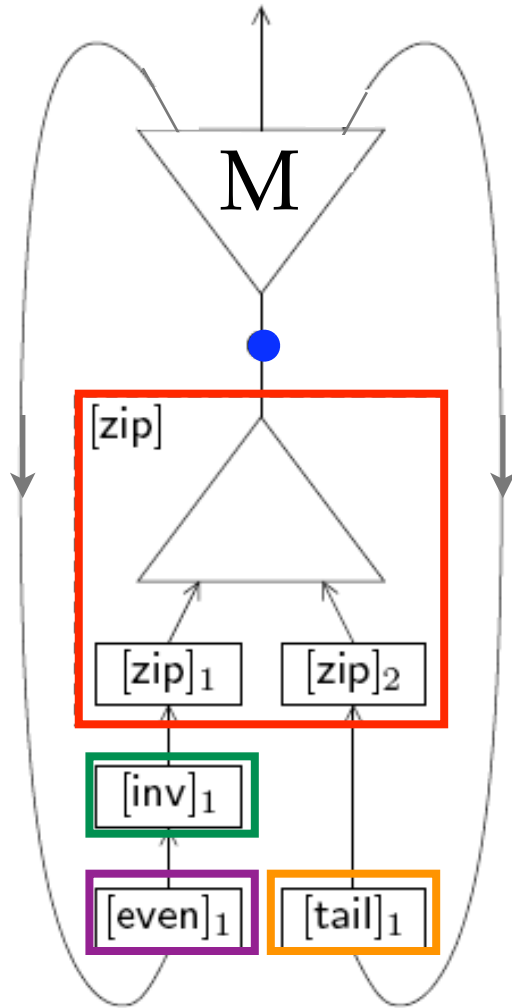
```
inv(1:s) = 0:inv(s)
```

--- GO ---



<http://fspc282.few.vu.nl/productivity/>

Morse stream M as pebble net and pebble net term



$$\begin{aligned}
 [\text{zip}] &= \text{gate}([\text{zip}]_1, [\text{zip}]_2) \\
 [\text{zip}]_1 &= -+[\text{zip}]_2 = -++[\text{zip}]_1 = \overline{-++} \\
 [\text{zip}]_2 &= +[\text{zip}]_1 = +-+[\text{zip}]_2 = \overline{+-+} \\
 [\text{inv}] &= \text{gate}([\text{inv}]_1) \\
 [\text{inv}]_1 &= -+[\text{inv}]_1 = \overline{-+} \\
 [\text{even}] &= \text{gate}([\text{even}]_1) \\
 [\text{even}]_1 &= -+[\text{odd}]_1 = -+-[\text{even}]_1 = \overline{-+-} \\
 [\text{odd}]_1 &= -[\text{even}]_1 = ---+[\text{odd}]_1 = \overline{---+} \\
 [\text{tail}] &= \text{gate}([\text{tail}]_1) \\
 [\text{tail}]_1 &= \overline{---+}
 \end{aligned}$$

$$[M] = \mu M. \bullet (\Delta (\text{box}(\overline{-++}, \text{box}(\overline{-+}, \text{box}(\overline{-+-}, M))), \text{box}(\overline{+-+}, \text{box}(\overline{- - +}, M))))$$

$$[M] = \mu M. \bullet (\overline{-++}(\overline{-+}(\overline{-+-}(M)))) \Delta \overline{+-+}(\overline{- - +}(M))$$

$$M = 0:\text{zip}(\text{inv}(\text{even}(M)), \text{tail}(M))$$

Simplification of net for Morse stream definition, proving productivity

$$[M] = \mu M. \bullet(\overline{-++}(\overline{-+}(\overline{-+-}(M)))) \Delta \overline{+--+}(\overline{--+}(M))$$

$$M = 0:\text{zip}(\text{inv}(\text{even}(M)), \text{tail}(M))$$

$$\begin{aligned} [M] &= \mu M. \bullet(\Delta(\text{box}(\overline{-++}, \text{box}(\overline{-+}, \text{box}(\overline{-+-}, M))), \text{box}(\overline{+--+}, \text{box}(\overline{--+}, M)))) \\ &\rightarrow_{R_2}^3 \mu M. \bullet(\Delta(\text{box}(\overline{-++-}, M), \text{box}(\overline{+-++}, M))) \\ &\rightarrow_{R_1 \cdot R_3} \mu M. \Delta(\text{box}(\overline{+--+}, \text{box}(\overline{-++-}, M)), \text{box}(\overline{+--+}, \text{box}(\overline{+-++}, M))) \\ &\rightarrow_{R_2}^2 \mu M. \Delta(\text{box}(\overline{+--+}, M), \text{box}(\overline{++-++}, M)) \\ &\rightarrow_{R_4} \Delta(\mu M. \text{box}(\overline{+--+}, M), \mu M. \text{box}(\overline{++-++}, M)) \\ &\rightarrow_{R_6} \Delta(\text{src}(\text{fix}(\overline{+--+}), \text{src}(\text{fix}(\overline{++-++}))) = \Delta(\text{src}(\infty), \text{src}(\infty)) \\ &\rightarrow_{R_7} \text{src}(\infty), \end{aligned}$$

productive!

N.B.: Intensional aspect: replacing a function in a context by an extensionnally equivalent, but ‘coarser’ definition, may introduce unproductivity

Note that the ‘fine’ definitions of `zip` and `even` are crucial in this setting. If we replace the definition of `zip` in the SCS for `M` by the ‘coarser’ one: $\text{zip}^*(x : \sigma, y : \tau) \rightarrow x : y : \text{zip}^*(\sigma, \tau)$ we obtain an SCS \mathcal{T}^* where:

$$\begin{aligned}
[M] &= \mu M. \bullet (\Delta(\text{box}(\overline{-++}, \text{box}(\overline{-+}, \text{box}(\overline{-+-}, M))), \text{box}(\overline{-++}, \text{box}(\overline{-+}, M)))) \\
&\rightarrow_{R_2}^3 \mu M. \bullet (\Delta(\text{box}(\overline{-++-}, M), \text{box}(\overline{-+}, M))) \\
&\rightarrow_{R_1 \cdot R_3} \mu M. \Delta(\text{box}(\overline{+-+}, \text{box}(\overline{-++-}, M)), \text{box}(\overline{+-+}, \text{box}(\overline{-+}, M))) \\
&\rightarrow_{R_2}^2 \mu M. \Delta(\text{box}(\overline{+-++-}, M), \text{box}(\overline{+-+}, M)) \\
&\rightarrow_{R_4 \cdot R_6} \Delta(\text{src}(\text{fix}(\overline{+-++-})), \text{src}(\text{fix}(\overline{+-+}))) = \Delta(\text{src}(\infty), \text{src}(\underline{1})) \rightarrow_{R_7} \text{src}(\underline{1})
\end{aligned}$$

Hence `M` is not productive in \mathcal{T}^* (here it produces only one element).

The automated productivity prover has been applied to:

```
-- stream layer --  
M = 0:1:even(M)  
  
-- function layer --  
even(a:s) = a:odd(s)  
odd(a:s) = even(s)  
  
-- data layer --
```

$M = 0:1:\text{even}(M)$

This is a pure stream specification for which we can decide productivity.

The translation of `even` is computed as follows:

$$\begin{aligned} [\text{even}] &= \Delta([\text{even}]_{1,0}) \\ [\text{even}]_{1,0} &= \mu(\text{even}_{1,0}). \wedge \left\{ -+ \wedge \left\{ \mu(\text{odd}_{1,0}). \wedge \left\{ - \wedge \left\{ \text{even}_{1,0} \right. \right. \right. \right. \\ &= \overline{-+-} \end{aligned}$$

The translation of `odd` is computed as follows:

$$\begin{aligned} [\text{odd}] &= \Delta([\text{odd}]_{1,0}) \\ [\text{odd}]_{1,0} &= \mu(\text{odd}_{1,0}). \wedge \left\{ - \wedge \left\{ \mu(\text{even}_{1,0}). \wedge \left\{ -+ \wedge \left\{ \text{odd}_{1,0} \right. \right. \right. \right. \\ &= \overline{---+} \end{aligned}$$

We translate M into a pebbleflow net and collapse it to a source:

$$\begin{aligned}
 [M] &= \mu M. \bullet(\bullet(\overline{-+-}(M))) \\
 &\rightarrow_c \mu M. +\overline{-+}(\bullet(\overline{-+-}(M))) \\
 &\rightarrow_c \mu M. +\overline{-+}(+\overline{-+}(\overline{-+-}(M))) \\
 &\rightarrow_c \mu M. +\overline{+-}(\overline{-+-}(M)) \\
 &\rightarrow_c \mu M. ++\overline{-+-}(M) \\
 &\rightarrow_c 4
 \end{aligned}$$

The specification of M is not productive (produces 4 elements).

Simplification of net for Morse stream definition, proving productivity

$$[M] = \mu M. \bullet(\overline{-++}(\overline{-+}(\overline{-+-}(M)))) \Delta \overline{+--+}(\overline{--+}(M))$$

$$M = 0:\text{zip}(\text{inv}(\text{even}(M)), \text{tail}(M))$$

$$\begin{aligned} [M] &= \mu M. \bullet(\Delta(\text{box}(\overline{-++}, \text{box}(\overline{-+}, \text{box}(\overline{-+-}, M))), \text{box}(\overline{+--+}, \text{box}(\overline{--+}, M)))) \\ &\rightarrow_{R_2}^3 \mu M. \bullet(\Delta(\text{box}(\overline{-++-}, M), \text{box}(\overline{+-+--+}, M))) \\ &\rightarrow_{R_1 \cdot R_3} \mu M. \Delta(\text{box}(\overline{+-+}, \text{box}(\overline{-++-}, M)), \text{box}(\overline{+-+}, \text{box}(\overline{+-+--+}, M))) \\ &\rightarrow_{R_2}^2 \mu M. \Delta(\text{box}(\overline{+-+ +-}, M), \text{box}(\overline{++-+--+}, M)) \\ &\rightarrow_{R_4} \Delta(\mu M. \text{box}(\overline{+-+ +-}, M), \mu M. \text{box}(\overline{++-+--+}, M)) \\ &\rightarrow_{R_6} \Delta(\text{src}(\text{fix}(\overline{+-+ +-})), \text{src}(\text{fix}(\overline{++-+--+}))) = \Delta(\text{src}(\infty), \text{src}(\infty)) \\ &\rightarrow_{R_7} \text{src}(\infty), \end{aligned}$$

productive!

