Grand Dyck consecutive patterns avoiding binary words

A.Bernini S.Bilotta E.Pergola R.Pinzani

UNIVERSITA' DEGLI STUDI DI FIRENZE

Dipartimento di Matematica e Informatica

Permutation Patterns 2013

Table of contents

1. Definitions and notations
2. A general approach
3. Main Result
4. Enumeration
5. Generalization
6. Further developments

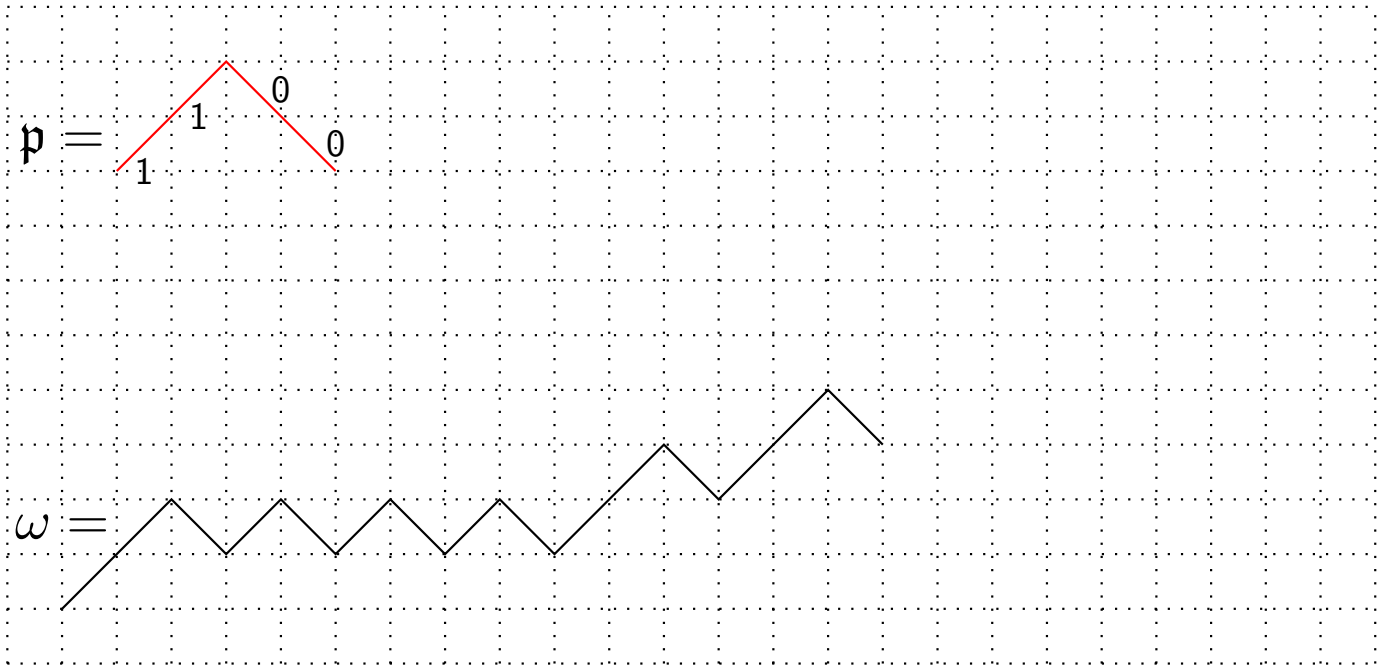
Binary words avoiding a pattern $\mathfrak{p}$

Let $F \subset \{0,1\}^*$ be the class of binary words ω such that $|\omega|_0 \leq |\omega|_1$ for any $\omega \in F$, $|\omega|_0$ and $|\omega|_1$ are the number of zeroes and ones in ω, respectively.

We are interested in studying the subclass $F^{[\mathfrak{p}]} \subset F$ of binary words excluding a given pattern $\mathfrak{p} = p_0 \dots p_{h-1} \in \{0,1\}^h$, i.e. the words $\omega \in F^{[\mathfrak{p}]}$ that do not admit a sequence of consecutive indices $i, i+1, \dots, i+h-1$ such that $\omega_i \omega_{i+1} \dots \omega_{i+h-1} = p_0 p_1 \dots p_{h-1}$.

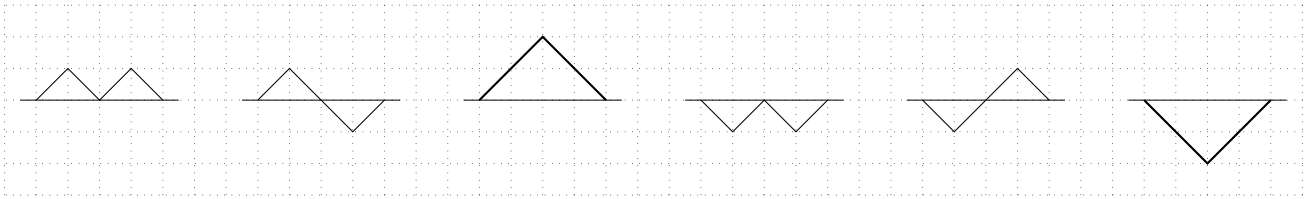
- R. Sedgewick & P. Flajolet (1996): *An Introduction to the Analysis of Algorithms*. Addison-Wesley, Reading, MA.

Example



Grand Dyck words

Let us denote by $\mathcal{G}_j$ the set of $2j$-length Grand Dyck words. It is well known that a Grand Dyck word δ is a binary string such that $|\delta|_0 = |\delta|_1$.



In this work, we consider the so called *bifix-free* Grand Dyck words, i.e. a Grand Dyck word δ is said to be *bifix-free* if and only if no prefix of δ is a suffix of δ.

Let us denote by $\mathcal{G}_j^b$ the set of $2j$-length bifix-free Grand Dyck words and we study the class $F^{[\mathfrak{p}]}$ with $\mathfrak{p} \in \mathcal{G}_j^b$, for any fixed $j \geq 1$.

A general approach

Let $F \subset \{0,1\}^*$ be the class of binary words ω such that $|\omega|_0 \leq |\omega|_1$ for any $\omega \in F$.

For example, in order to generate the language $F^{[\mathfrak{p}]}$ an "ad hoc" grammar (depending on the forbidden pattern $\mathfrak{p}$) should be defined . Consequently, for each pattern $\mathfrak{p}$ a different generating function enumerating the words in $F^{[\mathfrak{p}]}$ must to be computed.

Our aim is to determine a constructive algorithm proposing a more unified approach. The algorithm bases on a succession rule which allows us to obtain a generating function for the enumeration of each classes $F^{[\mathfrak{p}]}$, according to the number of ones. Such enumeration does not depend on the shape of the forbidden pattern $\mathfrak{p}$.

Succession rules

A *succession rule* Ω is a system constituted by an *axiom* (a), with $a \in \mathbb{N}$, and a set of *productions* of the form:

$$(k) \rightsquigarrow (e_1(k))(e_2(k)) \dots (e_k(k)), \qquad k \in \mathbb{N},\ e_i : \mathbb{N} \to \mathbb{N}.$$

A production constructs, for any given label (k), its *successors* $(e_1(k)), (e_2(k)), \dots, (e_k(k))$.

Compact notation:

$$\begin{cases} (a) & \\ (k) & \rightsquigarrow (e_1(k))(e_2(k)) \dots (e_k(k)) \end{cases}$$

- F. R. K. Chung, R. L. Graham, V. E. Hoggatt & M. Kleimann (1978): *The number of Baxter permutations*. Journal of Combinatorial Theory, Series A 24, pp. 382-394.

Generating trees

The rule Ω can be represented by means of a *generating tree*, that is a rooted tree whose vertices are the labels of Ω; where (a) is the label of the root and each node labelled (k) has k sons labelled $(e_1(k)), (e_2(k)), \ldots, (e_k(k))$, respectively. As usual, the root lies at level 0, and a node lies at level n if its parent lies at level $n-1$.

A succession rule describes the growth of a class of combinatorial objects if there exists a bijection between the object of size n and the nodes at level n in the generating tree, then a given object can be coded by the sequence of labels met from the root of the generating tree to the object itself.

Jumping succession rules

A *jumping succession rule* is a set of productions acting on the objects of a class and producing sons at different levels.

Compact notation:

$$\begin{cases} (a) & \\ (k) & \overset{1}{\rightsquigarrow} (e_1(k))(e_2(k))\dots(e_k(k)), \\ (k) & \overset{j}{\rightsquigarrow} (d_1(k))(d_2(k))\dots(d_k(k)). \end{cases}$$

- L. Ferrari, E. Pergola, R. Pinzani & S. Rinaldi (2003): *Jumping succession rules and their generating functions*. Discrete Mathematics 271, pp. 29-50.

Jumping and marked succession rules

A *jumping and marked succession rule* is a jumping succession rule where *marked* labels are considered together with usual ones. In this way a generating tree can support negative values if we consider a node labelled $(\overline{k})$ as opposed to a node labelled (k) lying on the same level.

Compact notation:

$$\begin{cases} (a) & \\ (k) & \overset{1}{\rightsquigarrow} (e_1(k))(e_2(k))\dots(e_k(k)), \\ (k) & \overset{j}{\rightsquigarrow} (\overline{d_1(k)})(\overline{d_2(k)})\dots(\overline{d_k(k)}). \end{cases}$$

Jumping and marked succession rules

Compact notation:

$$\begin{cases} (a) & \\ (k) & \overset{1}{\rightsquigarrow} (e_1(k))(e_2(k))\dots(e_k(k)), \\ (k) & \overset{j}{\rightsquigarrow} (\overline{d_1(k)})(\overline{d_2(k)})\dots(\overline{d_k(k)}). \end{cases}$$

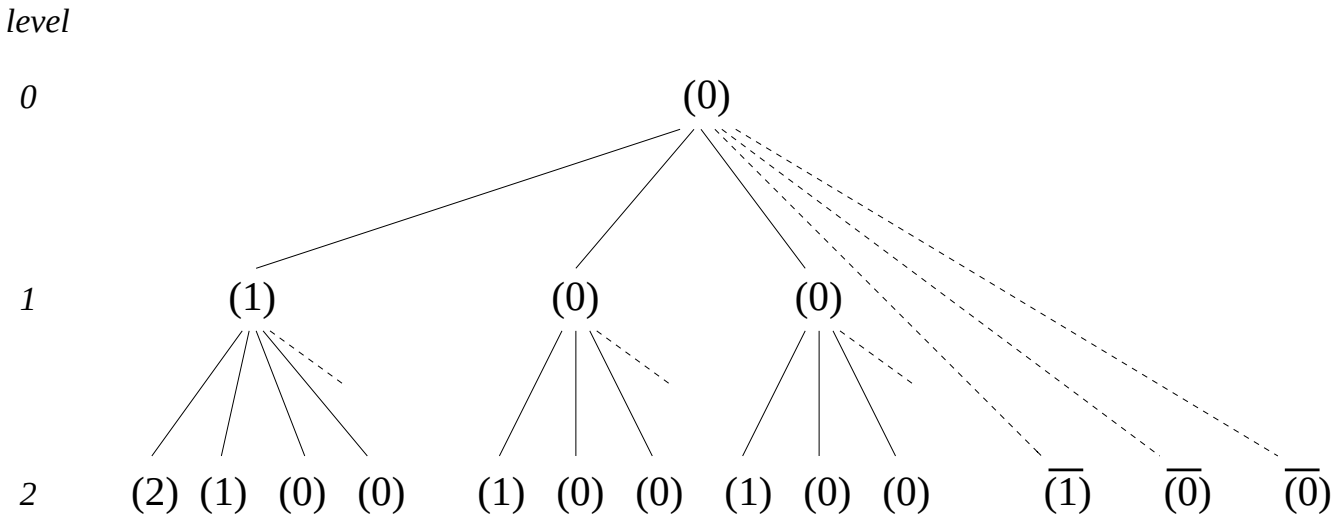
describes also the behavior for $(\overline{k})$:

$$\begin{cases} (\overline{k}) & \overset{1}{\rightsquigarrow} (\overline{e_1(k)})(\overline{e_2(k)})\dots(\overline{e_k(k)}), \\ (\overline{k}) & \overset{j}{\rightsquigarrow} (d_1(k))(d_2(k))\dots(d_k(k)). \end{cases}$$

having $(\overline{\overline{k}}) = (k)$.

Example

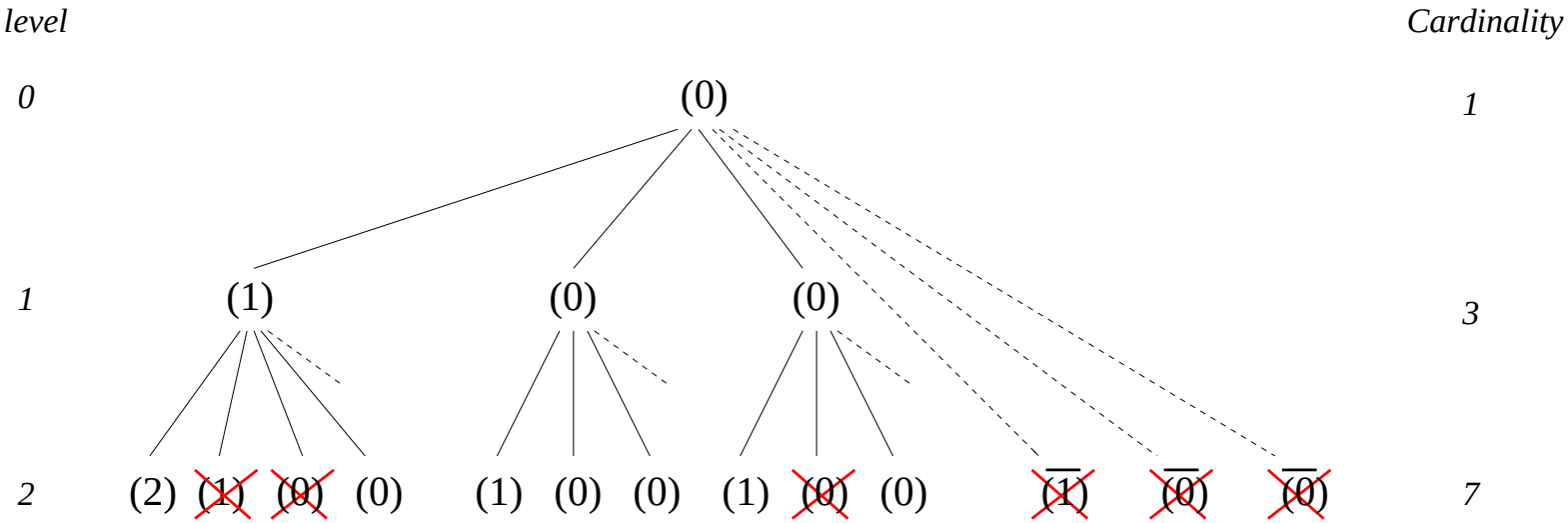
$$\begin{cases} (0) & \\ (k) & \overset{1}{\rightsquigarrow} (k+1)(k)\cdots(1)(0)(0) \\ (k) & \overset{2}{\rightsquigarrow} (\overline{k+1})(\overline{k})\cdots(\overline{1})(\overline{0})(\overline{0}) \end{cases}$$



$(\overline{\overline{k}}) = (k)$

Example

$$\begin{cases} (0) & \\ (k) & \overset{1}{\rightsquigarrow} (k+1)(k)\cdots(1)(0)(0) \\ (k) & \overset{2}{\rightsquigarrow} (\overline{k+1})(\overline{k})\cdots(\overline{1})(\overline{0})(\overline{0}) \end{cases}$$



$$(\overline{\overline{k}}) = (k)$$

Theorem

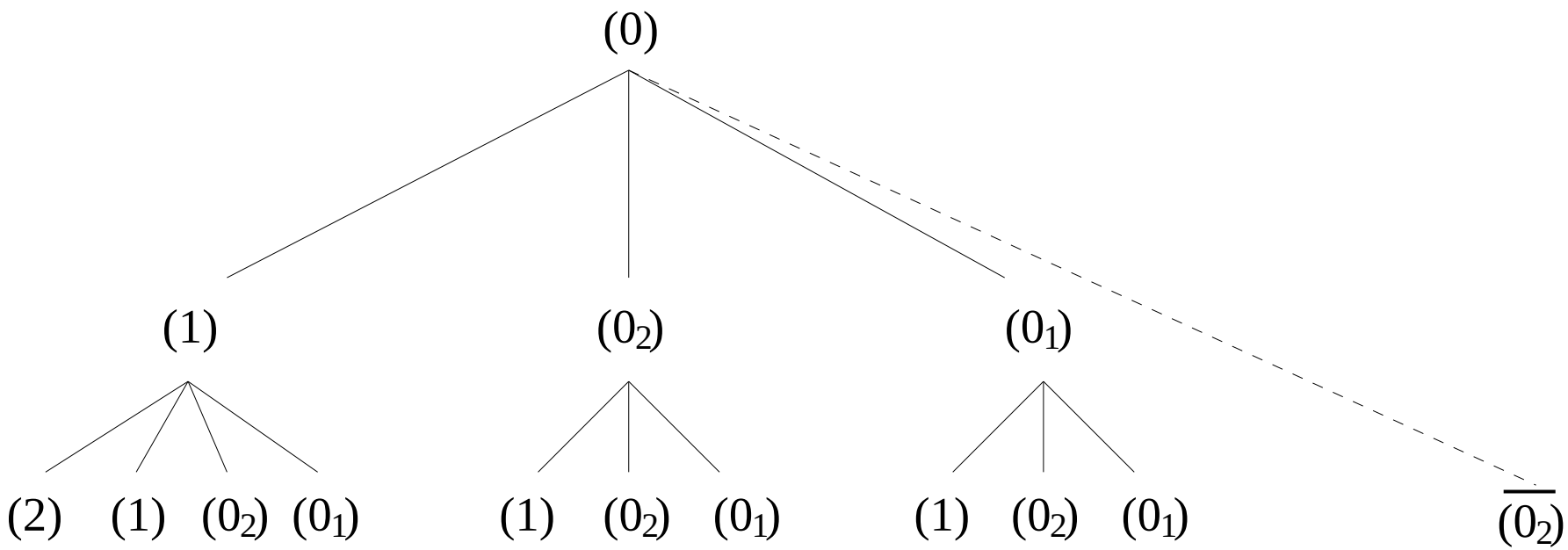
Theorem

The generating tree of the paths in $F^{[\mathfrak{p}]}$, where $\mathfrak{p} \in \mathcal{G}_j^b$, according to the number of rise steps, is isomorphic to the tree having its root labelled (0) and recursively defined by the succession rule:

$$\begin{cases} (k) \overset{1}{\rightsquigarrow} (k+1)(k)\dots(1)(0_2)(0_1) & k \geq 0 \\ (0) \overset{j}{\rightsquigarrow} (\overline{0_2}) & \\ (k) \overset{j}{\rightsquigarrow} (\overline{k})(\overline{k-1})\dots(\overline{1})(\overline{0_2})(\overline{0_1}) & k \geq 1 \end{cases}$$