N.B.: Intensional aspect: replacing a function in a context by an extensionnally equivalent, but ‘coarser’ definition, may introduce unproductivity

Note that the ‘fine’ definitions of `zip` and `even` are crucial in this setting. If we replace the definition of `zip` in the SCS for `M` by the ‘coarser’ one: $\text{zip}^*(x : \sigma, y : \tau) \rightarrow x : y : \text{zip}^*(\sigma, \tau)$ we obtain an SCS \mathcal{T}^* where:

$$\begin{aligned}
 [M] &= \mu M. \bullet (\Delta(\text{box}(\overline{-++}, \text{box}(\overline{-+}, \text{box}(\overline{-+-}, M))), \text{box}(\overline{-++}, \text{box}(\overline{-+}, M)))) \\
 &\rightarrow_{\text{R2}}^3 \mu M. \bullet (\Delta(\text{box}(\overline{-++-}, M), \text{box}(\overline{-+}, M))) \\
 &\rightarrow_{\text{R1} \cdot \text{R3}} \mu M. \Delta(\text{box}(\overline{+-+}, \text{box}(\overline{-++-}, M)), \text{box}(\overline{+-+}, \text{box}(\overline{-+}, M))) \\
 &\rightarrow_{\text{R2}}^2 \mu M. \Delta(\text{box}(\overline{+-++-}, M), \text{box}(\overline{+-+}, M)) \\
 &\rightarrow_{\text{R4} \cdot \text{R6}} \Delta(\text{src}(\text{fix}(\overline{+-++-})), \text{src}(\text{fix}(\overline{+-+}))) = \Delta(\text{src}(\infty), \text{src}(\underline{1})) \rightarrow_{\text{R7}} \text{src}(\underline{1})
 \end{aligned}$$

Hence `M` is not productive in \mathcal{T}^* (here it produces only one element).

The automated productivity prover has been applied to:

```
-- stream layer --  
M = 0:1:even(M)  
  
-- function layer --  
even(a:s) = a:odd(s)  
odd(a:s) = even(s)  
  
-- data layer --
```

JOERG'S EXAMPLE REVISITED

$M = 0:1:\text{even}(M)$

This is a pure stream specification for which we can decide productivity.

The translation of `even` is computed as follows:

$$\begin{aligned} [\text{even}] &= \Delta([\text{even}]_{1,0}) \\ [\text{even}]_{1,0} &= \mu(\text{even}_{1,0}). \wedge \left\{ -+ \wedge \left\{ \mu(\text{odd}_{1,0}). \wedge \left\{ - \wedge \left\{ \text{even}_{1,0} \right. \right. \right. \right. \\ &= \overline{-+-} \end{aligned}$$

The translation of `odd` is computed as follows:

$$\begin{aligned} [\text{odd}] &= \Delta([\text{odd}]_{1,0}) \\ [\text{odd}]_{1,0} &= \mu(\text{odd}_{1,0}). \wedge \left\{ - \wedge \left\{ \mu(\text{even}_{1,0}). \wedge \left\{ -+ \wedge \left\{ \text{odd}_{1,0} \right. \right. \right. \right. \\ &= \overline{---+} \end{aligned}$$

OUTPUT FOR JOERG'S EXAMPLE

We translate M into a pebbleflow net and collapse it to a source:

$$\begin{aligned} [M] &= \mu M. \bullet(\bullet(\overline{-+-}(M))) \\ &\rightarrow_c \mu M. +\overline{-+}(\bullet(\overline{-+-}(M))) \\ &\rightarrow_c \mu M. +\overline{-+}(\overline{+-+}(\overline{-+-}(M))) \\ &\rightarrow_c \mu M. +\overline{+-}(\overline{-+-}(M)) \\ &\rightarrow_c \mu M. ++\overline{-+-}(M) \\ &\rightarrow_c 4 \end{aligned}$$

The specification of M is not productive (produces 4 elements).

Data-Oblivious Stream Productivity

Jörg Endrullis

Clemens Grabmayer

Dimitri Hendriks

Mon Jun 16 14:47:52 CEST 2008

The automated productivity prover has been applied to:

```
-- stream layer --
M = 0:0:0:0:even(M)

-- function layer --
zip(a:s,t) = a:zip(t,s)
tail(a:s) = s
inv(0:s) = 1:inv(s)
inv(1:s) = 0:inv(s)
even(a:s) = a:odd(s)
odd(a:s) = even(s)

-- data layer --
```

This is a pure stream specification for which we can decide productivity.

We translate M into a pebbleflow net and collapse it to a source:

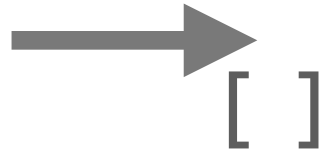
$$\begin{aligned}
[M] &= \mu M. \bullet(\bullet(\bullet(\bullet(\overline{-+-}(M)))))) \\
&\rightarrow_c \mu M. +\overline{-+}(\bullet(\bullet(\bullet(\bullet(\overline{-+-}(M)))))) \\
&\rightarrow_c \mu M. +\overline{-+}(\overline{-+}(\bullet(\bullet(\bullet(\bullet(\overline{-+-}(M)))))) \\
&\rightarrow_c \mu M. +\overline{-+}(\bullet(\bullet(\overline{-+-}(M)))) \\
&\rightarrow_c \mu M. +\overline{-+}(\overline{-+}(\bullet(\overline{-+-}(M)))) \\
&\rightarrow_c \mu M. ++\overline{-+}(\bullet(\overline{-+-}(M))) \\
&\rightarrow_c \mu M. ++\overline{-+}(\overline{-+}(\overline{-+-}(M))) \\
&\rightarrow_c \mu M. +++\overline{-+}(\overline{-+-}(M)) \\
&\rightarrow_c \mu M. ++++\overline{-+-}(M) \\
&\rightarrow_c 8
\end{aligned}$$

The specification of M is not productive (produces 8 elements).

Terms
 $S = 0: \text{tail}(S)$

orthogonal TRS

abstraction
translation



productivity
preserving

*rational pebble
net with boxes*
 $\mu S. \bullet(\text{--}+(S))$

orthogonal CRS

*box composition and fixed point
defined coinductively or with
infinitary rewriting*

productivity
preserving

simplification to
source nets
 k or ∞

not orthogonal TRS,
but SN and CR

*lazy natural number
nets* $\mathbb{N}^\infty = \{ \underline{0}, \underline{1}, \underline{2}, \dots, \underline{\infty} \}$

Productivity is decidable for pure stream specifications

$$R_1 \quad \bullet(N) \rightarrow +\overline{-\overline{+}}(N)$$

composition

$$R_2 \quad \sigma(\tau(N)) \rightarrow (\sigma \circ \tau)(N)$$

simplification rules for pebble nets, not

$$R_3 \quad \mu x.N \rightarrow N \quad (x \notin \text{fv}(N))$$

orthogonal, but SN

fixed point

$$R_4 \quad \mu x.\sigma(x) \rightarrow \underline{\text{fix}(\sigma)}$$

and CR. Normal forms

$$R_5 \quad \underline{m} \Delta \underline{n} \rightarrow \min(\underline{m}, \underline{n})$$

are finite sources \underline{n} , or

infinite source $\underline{\infty}$

(lazy natural numbers)

$$R_6 \quad \sigma(\underline{k}) \rightarrow \sigma \circ \underline{k}$$

$$R_7 \quad \sigma(N_1 \Delta N_2) \rightarrow \sigma(N_1) \Delta \sigma(N_2)$$

$$R_8 \quad \mu x.(N_1 \Delta N_2) \rightarrow \mu x.N_1 \Delta \mu x.N_2$$

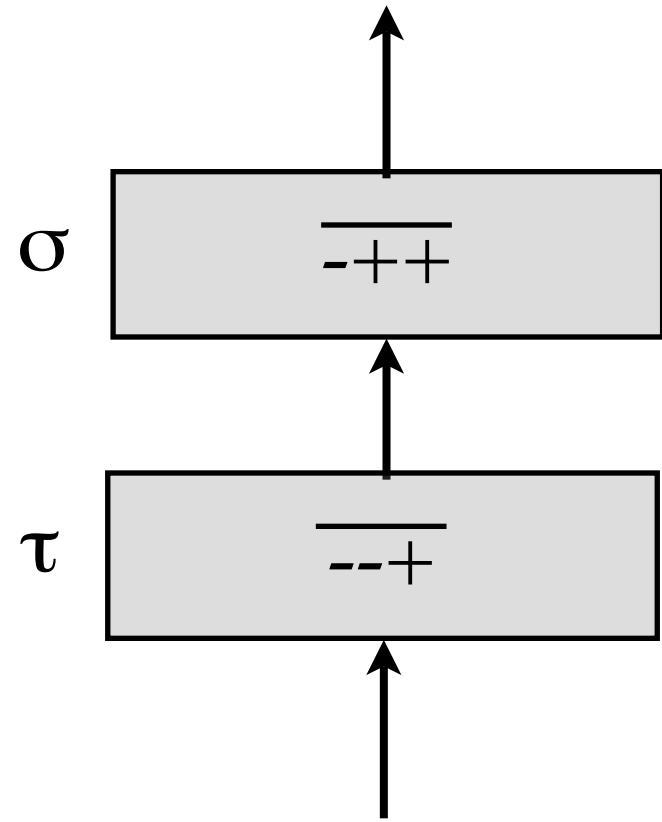
$$R_9 \quad \mu x.x \rightarrow \underline{0}$$

Composition of boxes $\sigma \circ \tau$

$$+\sigma \circ \tau = +(\sigma \circ \tau)$$

$$-\sigma \circ +\tau = \sigma \circ \tau$$

$$-\sigma \circ -\tau = -(-\sigma \circ \tau)$$



coinductive definition, or equivalently, defined by infinitary rewriting; under the proviso that I/O-sequences contain infinitely many +’s; there are no infinite sinks.

composition of boxes is associative

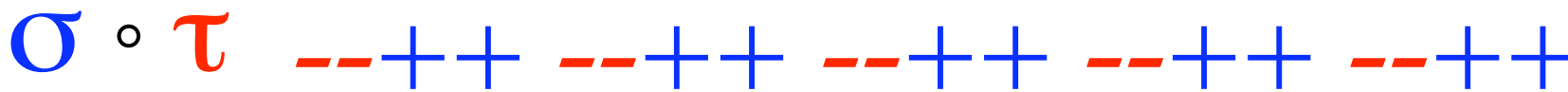
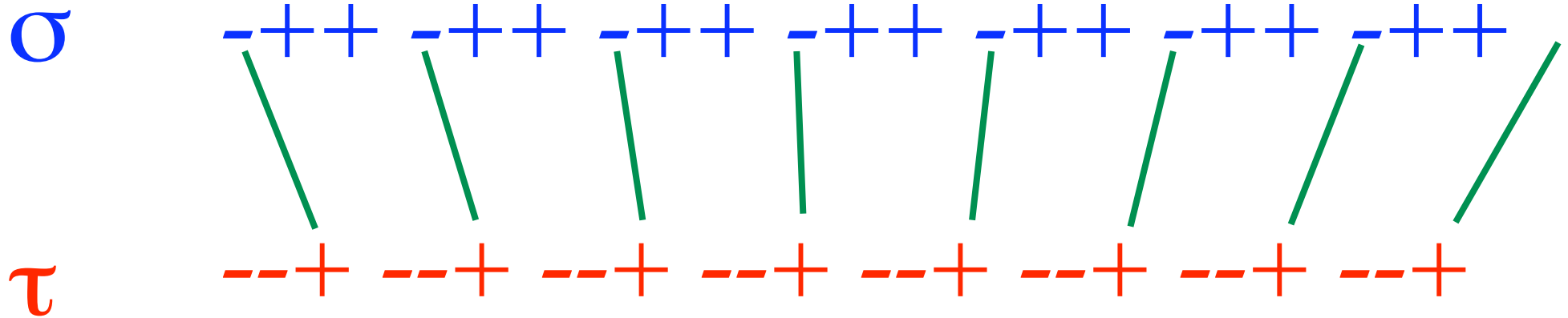
Example:

Composition of boxes $\sigma \circ \tau$

$$+\sigma \circ \tau = +(\sigma \circ \tau)$$

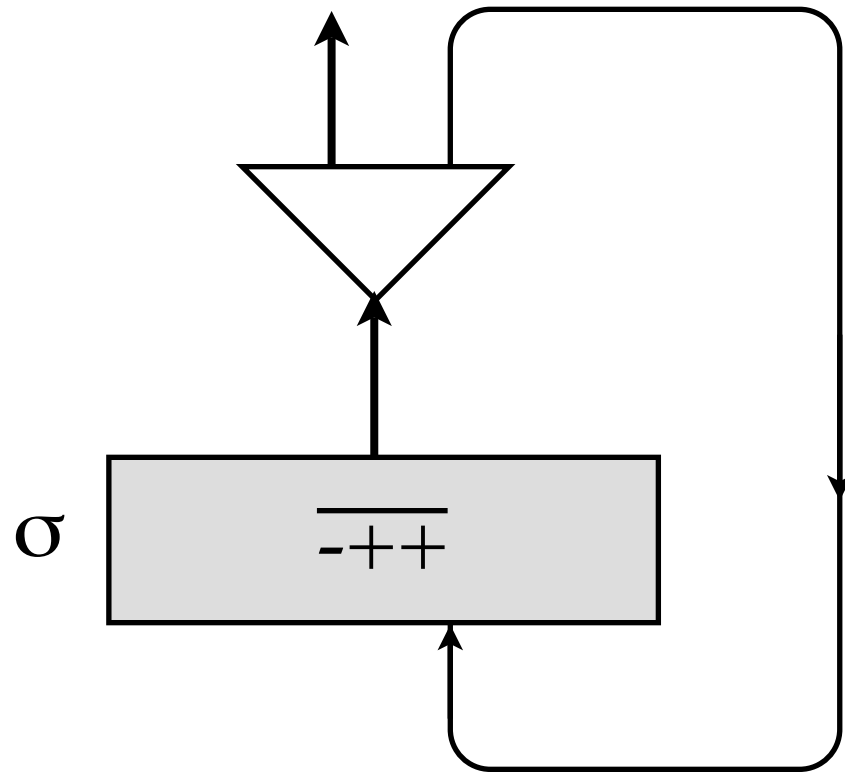
$$-\sigma \circ +\tau = \sigma \circ \tau$$

$$-\sigma \circ -\tau = -(-\sigma \circ \tau)$$



again rational

Fixed point of boxes $\text{fix}(\sigma) = \sigma \circ \text{fix}(\sigma) \in N^\infty$