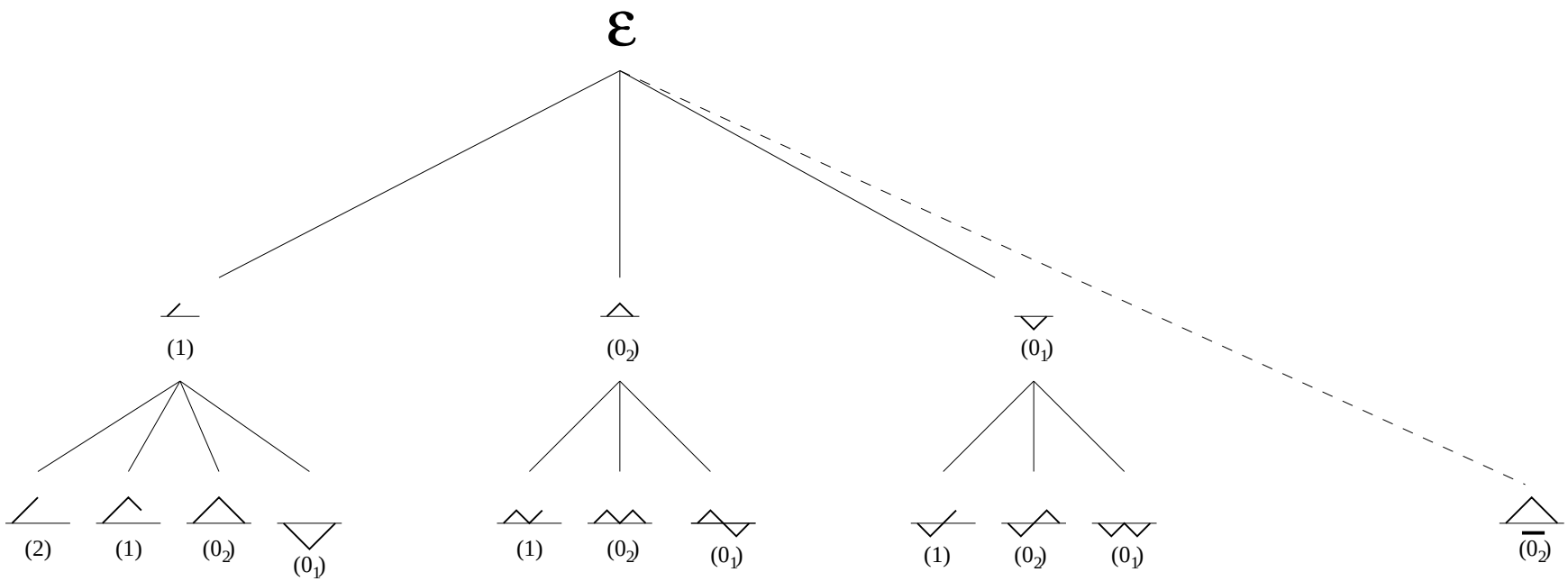
Method

Each node at level n and labelled with (k), $k \geq 0$, represents a path on the discrete plane having n rise steps and ending at the ordinate k.



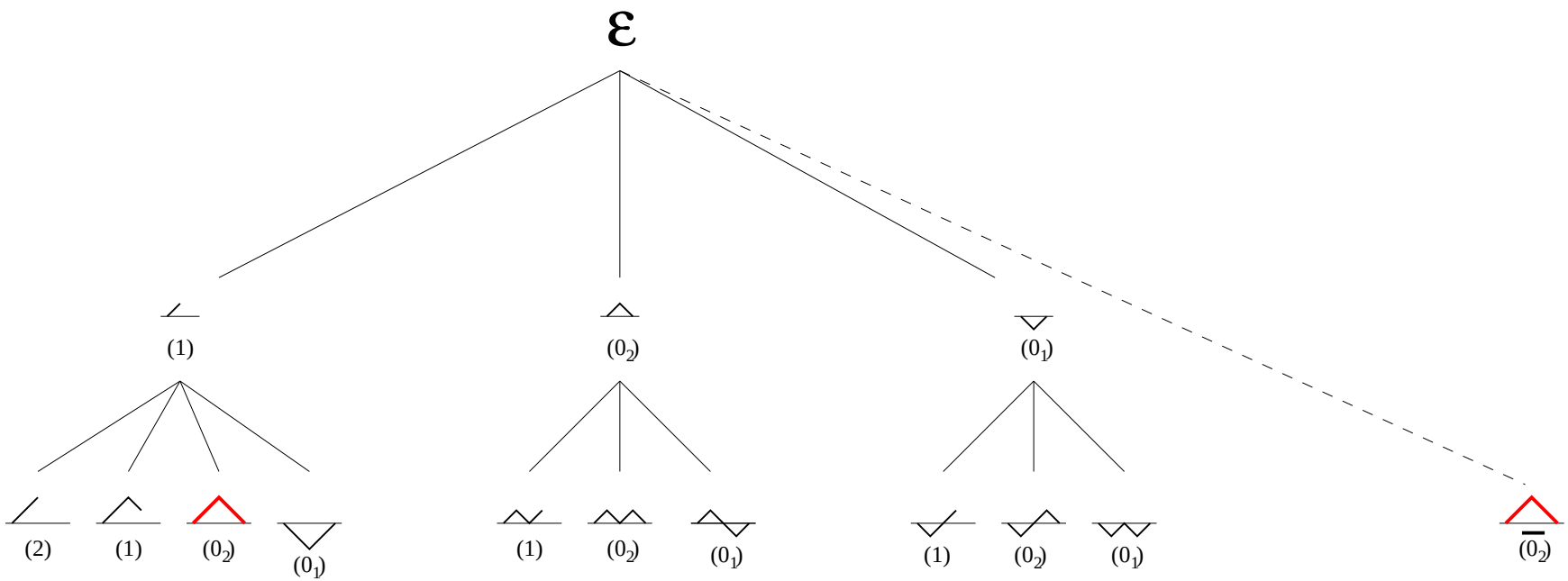
Method

Each node at level n and labelled with (k), $k \geq 0$, represents a path on the discrete plane having n rise steps and ending at the ordinate k.



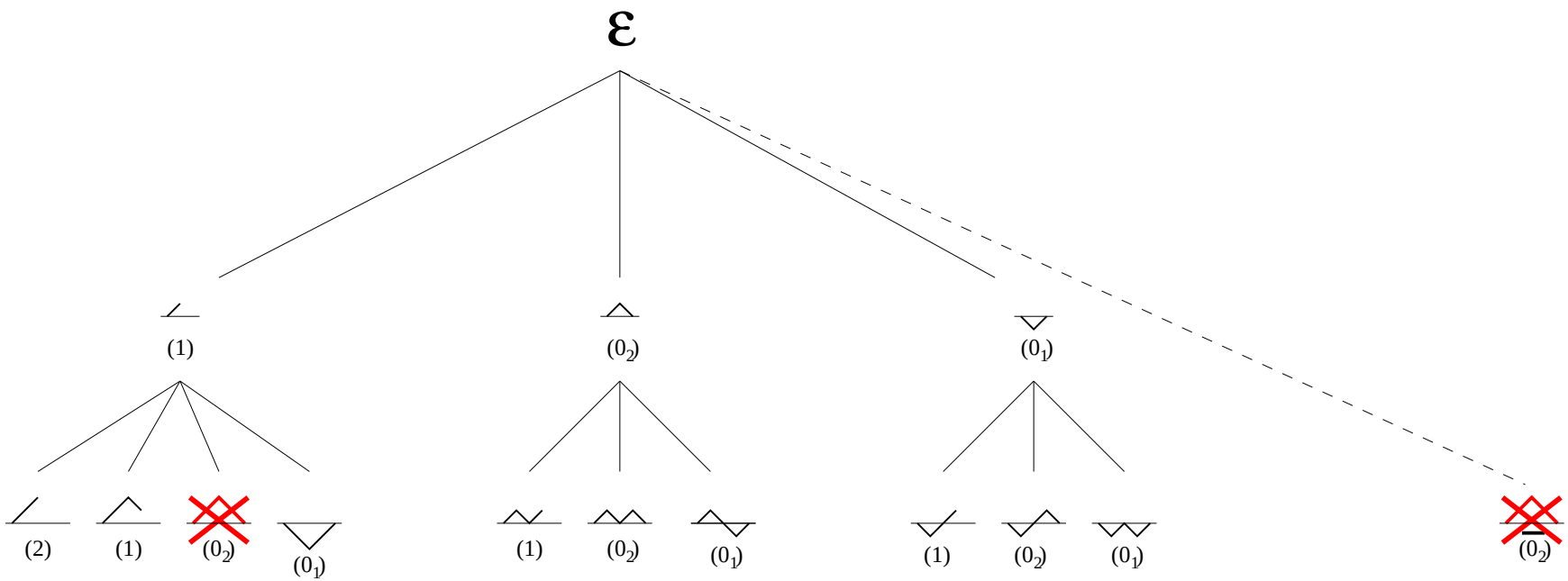
Method

Each node at level n and labelled with (k), $k \geq 0$, represents a path on the discrete plane having n rise steps and ending at the ordinate k.



Method

Each node at level n and labelled with (k), $k \geq 0$, represents a path on the discrete plane having n rise steps and ending at the ordinate k.