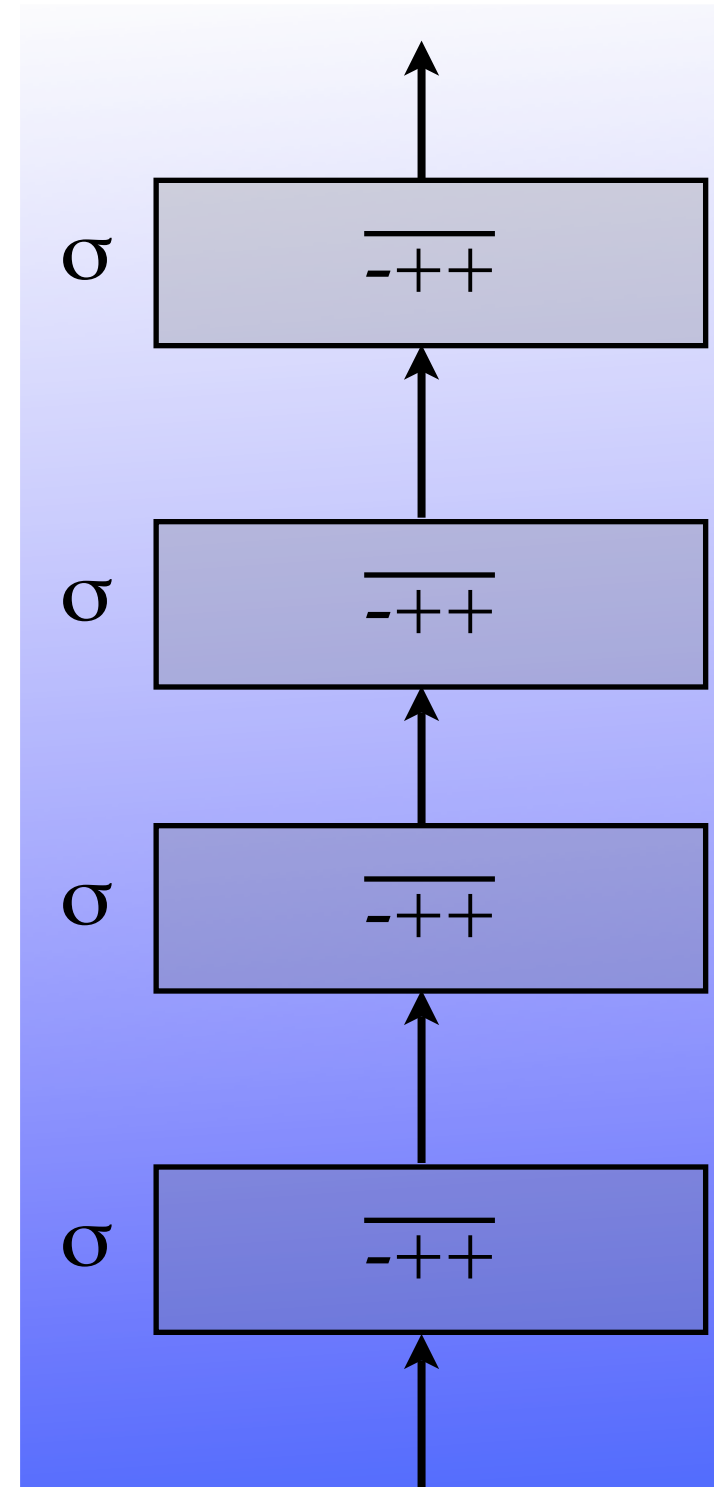


$$\text{fix}(+\sigma) = s(\underline{\text{fix}(\delta(\sigma))})$$

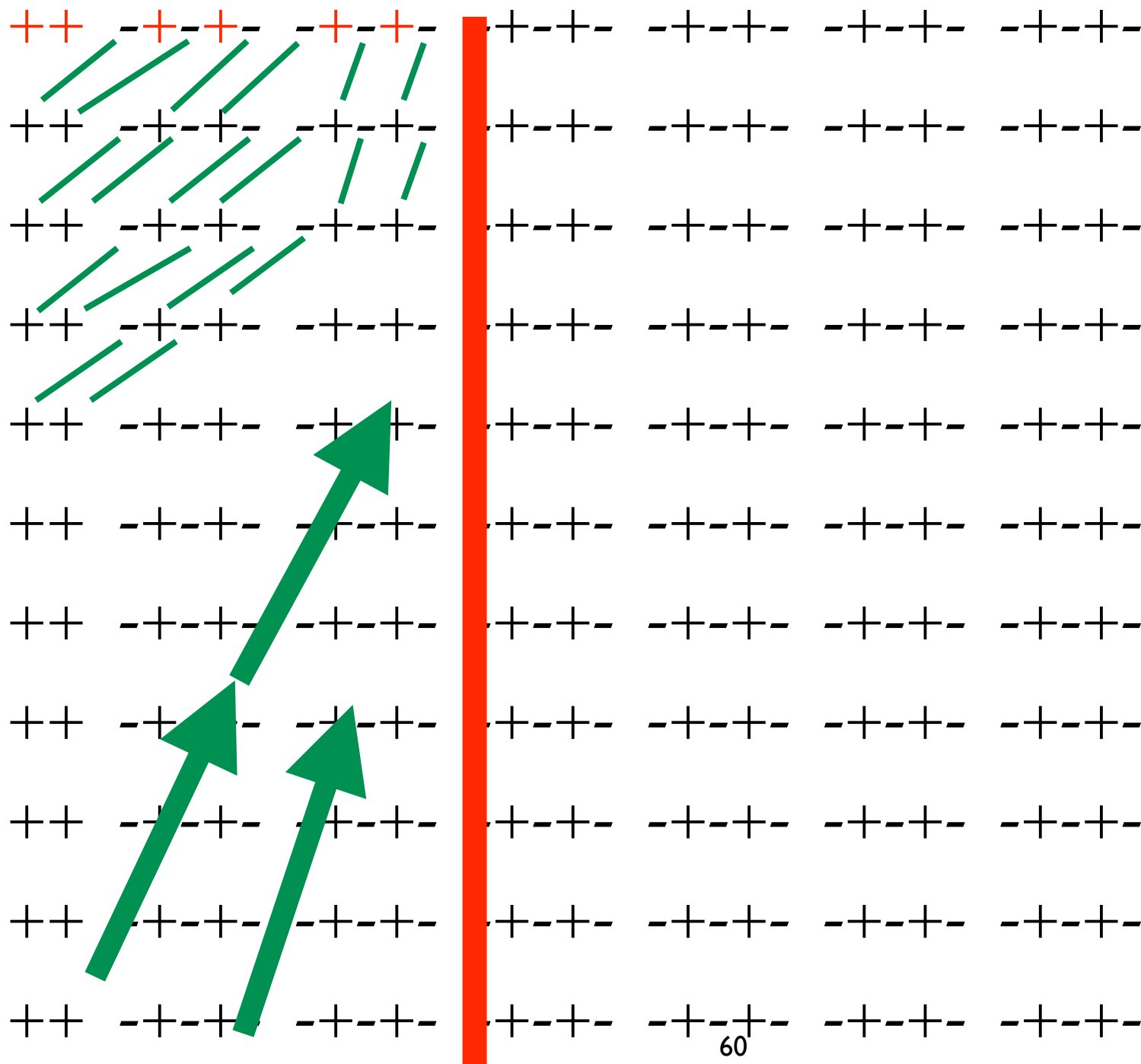
$$\text{fix}(-\sigma) = \underline{0}$$

$$\delta(+\sigma) = + \delta(\sigma)$$

$$\delta(-\sigma) = \sigma$$



$$\sigma = ++\alpha, \alpha = \overline{-+--+} \quad \text{fix}(\sigma) = 6$$



Example: fixed point $\text{fix}(\sigma)$

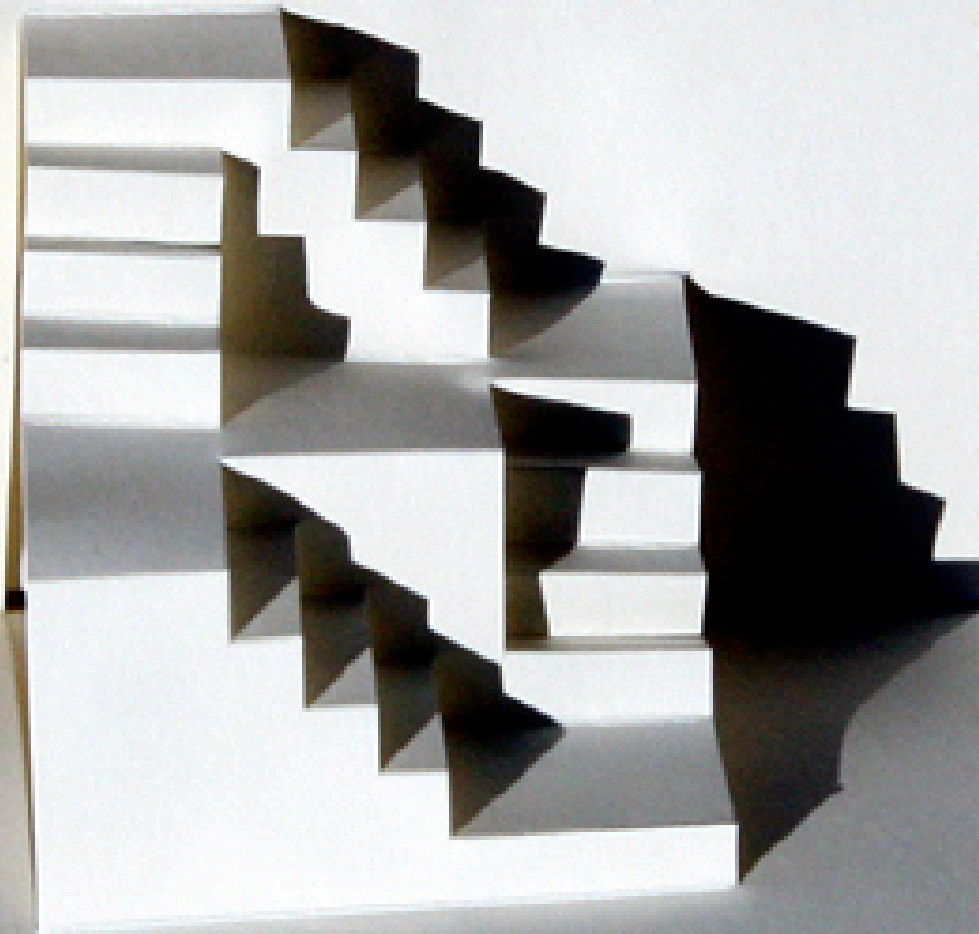
$$\sigma = ++\alpha, \alpha = \overline{-+-+}$$

$$\begin{aligned} \text{fix}(\sigma) &= 1 + \text{fix}(\delta(+\alpha)) = \\ &= 1 + \text{fix}(+(\delta(\alpha))) = \\ &= 1 + \text{fix}(+(\delta(-+-\alpha))) = \\ &= 1 + \text{fix}(+(+-\alpha)) = \\ &= 1 + \text{fix}(+(+-\alpha)) = \\ &= 2 + \text{fix}(\delta(+--\alpha)) = \\ &= 2 + \text{fix}(+(\delta(-+-\alpha))) = \\ &= 2 + \text{fix}(+(+-\alpha)) = \\ &= 3 + \text{fix}(\delta(+-\alpha)) = 3 + \text{fix}(+\delta(-\alpha)) = 3 + \text{fix}(+\alpha) = \\ &= 3 + \text{fix}(+\alpha) = 4 + \text{fix}(\delta(\alpha)) = 4 + \text{fix}(+-\alpha) = \\ &= 5 + \text{fix}(\delta(-+-\alpha)) = 5 + \text{fix}(+-\alpha) = 6 + \text{fix}(\delta(+-\alpha)) = \\ &= 6 + \text{fix}(-\alpha) = 6 + 0 = 6 \end{aligned}$$

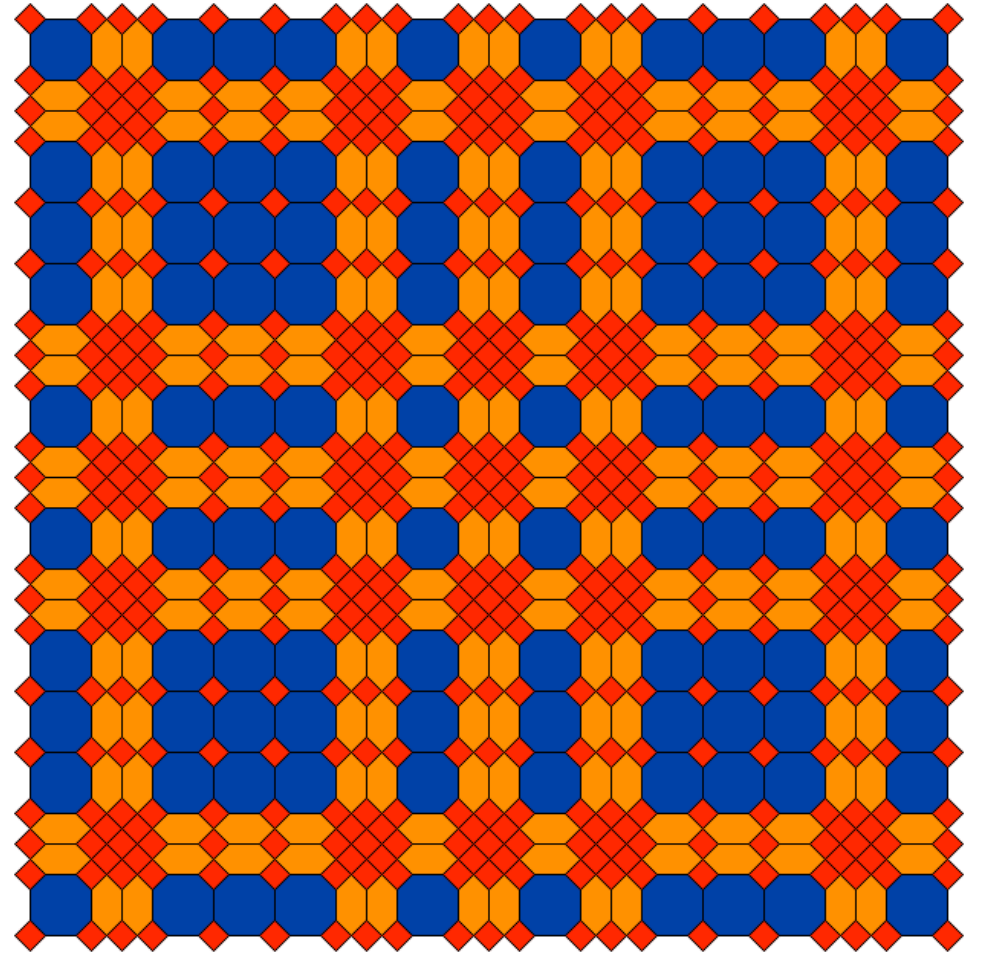
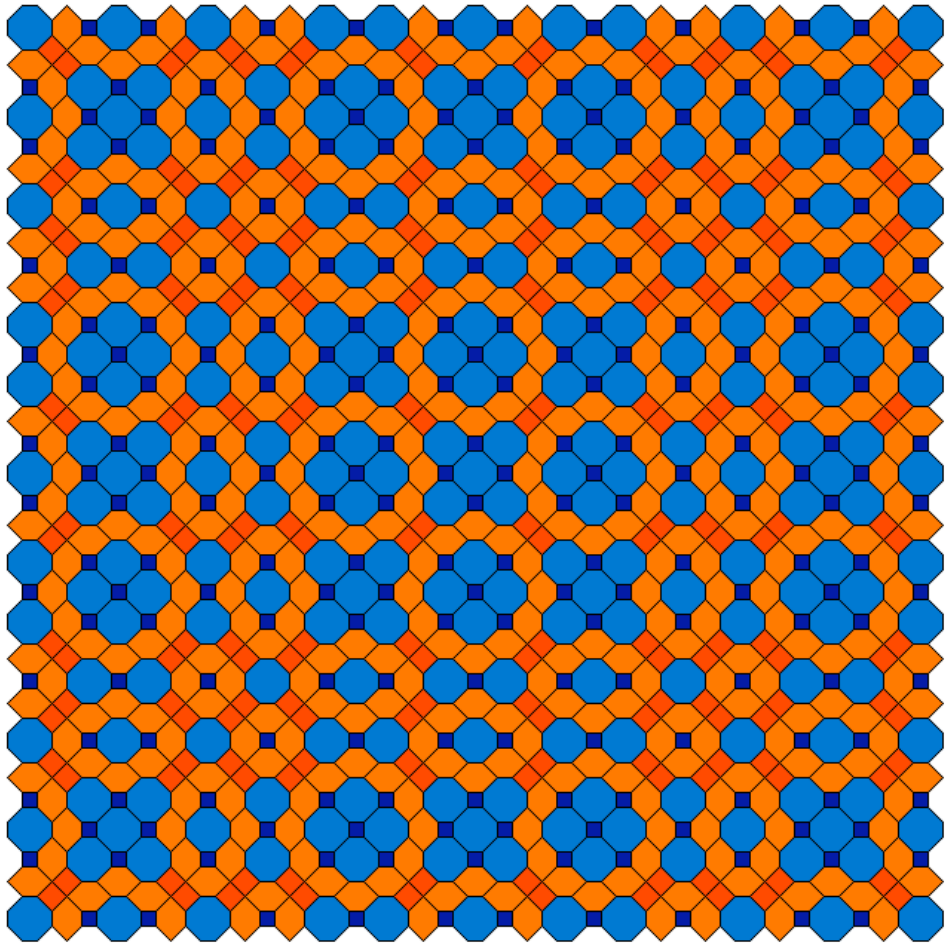
$$\begin{aligned} \text{fix}(+\sigma) &= s(\text{fix}(\delta(\sigma))) \\ \text{fix}(-\sigma) &= \underline{0} \\ \delta(+\sigma) &= +\delta(\sigma) \\ \delta(-\sigma) &= \sigma \end{aligned}$$

Chapter 5

Comparing streams



TAPESTRIES OF MORSE AND TOEPLITZ SEQUENCE



WHEN THUE-MORSE MEETS KOCH

JUN MA AND JUDY HOLDENER

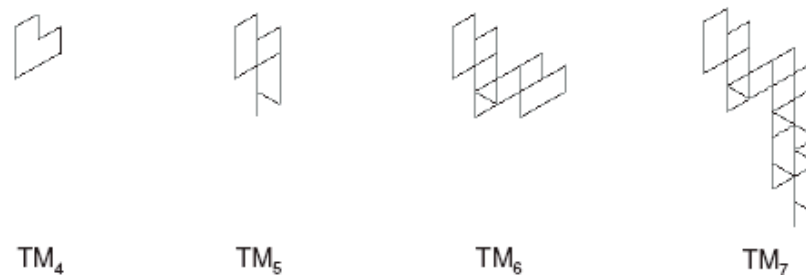


FIGURE 1. Thue-Morse turtle programs of degrees 4 through 7

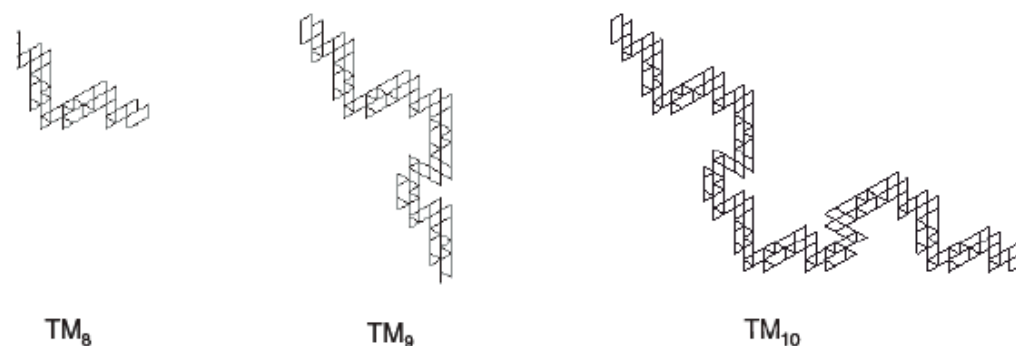


FIGURE 2. Thue-Morse turtle programs of degrees 8 through 10

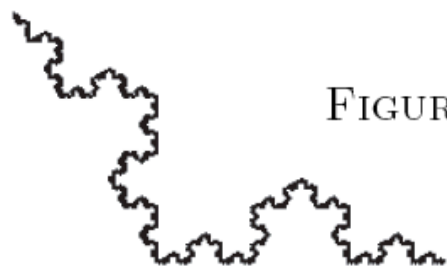
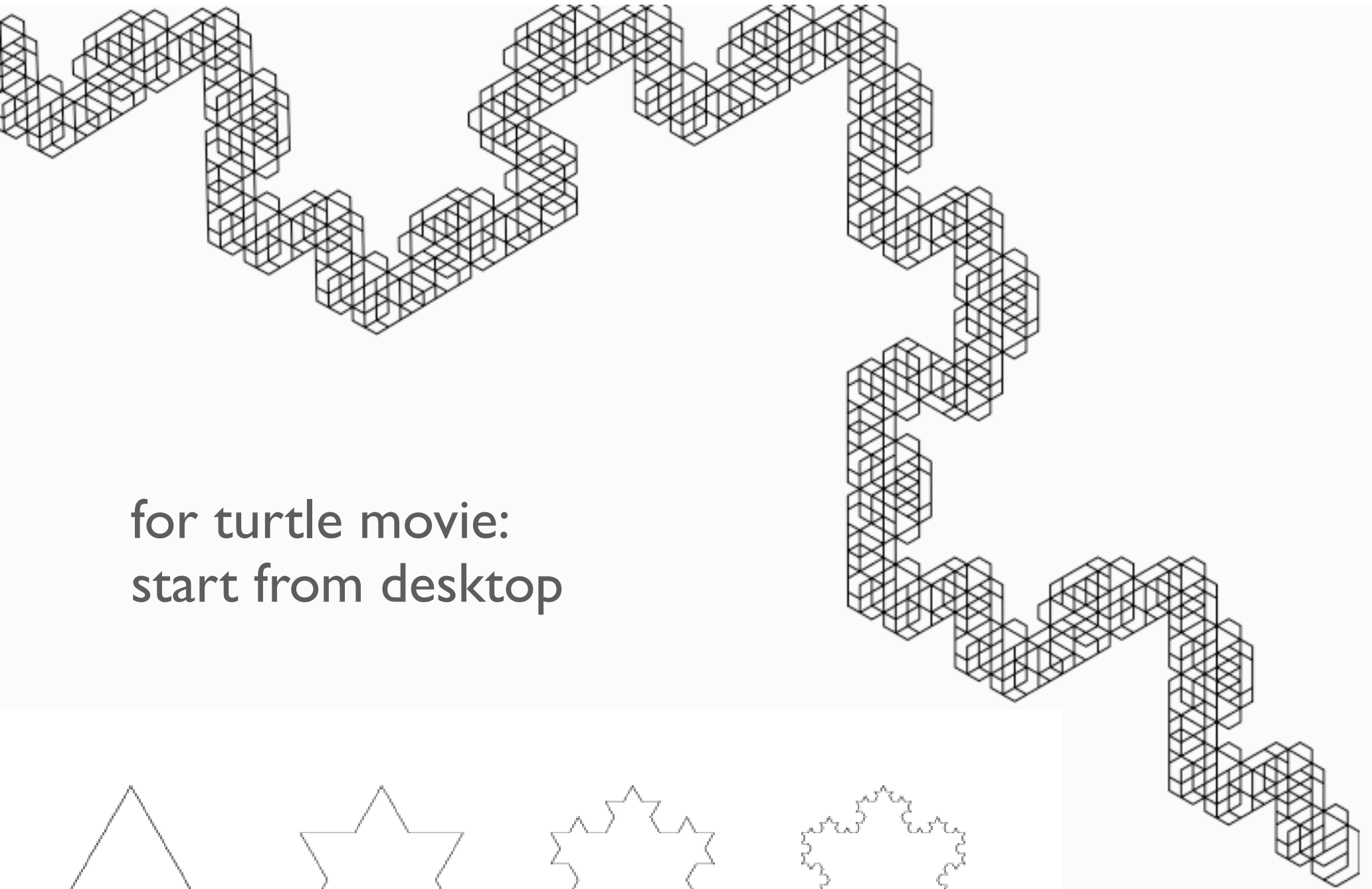
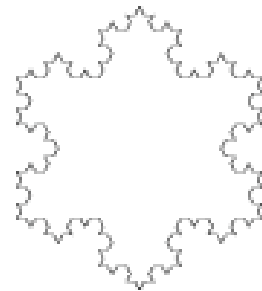
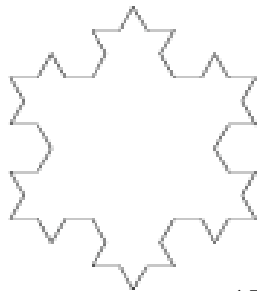
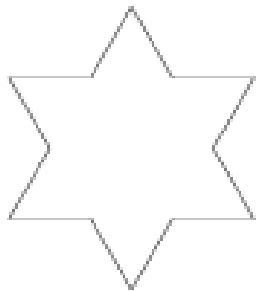
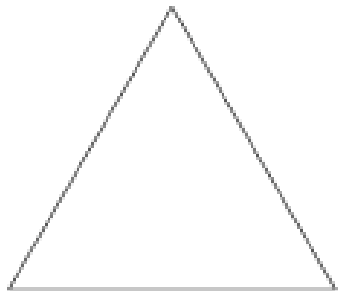


FIGURE 3. The Thue-Morse turtle program of degree 14



for turtle movie:
start from desktop

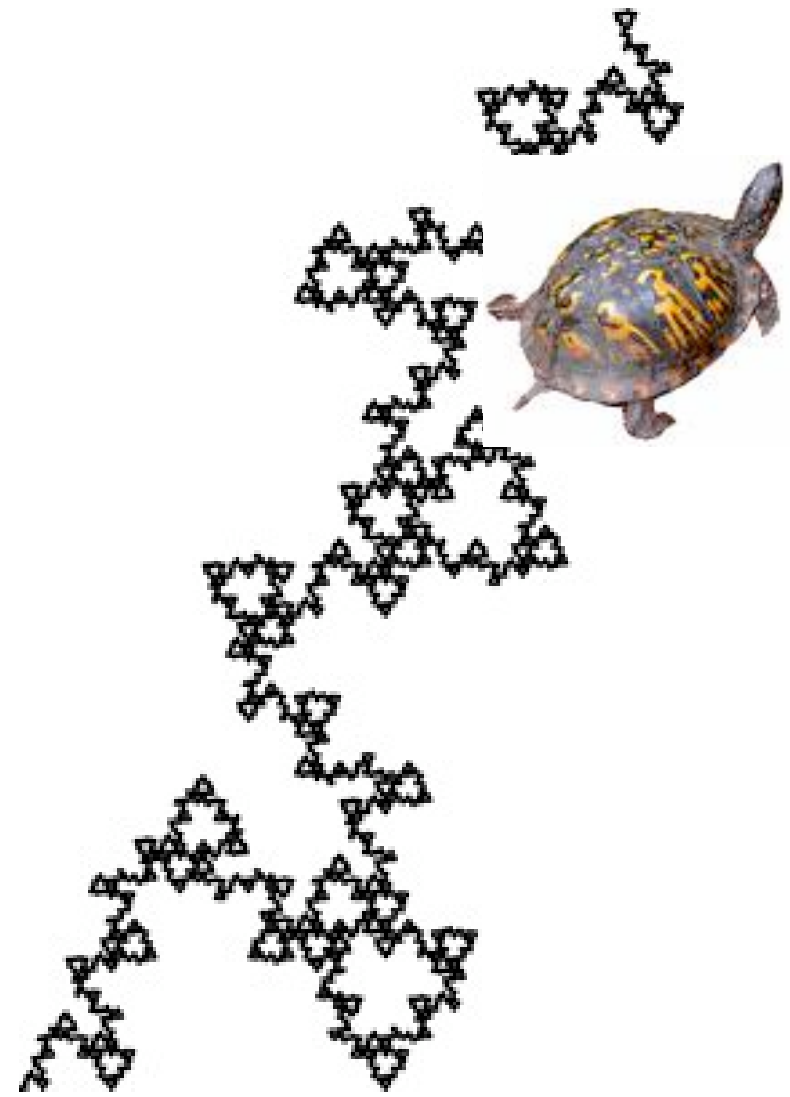
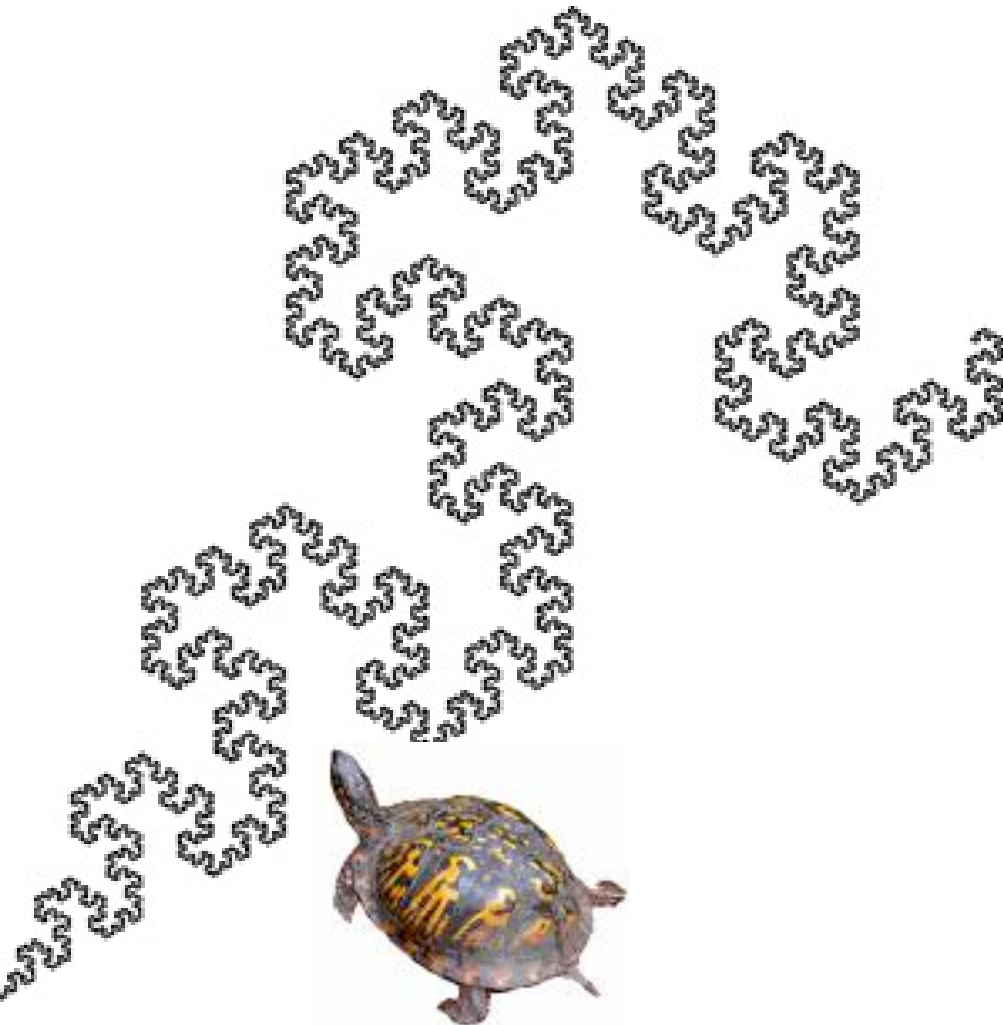