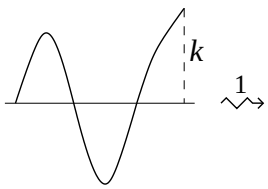


The constructive algorithm: First step

$$(k) \overset{1}{\rightsquigarrow} (k+1)\,(k)\,\ldots\,(1)\,(0_2)\,(0_1)$$

The constructive algorithm: First step

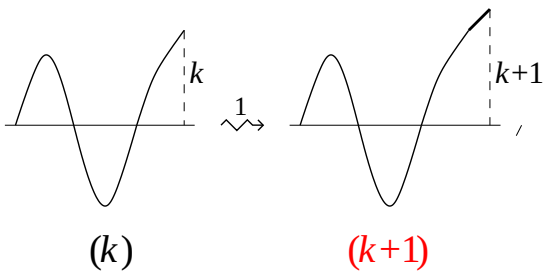
$$(k) \overset{1}{\rightsquigarrow} (k+1)\,(k)\,\ldots\,(1)\,(0_2)\,(0_1)$$



(k)

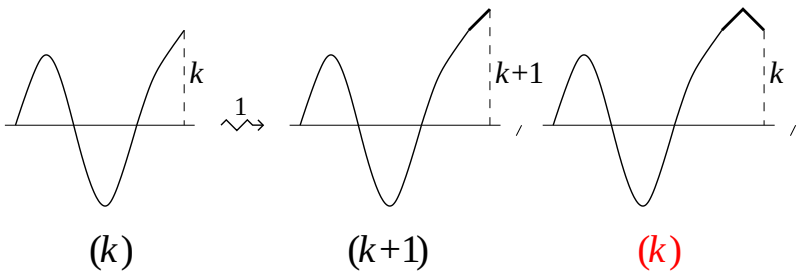
The constructive algorithm: First step

$$(k) \overset{1}{\rightsquigarrow} (k+1)\,(k)\,\ldots\,(1)\,(0_2)\,(0_1)$$



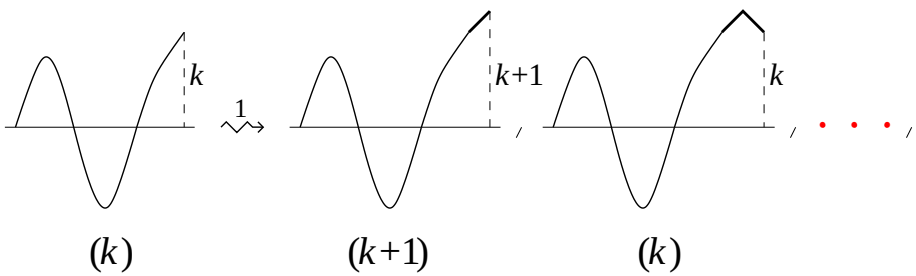
The constructive algorithm: First step

$$(k) \overset{1}{\rightsquigarrow} (k+1)\,(k) \ldots (1)\,(0_2)\,(0_1)$$



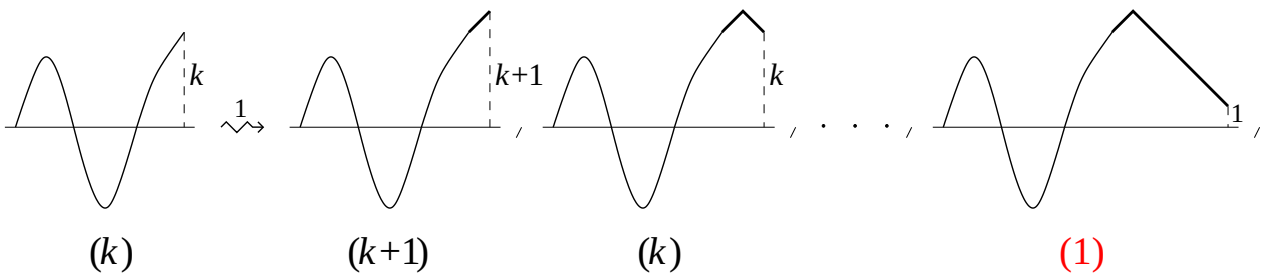
The constructive algorithm: First step

$$(k) \overset{1}{\rightsquigarrow} (k+1)\,(k)\ldots(1)\,(0_2)\,(0_1)$$



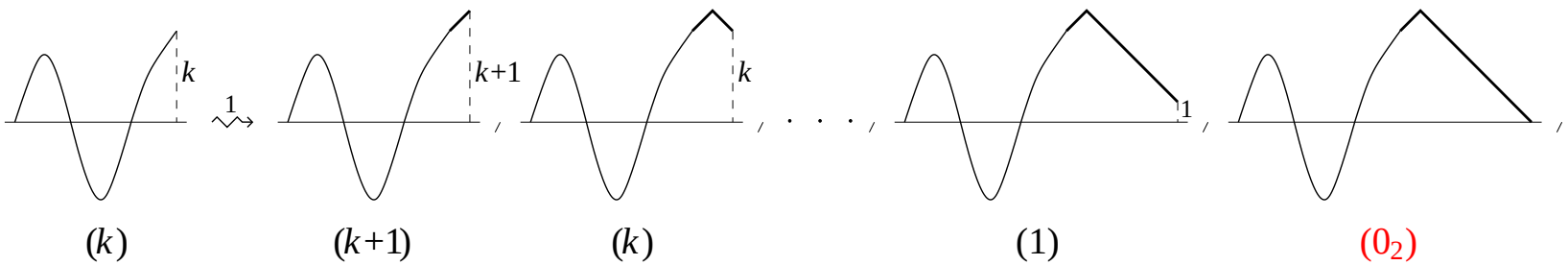
The constructive algorithm: First step

$$(k) \overset{1}{\rightsquigarrow} (k+1)\,(k)\,\ldots\,(1)\,(0_2)\,(0_1)$$



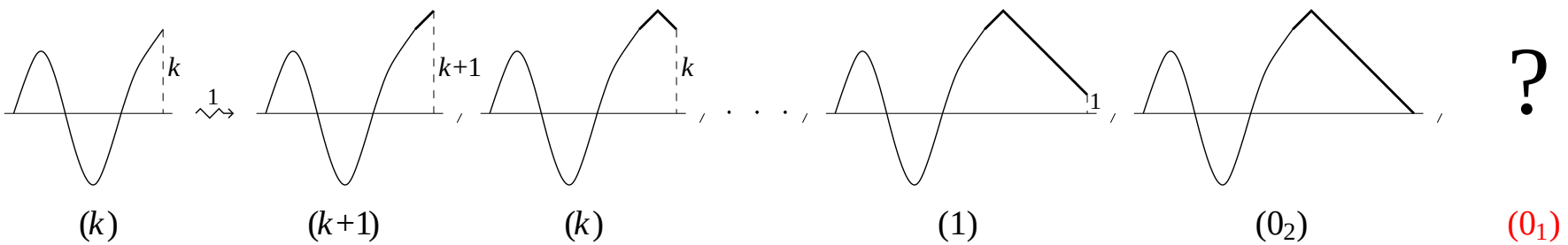
The constructive algorithm: First step

$$(k) \overset{1}{\rightsquigarrow} (k+1)\,(k)\,\ldots\,(1)\,(0_2)\,(0_1)$$



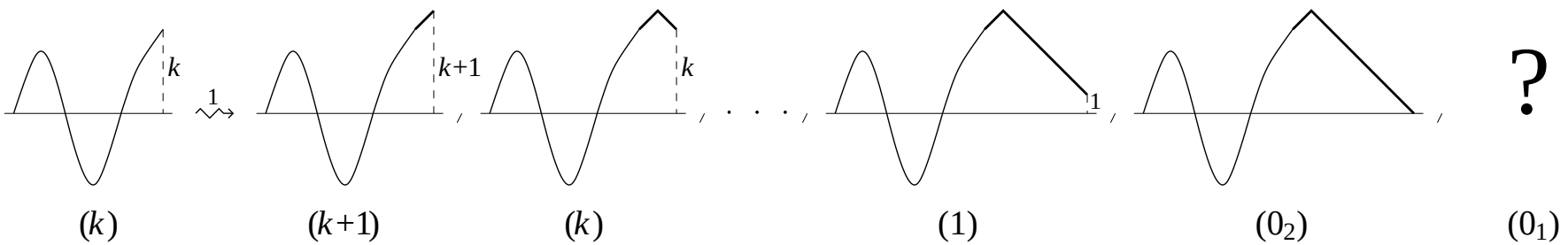
The constructive algorithm: First step

$$(k) \overset{1}{\rightsquigarrow} (k+1)\,(k)\,\ldots\,(1)\,(0_2)\,(0_1)$$



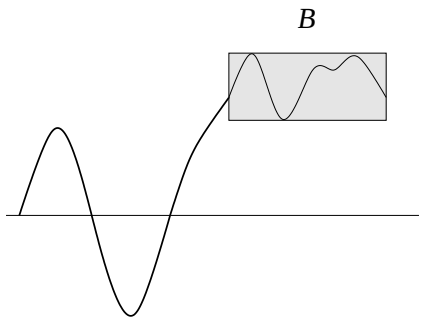
The constructive algorithm: First step

$$(k) \overset{1}{\rightsquigarrow} (k+1)\,(k) \ldots (1)\,(0_2)\,(0_1)$$



The constructive algorithm: a marked forbidden pattern

We define a *marked forbidden pattern* $\mathfrak{p}$ as a pattern $\mathfrak{p} \in \mathcal{G}_j^b$ whose steps cannot be split, that is, they must always be contained all together in that defined sequence. We say that a point is strictly contained in a given marked forbidden pattern $\mathfrak{p}$ if it is in $\mathfrak{p}$ and it is different from both its initial point and its last point. We denote a marked forbidden pattern $\mathfrak{p}$ by drawing its minimal bounding rectangle B.

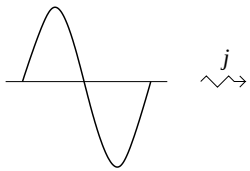


The constructive algorithm: First step

$$(0) \overset{j}{\rightsquigarrow} \overline{(0_2)}$$

The constructive algorithm: First step

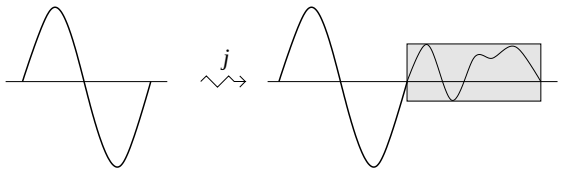
$$(0) \overset{j}{\rightsquigarrow} \overline{(0_2)}$$



(0)

The constructive algorithm: First step

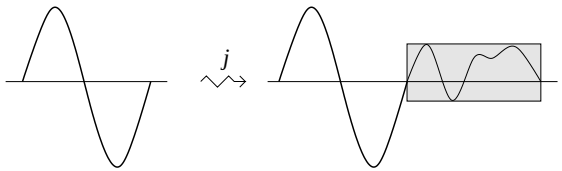
$$(0) \overset{j}{\rightsquigarrow} \overline{(0_2)}$$



(0) $\qquad\qquad$ $\overline{(0_2)}$

The constructive algorithm: First step

$$(0) \overset{j}{\rightsquigarrow} \overline{(0_2)}$$



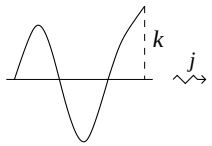
(0) $\qquad$ $\overline{(0_2)}$

The constructive algorithm: First step

$$(k) \overset{j}{\rightsquigarrow} \overline{(k)}\,\overline{(k\text{-}1)} \cdots \overline{(1)}\,\overline{(0_2)}\,\overline{(0_1)}$$

The constructive algorithm: First step

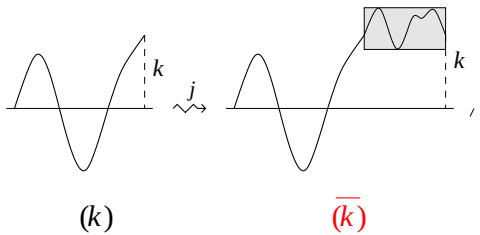
$$(k) \overset{j}{\rightsquigarrow} \overline{(k)}\,\overline{(k\text{-}1)} \cdots \overline{(1)}\,\overline{(0_2)}\,\overline{(0_1)}$$



(k)

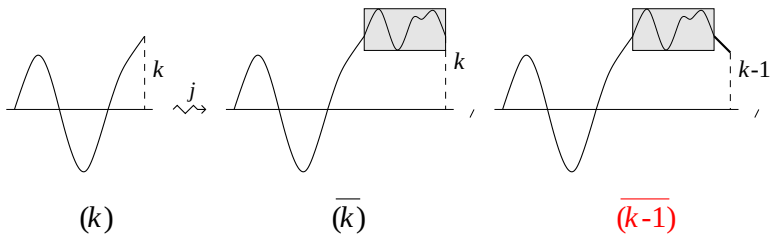
The constructive algorithm: First step

$$(k) \overset{j}{\rightsquigarrow} \overline{(k)}\,\overline{(k\text{-}1)} \cdots \overline{(1)}\,\overline{(0_2)}\,\overline{(0_1)}$$



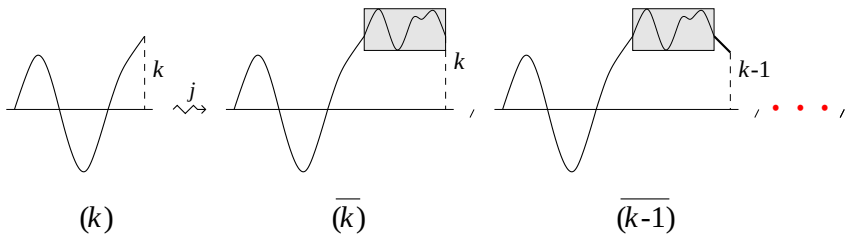
The constructive algorithm: First step

$$(k) \overset{j}{\rightsquigarrow} \overline{(k)}\,\overline{(k\text{-}1)} \cdots \overline{(1)}\,\overline{(0_2)}\,\overline{(0_1)}$$



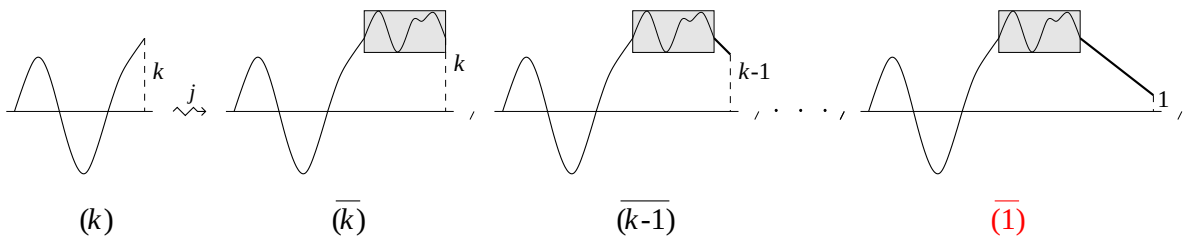
The constructive algorithm: First step

$$(k) \overset{j}{\rightsquigarrow} \overline{(k)}\,\overline{(k\text{-}1)} \cdots \overline{(1)}\,\overline{(0_2)}\,\overline{(0_1)}$$



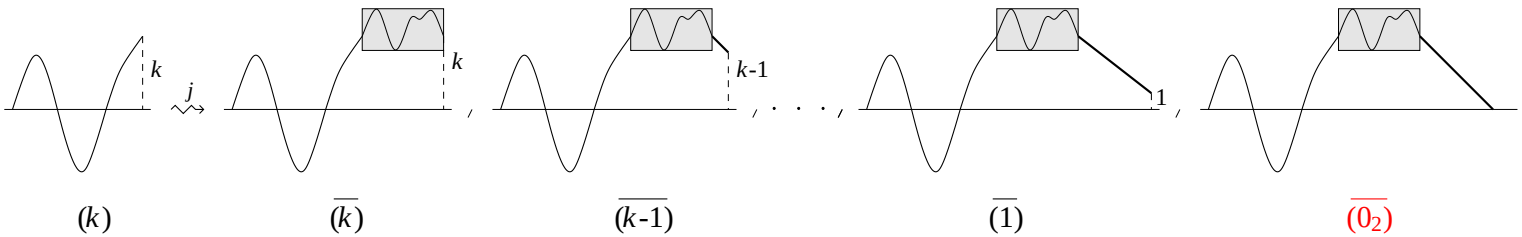
The constructive algorithm: First step

$$(k) \overset{j}{\rightsquigarrow} \overline{(k)}\,\overline{(k\text{-}1)} \cdots \overline{(1)}\,\overline{(0_2)}\,\overline{(0_1)}$$



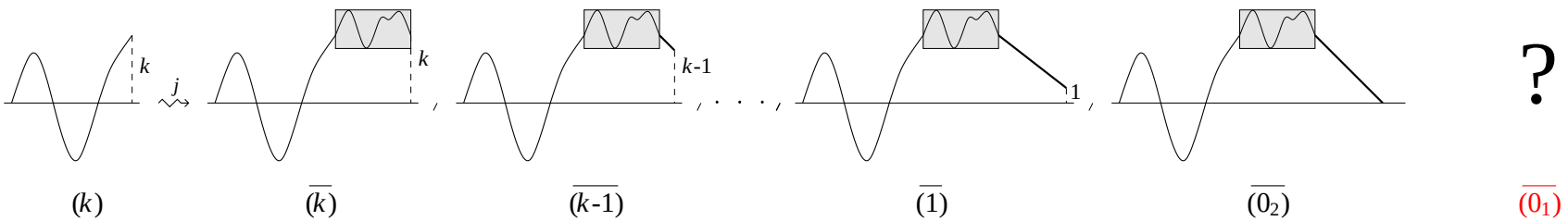
The constructive algorithm: First step

$$(k) \overset{j}{\rightsquigarrow} \overline{(k)}\,\overline{(k\text{-}1)} \cdots \overline{(1)}\,\overline{(0_2)}\,\overline{(0_1)}$$



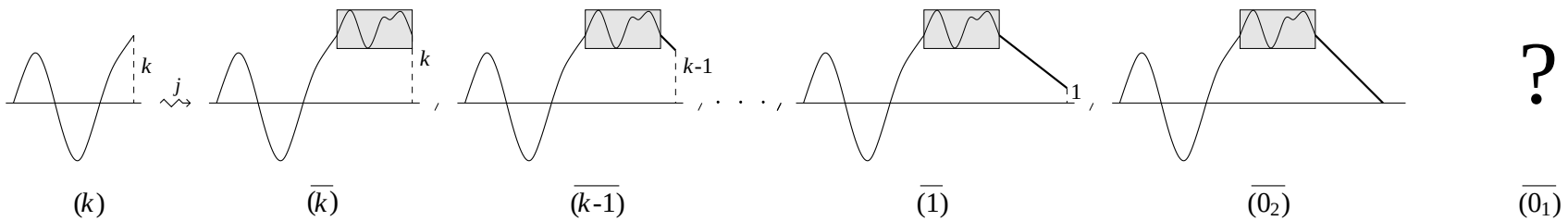
The constructive algorithm: First step

$$(k) \overset{j}{\rightsquigarrow} \overline{(k)}\,\overline{(k\text{-}1)} \cdots \overline{(1)}\,\overline{(0_2)}\,\overline{(0_1)}$$