Mephisto Waltz as stream specification

D0L system:

0 → 001

1 → 110



mephisto = h (0: (tail mephisto))

where

$$h (0:s) = 0:0:1:(h s)$$

$$h (1:s) = 1:1:0:(h s)$$

001 001 110 001 001 110 110 110 001 001 001 110 001 001 110 110 110 001

Fibonacci

$a \rightarrow ab,$

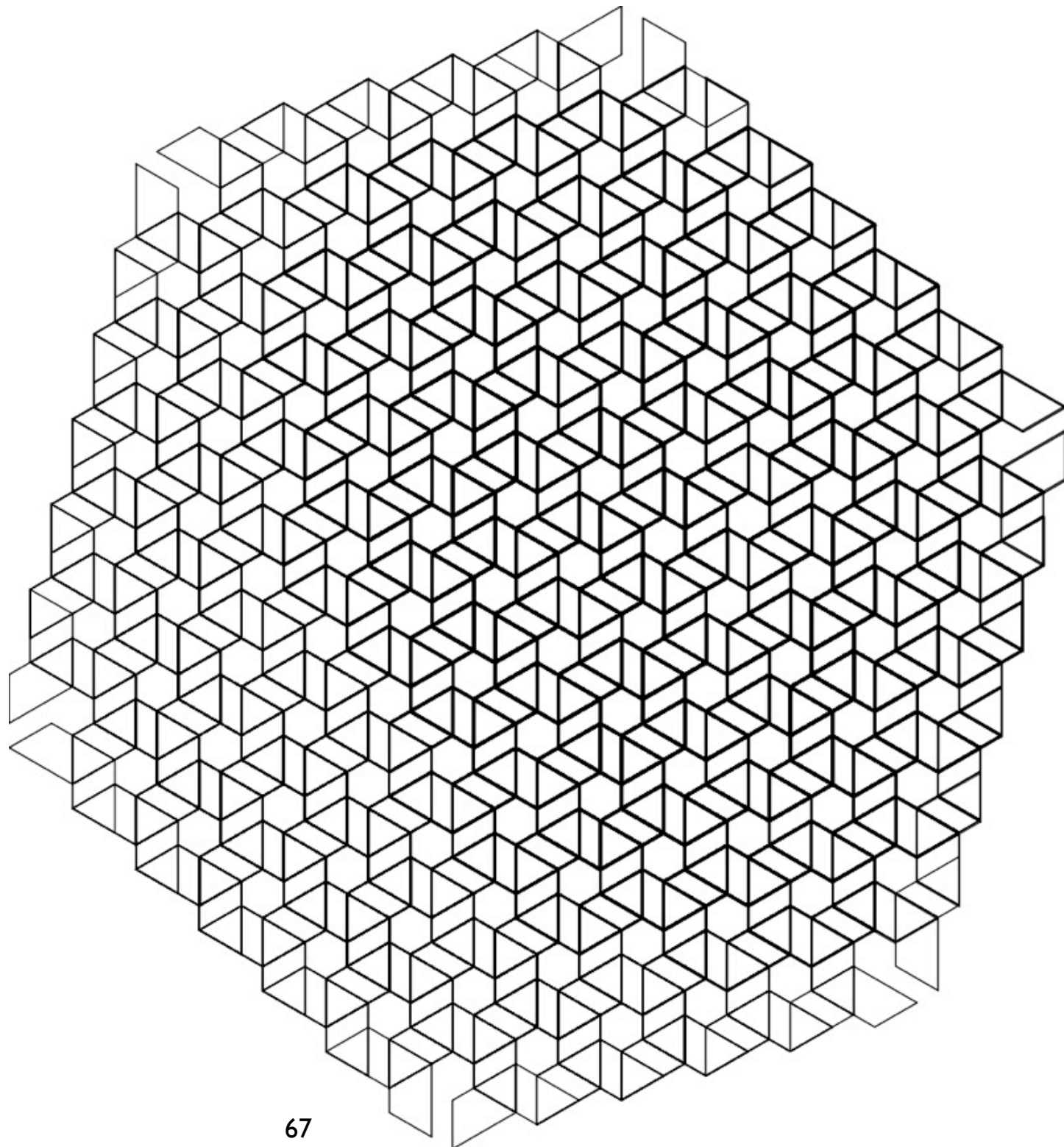
$b \rightarrow a$

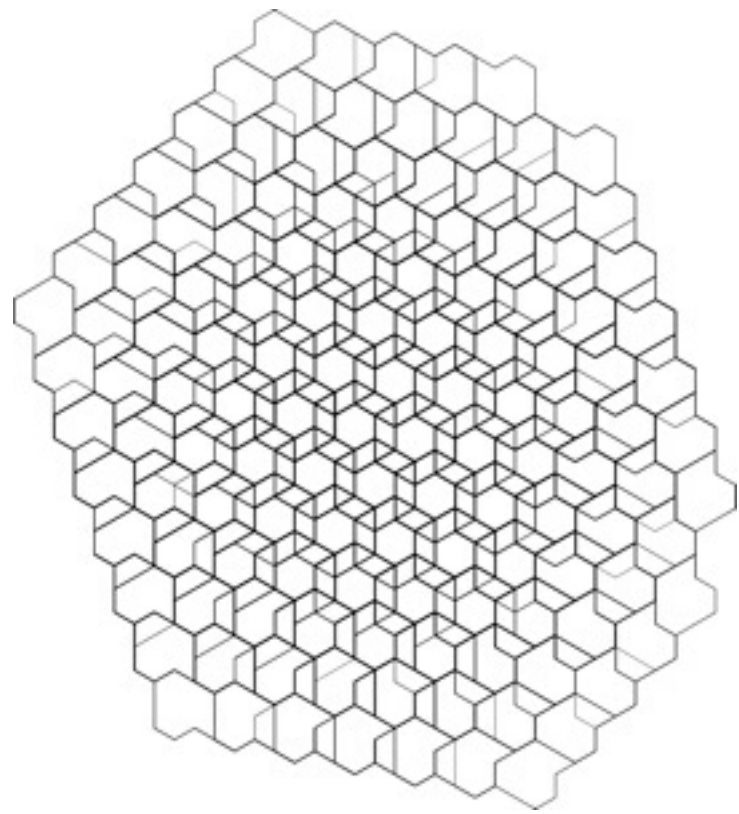
ab

aba

abaab

abaababa





Fibonacci turtle trajectory

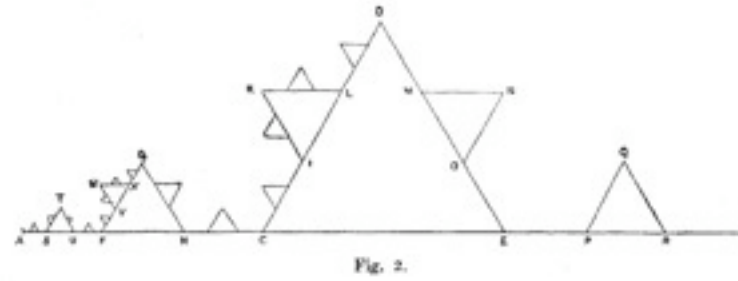
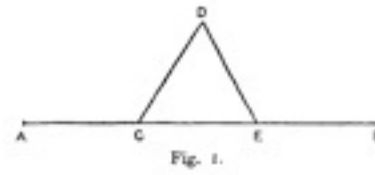
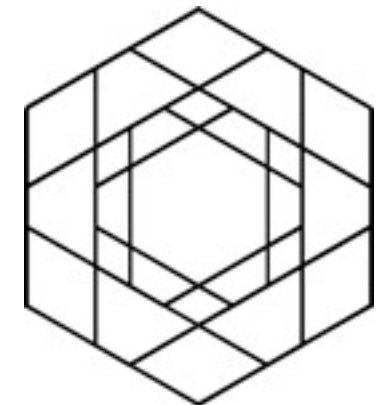
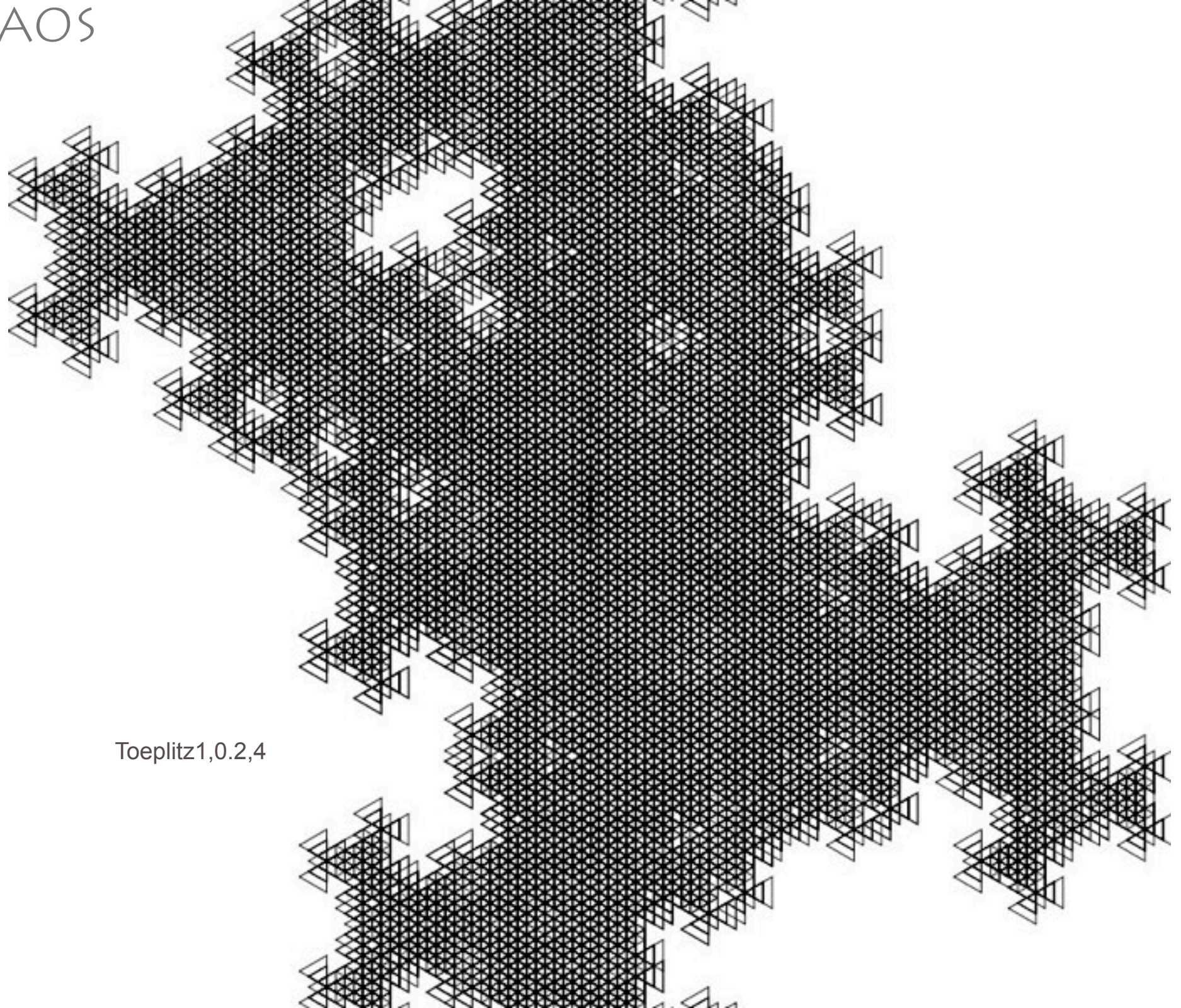