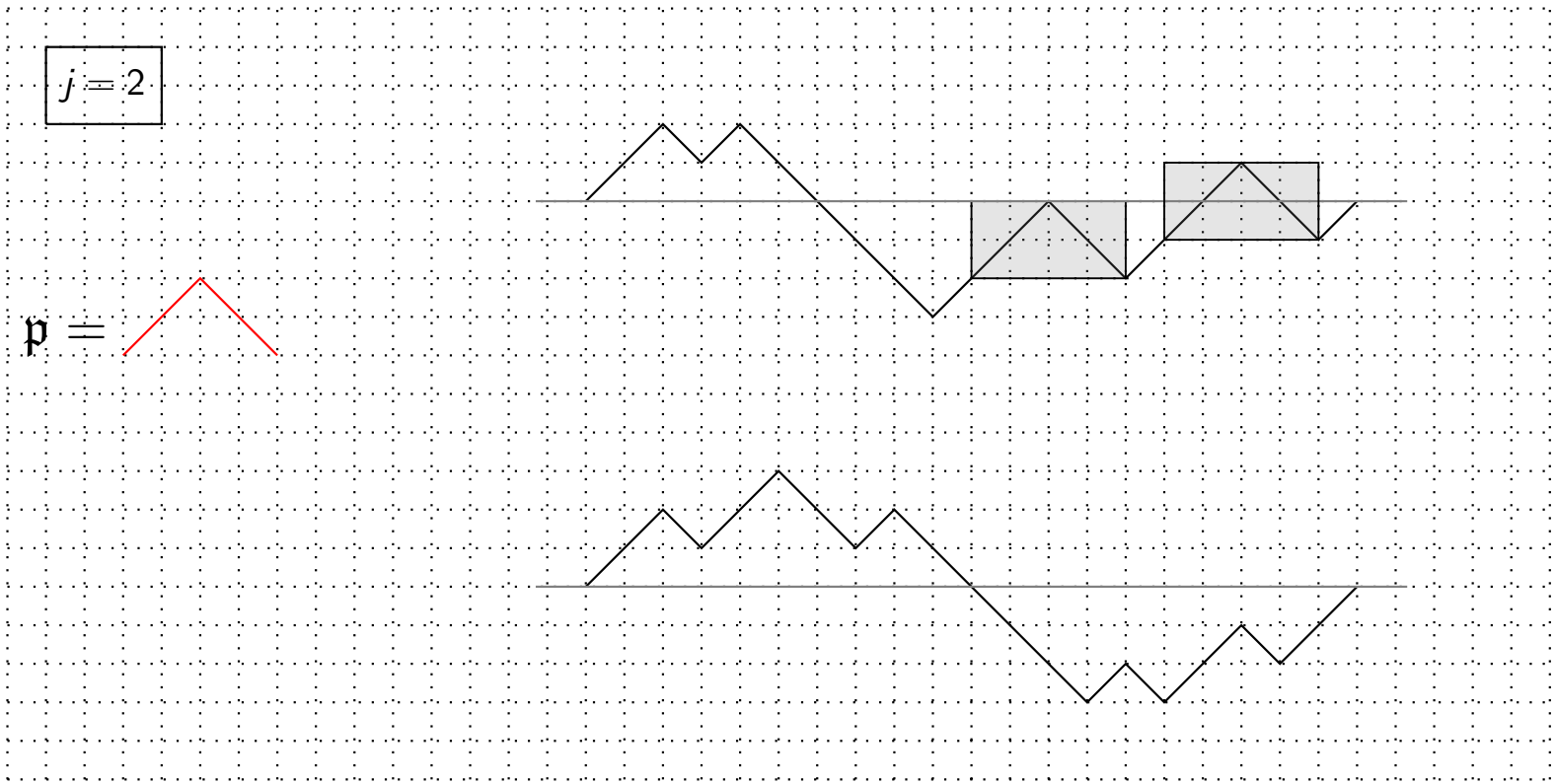
The constructive algorithm: First step

$$(k) \overset{j}{\rightsquigarrow} \overline{(k)}\,\overline{(k\text{-}1)} \cdots \overline{(1)}\,\overline{(0_2)}\,\overline{(0_1)}$$



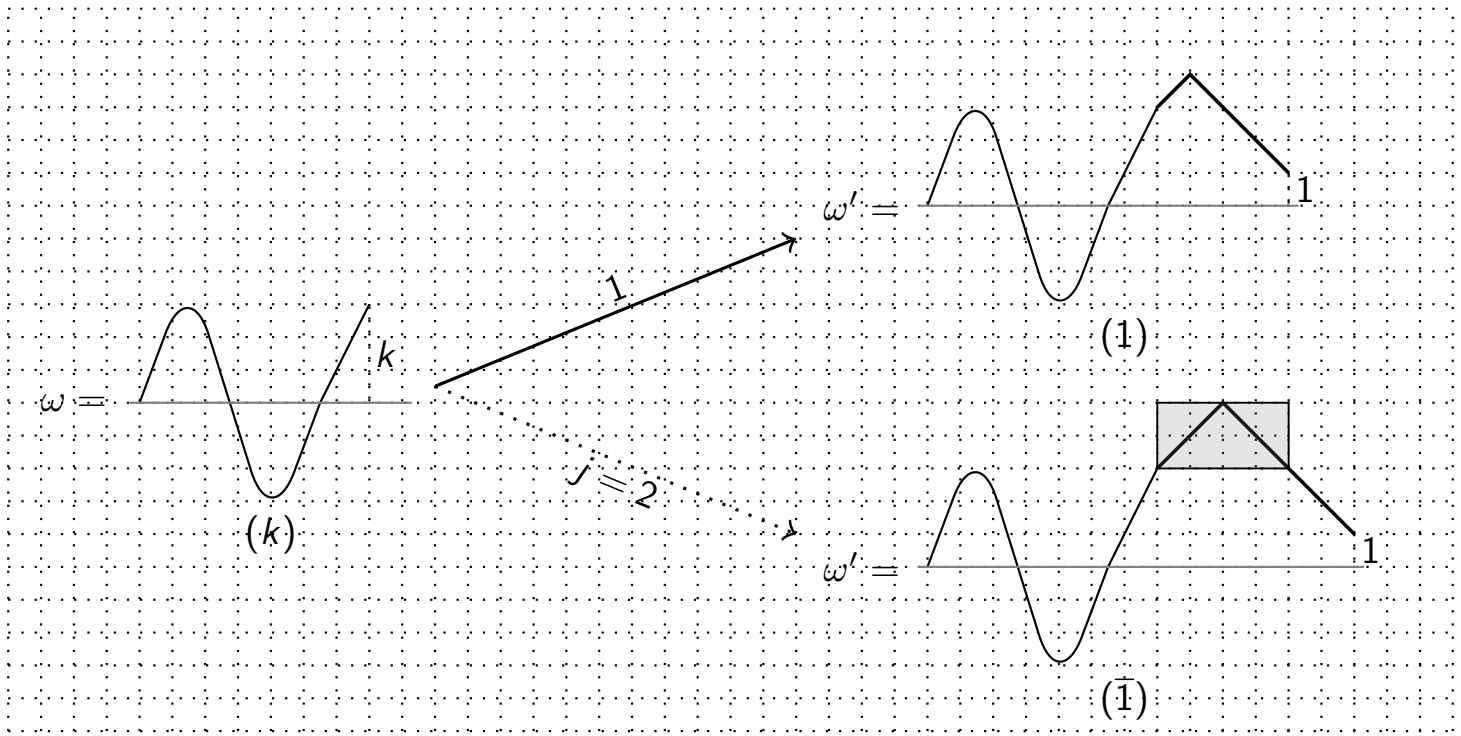
Problem

Lattice paths which end on the x-axis by a rise step are never obtained.



The constructive algorithm: Second step

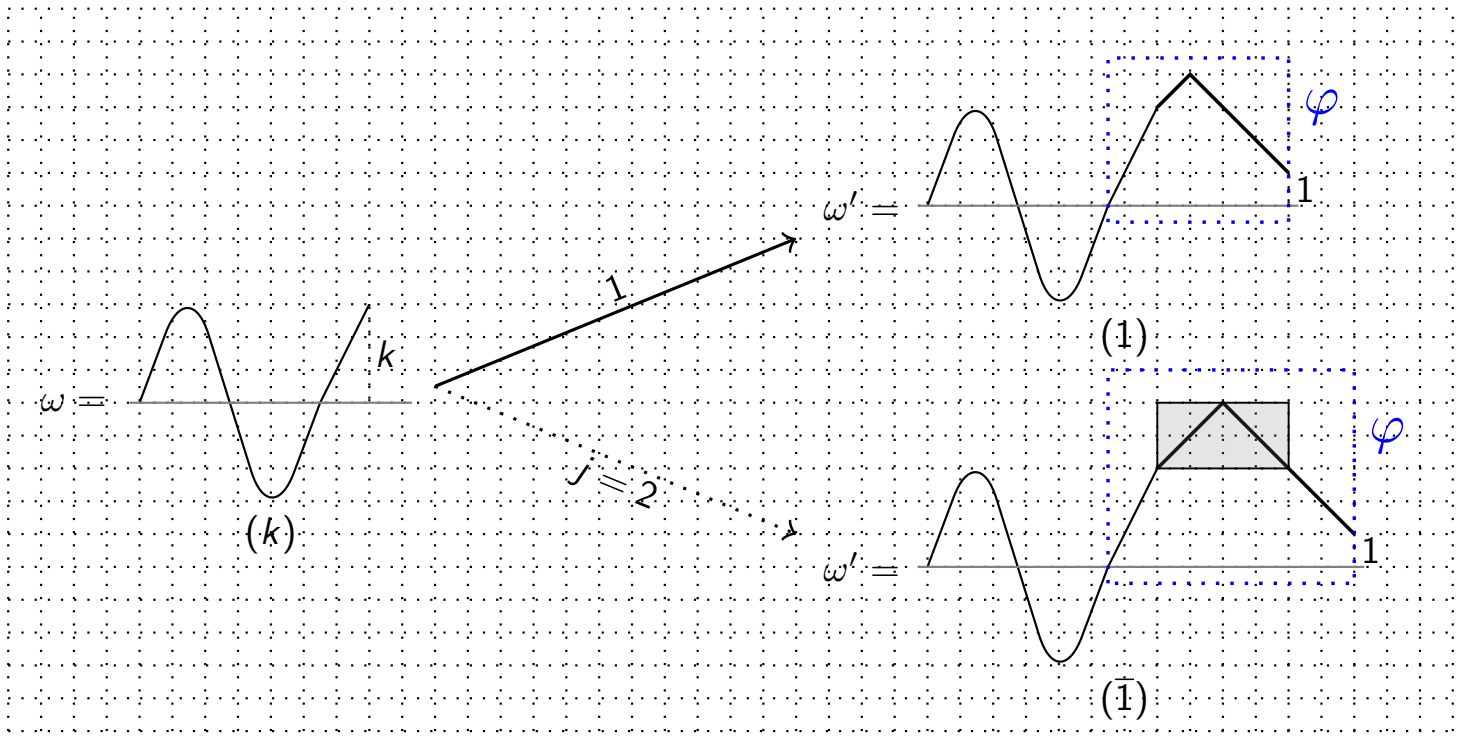
First action in order to obtain the label (0_1)



$\omega' = v\varphi$, being φ the rightmost suffix in ω' beginning from the x-axis with a rise step where each marked forbidden pattern (if φ contains at least one of them) has its initial point with positive ordinate.

The constructive algorithm: Second step

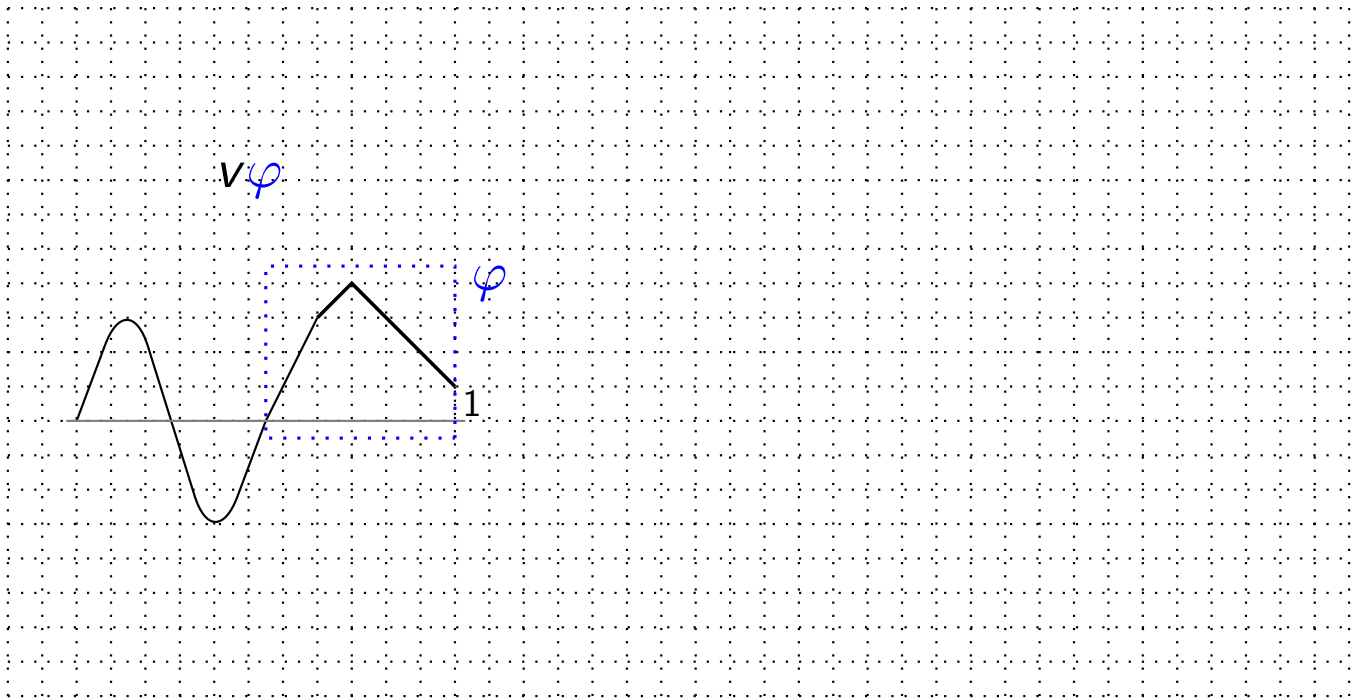
First action in order to obtain the label (0_1)



$\omega' = v\varphi$, being φ the rightmost suffix in ω' beginning from the x-axis with a rise step where each marked forbidden pattern (if φ contains at least one of them) has its initial point with positive ordinate.

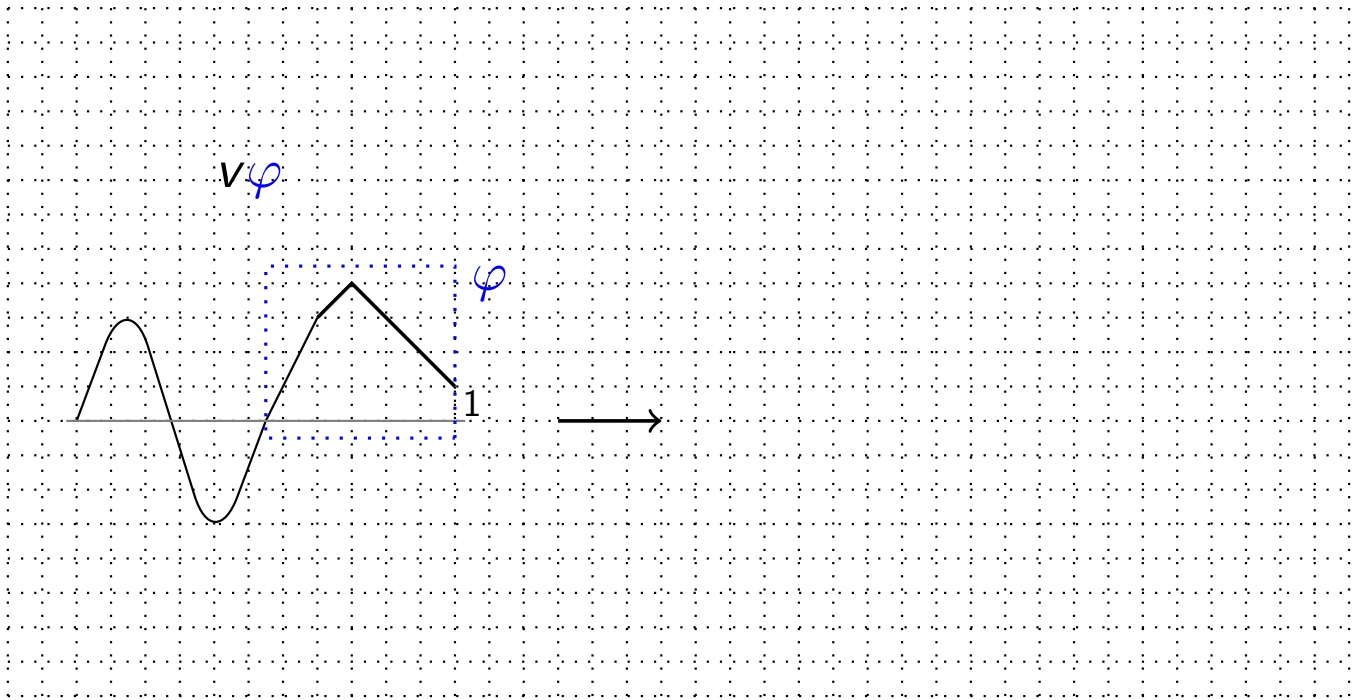
The constructive algorithm: Second step - case 1

φ does not contain any marked forbidden pattern



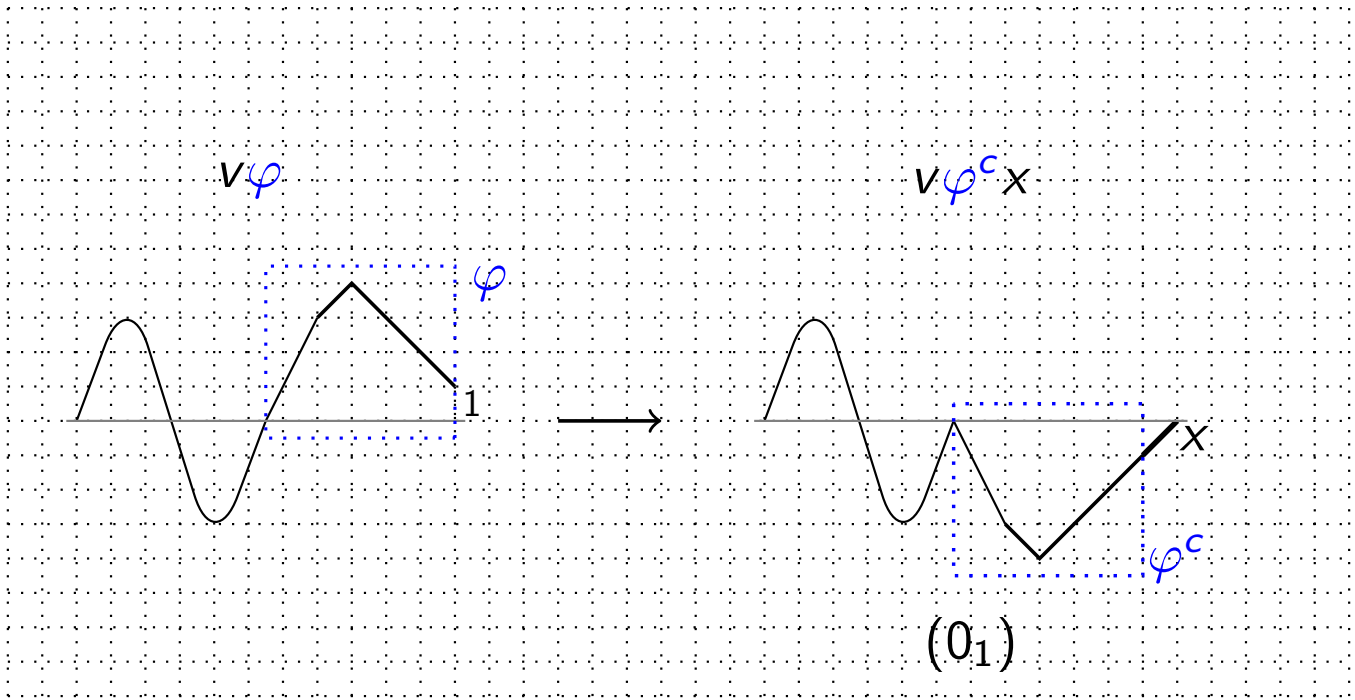
The constructive algorithm: Second step - case 1

φ does not contain any marked forbidden pattern



The constructive algorithm: Second step - case 1

φ does not contain any marked forbidden pattern



The constructive algorithm: Second step - case 2

φ contains at least one marked forbidden pattern

- Find z the leftmost point in φ having highest ordinate and not strictly contained in a marked forbidden pattern.
- Apply the *cut and paste* operation.