Figure in von Koch's paper (1906)

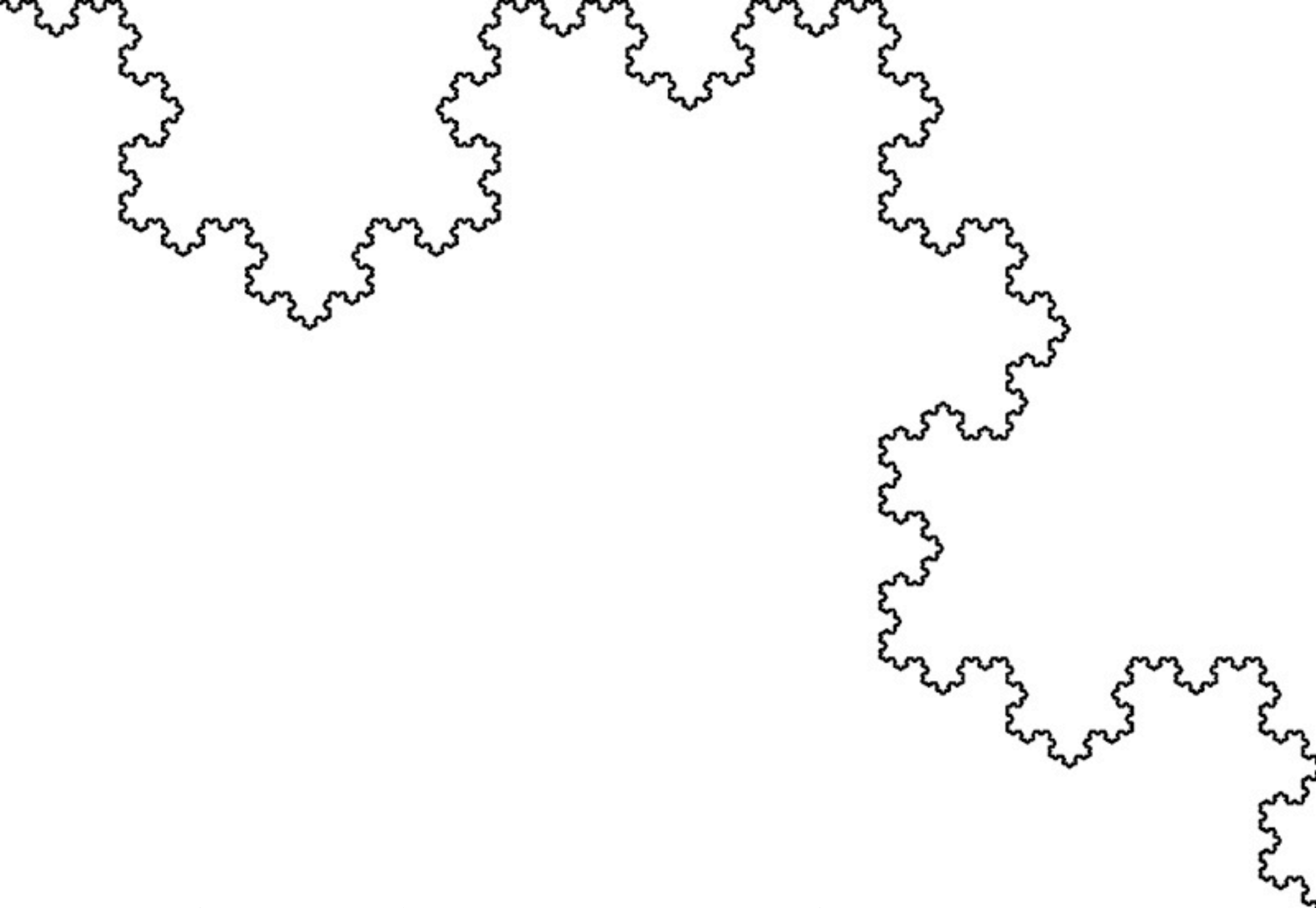


Toeplitz turtle trajectories. The Toeplitz stream produces, as a turtle program, sometimes fractal-like patterns, and with other choices for movements corresponding to 0, 1, also various non-fractal patterns.

CHAOS



Toeplitz1,0.2,4



THE SNOW FLAKE OF HELGE VON KOCH

Question:

toeplitz and morse are strongly related:

diff morse = toeplitz,

toeplitz = 1-undiff morse.

Is sierpinsky related to morse? Is it 'essentially' different? How can such streams be compared?

E.g. define a notion of transforming streams via some kind of finite state transducers?

Yes, we can!

D0L sequences (morphic sequences)

Definition

A D0L sequence is defined as $D = \langle \Sigma, w, \phi \rangle$, where:

- ▶ Σ is an alphabet
- ▶ $w \in \Sigma^+$ is the starting word (called axiom)
- ▶ $\phi : \Sigma \rightarrow \Sigma^*$ is a morphism

The stream M_D defined by D is

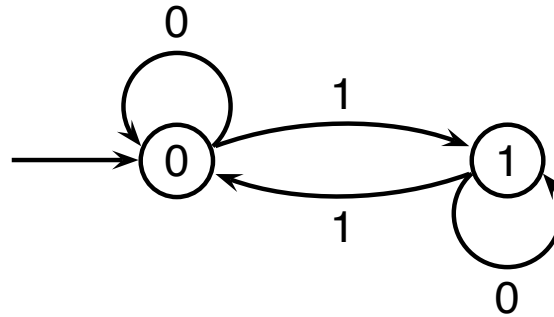
$$M_D = \lim_{n \rightarrow \infty} \phi^n(w)$$

For example Thue-Morse is defined by: $\langle \{0, 1\}, 0, 0 \mapsto 01 \text{ \& } 1 \mapsto 10 \rangle$.

- ▶ 0
- ▶ $01 = \phi(0)$
- ▶ $0110 = \phi(01)$
- ▶ $01101001 = \phi(0110)$

Automatic Sequences

A DFAO A is a det., finite automaton with output $\in \Sigma$ in each state:



The output of $A(w)$ is the output of the final state when A reads w .

Definition

A sequence M is k -automatic if there exists DFAO A :

$$M(i) = \text{output of } A \text{ reading the representation of } i \text{ to the base } k$$

The above automaton defines Thue-Morse 0110...:

$$A(0) = 0, A(1) = 1, A(10) = 1, A(11) = 0, \dots$$

Automatic sequences (alternative definition)

Theorem (Cobham)

A sequence M is k -automatic if M is the coding ($c : \Sigma \rightarrow \Gamma$) of a D0L word with respect to a k -uniform morphism ϕ , ($\forall a \in \Sigma. |\phi(a)| = k$).

Proof sketch.

Let M be a k automatic sequence and A the corresponding DFAO:

- ▶ $\delta : Q \times \{0, \dots, k-1\} \rightarrow Q$ transition function, and
- ▶ $\Lambda : Q \rightarrow \Sigma$ output.

Define a D0L sequence M by $w = q_0$ and for all $q \in Q$ let:

$$\phi(q) = \delta(q, 0)\delta(q, 1)\dots\delta(q, k-1),$$

w.l.o.g. $\delta(q_0, 0) = q_0$. Then the $M(k \cdot i + j) = \delta(M(i), j)$:

- ▶ $\delta(0)\dots\delta(k-1)$
- ▶ $\delta(0)\dots\delta(k-1)\delta(\delta(1), 0)\delta(\delta(1), 1)\dots\delta(\delta(k-1), k-1)\dots$

Toeplitz sequences

Definition (Toeplitz sequences)

Let $x \in \Sigma(\Sigma \uplus \{?\})^*$, then the Toeplitz word T_x is

$T_x = x^\omega$, where at the places of ? we successively fill in T_x itself

Example (Toeplitz sequence, $x = 101?$)

$$x^\omega = 101?101?101?101?101?101?...$$

$$T_x = 101110101011101110111010...$$

Sturmian sequences

Definition (Sturmian sequences)

Let θ, ρ be real number, then for $n \geq 1$ define

$$s_n = \lfloor (n+1)\theta + \rho \rfloor - \lfloor n\theta + \rho \rfloor$$

$$s'_n = \lceil (n+1)\theta + \rho \rceil - \lceil n\theta + \rho \rceil$$

then $s_1 s_2 s_3 \dots$ and $s'_1 s'_2 s'_3 \dots$ are Sturmian sequences.

Example (Fibonacci sequence)

For $\theta = (\sqrt{5} - 1)/2$ we get the Fibonacci sequence F :

$$F = 1011010110\dots$$

Pure stream format

Example

$\text{alt} \rightarrow 0 : 1 : \text{alt}$

$\text{fib} \rightarrow 0 : 1 : \text{sum}(\text{fib})$

$T \rightarrow 0 : \text{zip}(\text{inv}(T), \text{tail}(T))$

$J \rightarrow 0 : 1 : \text{even}(J)$

$\text{tail}(x : \sigma) \rightarrow \sigma$

$\text{inv}(x : \sigma) \rightarrow (1 - x) : \text{inv}(\sigma)$

$\text{zip}(x : \sigma, \tau) \rightarrow x : \text{zip}(\tau, \sigma)$

$\text{even}(x : \sigma) \rightarrow x : \text{odd}(\sigma)$

$\text{odd}(x : \sigma) \rightarrow \text{even}(\sigma)$

$\text{sum}(x : \sigma) \rightarrow \text{sum_aux}(\sigma, x)$

$\text{sum_aux}(y : \sigma, x) \rightarrow (x + y) : \text{sum_aux}(\sigma, y)$

Complexity of streams

Definition

M is recurrent if every prefix p of M occurs infinitely often in M .

Definition

M is uniform recurrent for every prefix p of M there exists $n \in \mathbb{N}$ such that every subword of M of length $\geq n$ contains p .

Definition

M is eventual recurrent if a suffix of M is recurrent.

M is eventual uniform recurrent if a suffix of M is uniform recurrent.

Subword complexity

Definition

The subword complexity $p_M(n)$ of a stream M is the number of distinct length- n subwords of M .

Complexity bounds for D0L and automatic sequences:

- ▶ morphic sequences: linear
- ▶ automatic sequences: quadratic
- ▶ Sturmian sequences: lowest possible

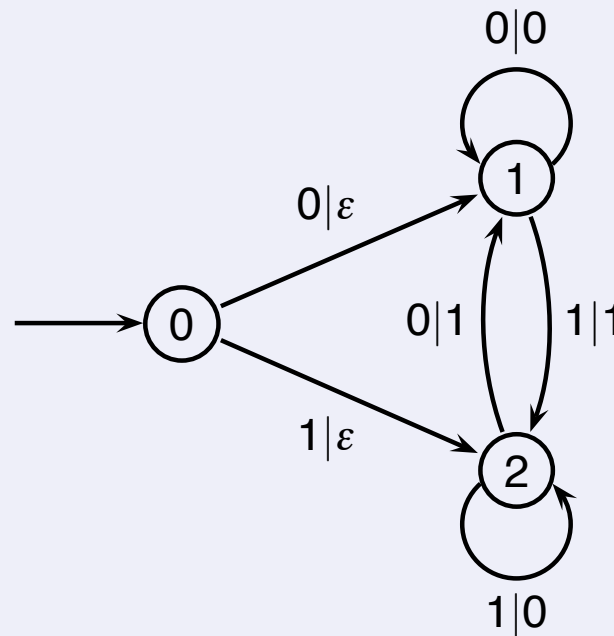
Transducing streams

We transduce streams using **deterministic Mealy automata (DMA)**.

- ▶ output words $\in \Sigma^*$ along the edges

Example

The following automaton computes the diff of a stream:



Thus it reduces Morse to Toeplitz.

01101001 ... \rightarrow 1011101 ...

Partial order of stream degrees

We write

- ▶ $M \triangleright_A N$ to denote that the DMA A reduces M to N , and
- ▶ $M \triangleright N$ if $M \triangleright_A N$ for some DMA A .

Reducibility is reflexive and transitive, that is $\triangleright^* \subseteq \triangleright$.

Definition (Equivalence of streams)

$$\diamond := \triangleright \cap \triangleleft$$

We use $\sigma^\diamond := \{\tau \mid \sigma \diamond \tau\}$ to denote the equivalence class of σ .

Note that \triangleright implies a partial order on the equivalence classes w.r.t. \diamond .

- ▶ We are interested in the hierarchy of streams created by \triangleright .

Hierarchy of streams

- ▶ The least degree 0 are the eventually periodic streams.
- ▶ Toeplitz and Morse interreducible.
- ▶ Morphic streams closed under transducers.
- ▶ OPEN PROBLEM: is Sierpinsky interreducible with Morse?
- ▶ There exist infinite ascending chains.
- ▶ There exist streams without supremum.

Primes

Definition

A stream M is prime if there exists no N such that:

$$M \triangleright N \qquad 0 \not\triangleright N \qquad N \not\triangleright M,$$

that is N is strictly in between M and 0.

- ▶ OPEN PROBLEM: Is Morse prime? (experiments confirm that)
- ▶ OPEN PROBLEM: How many primes are out there?

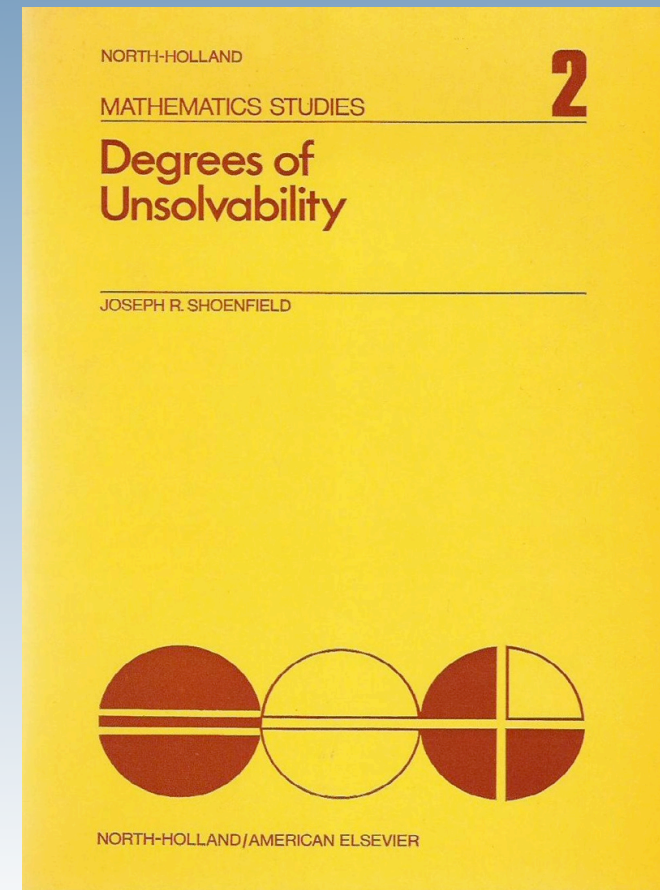
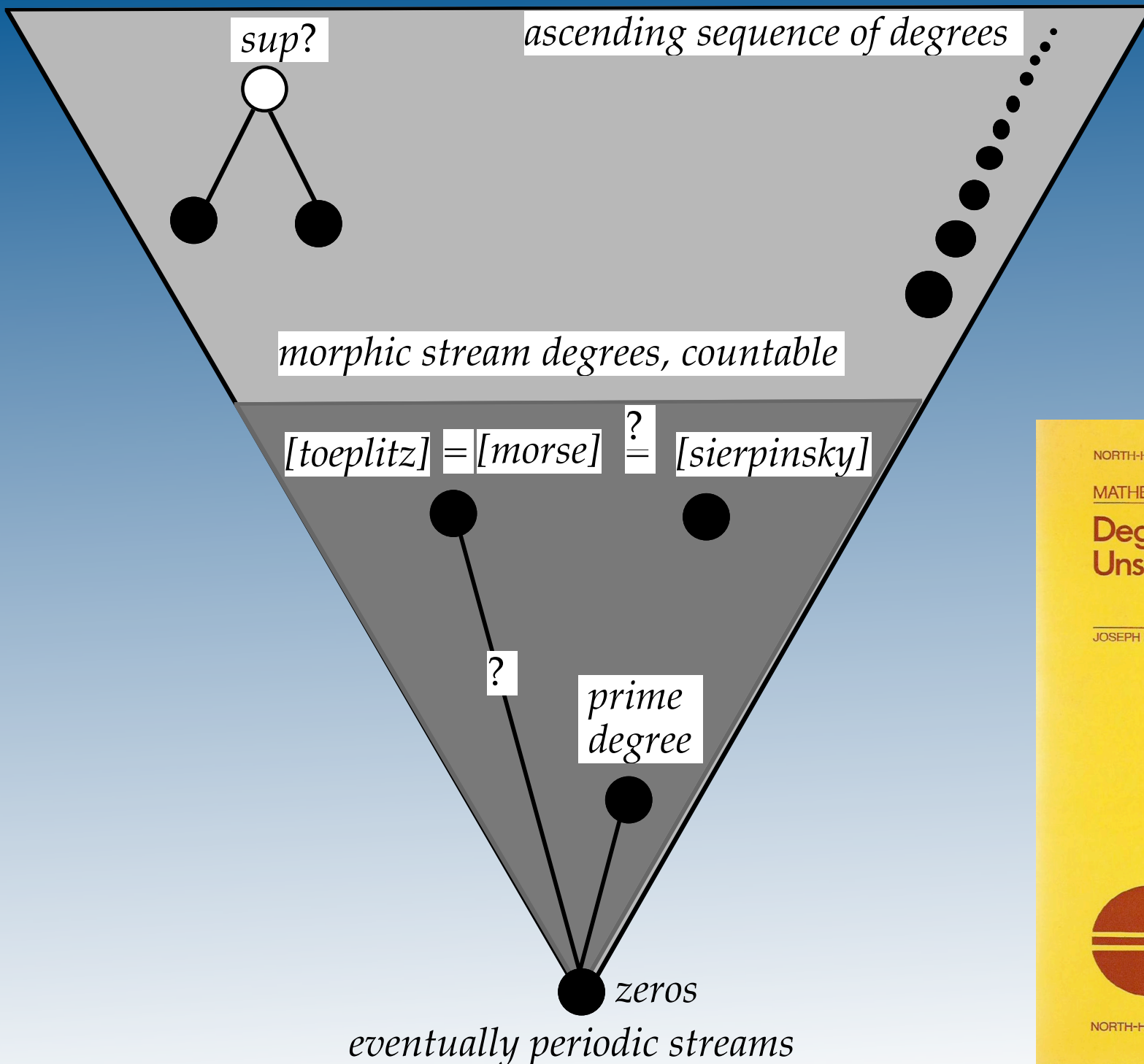
Example (Example of a prime stream)

$$f(n) := \sum_{i=0}^n i = \frac{n \cdot (n+1)}{2}$$

$$M(n) := \begin{cases} 1, & \text{if } \exists m \in \mathbb{N}. n = f(m) \\ 0, & \text{otherwise} \end{cases}$$

$M = 1101001000100001000001 \dots$

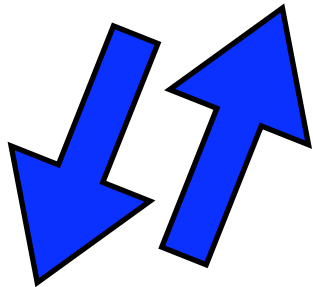
partial order of stream degrees, uncountable



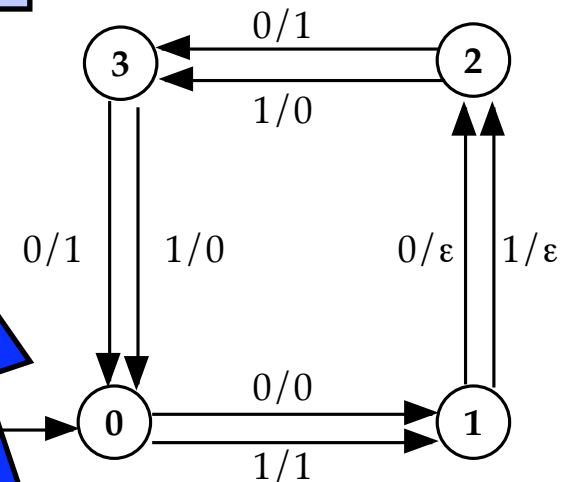
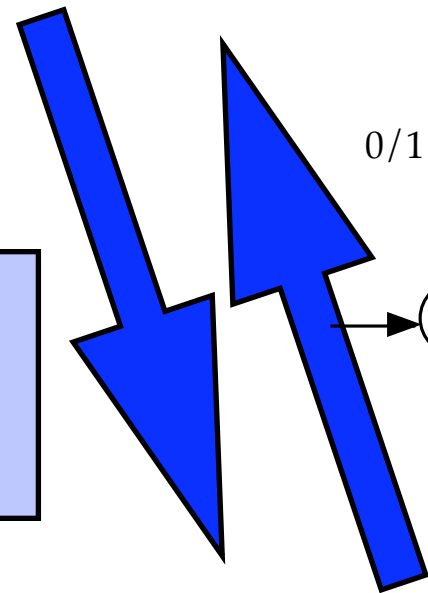
two failing experiments to cut the morse diamond into smaller nontrivial (i.e. not evt periodic) pieces



morse = |00|0| |00||0|00|...



3-morse = |00|0| |00||0|00|...



morse/3 = |00|0| |00||0|00|...
 = |||||0| |||||0| |||||0|...