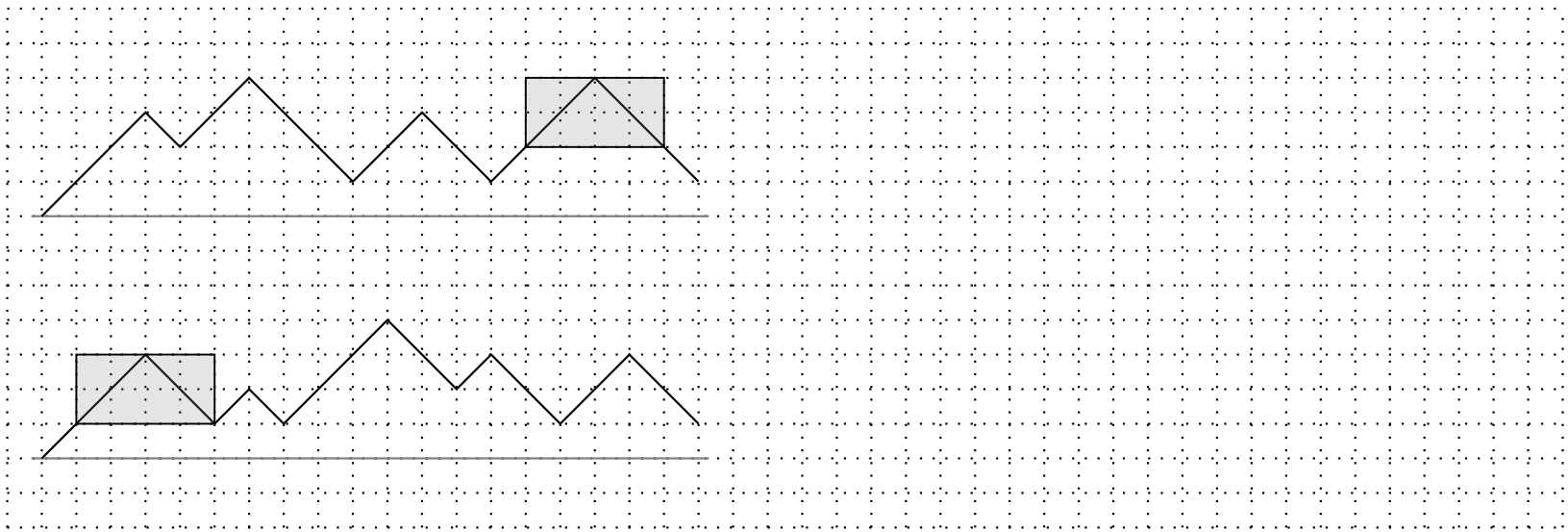
The constructive algorithm: Second step - case 2

φ contains at least one marked forbidden pattern

- Find z the leftmost point in φ having highest ordinate and not strictly contained in a marked forbidden pattern.
- Apply the *cut and paste* operation.

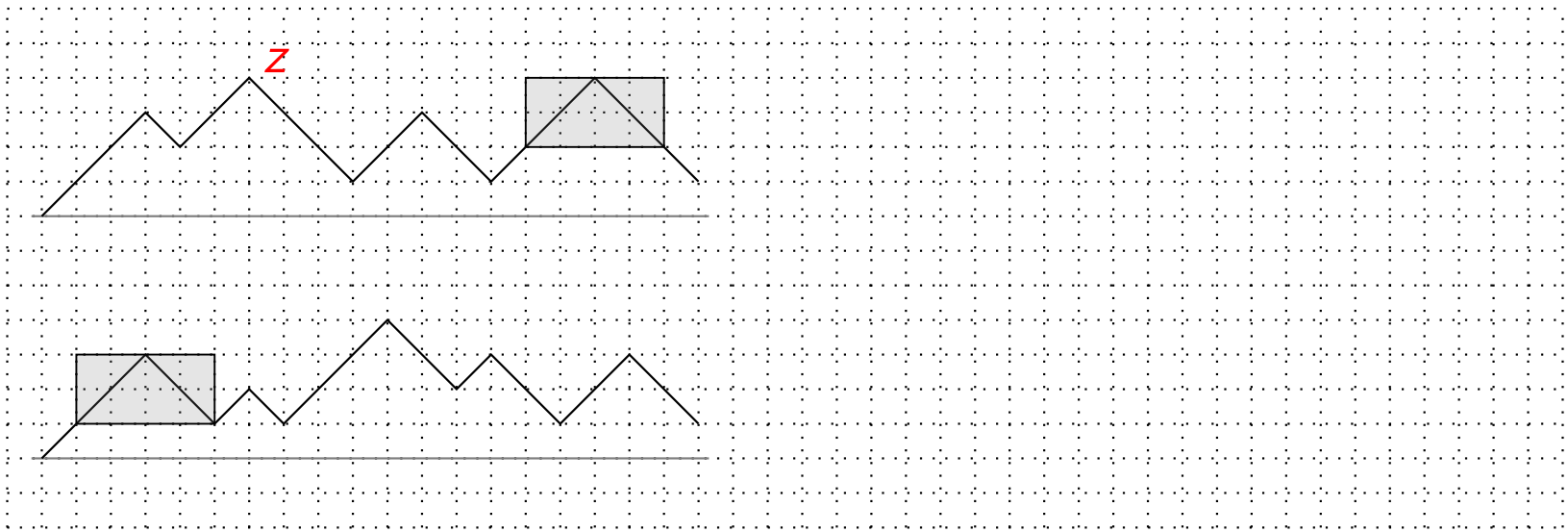
The constructive algorithm: Second step - case 2

φ contains at least one marked forbidden pattern



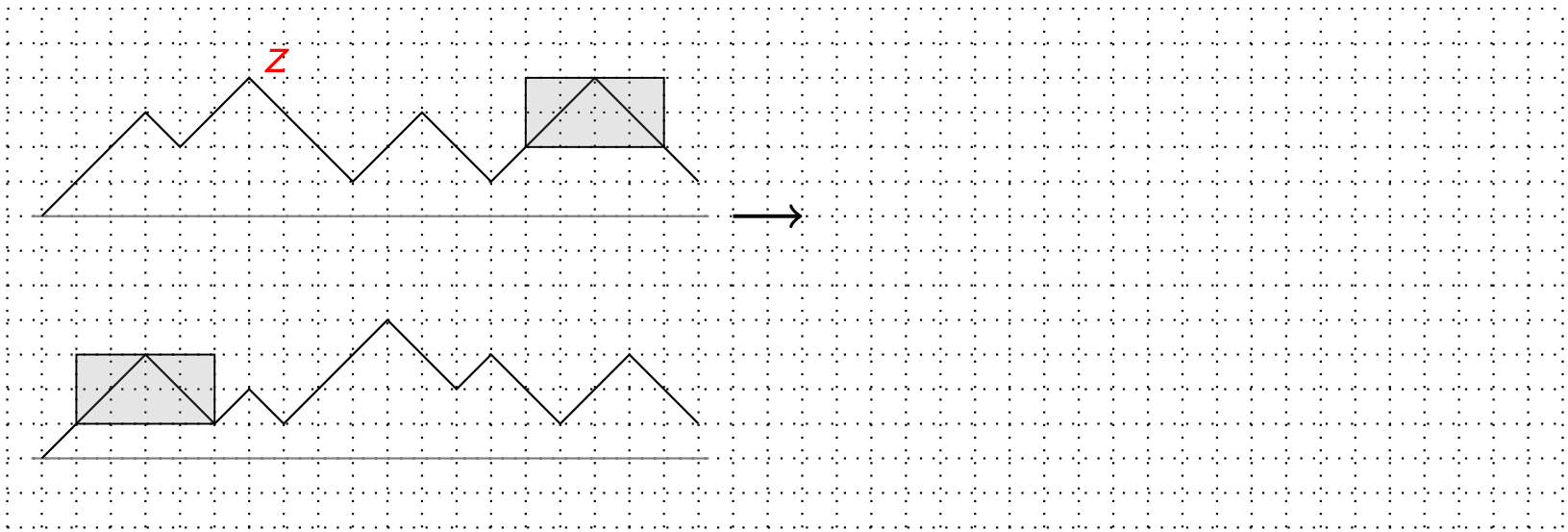
The constructive algorithm: Second step - case 2

φ contains at least one marked forbidden pattern



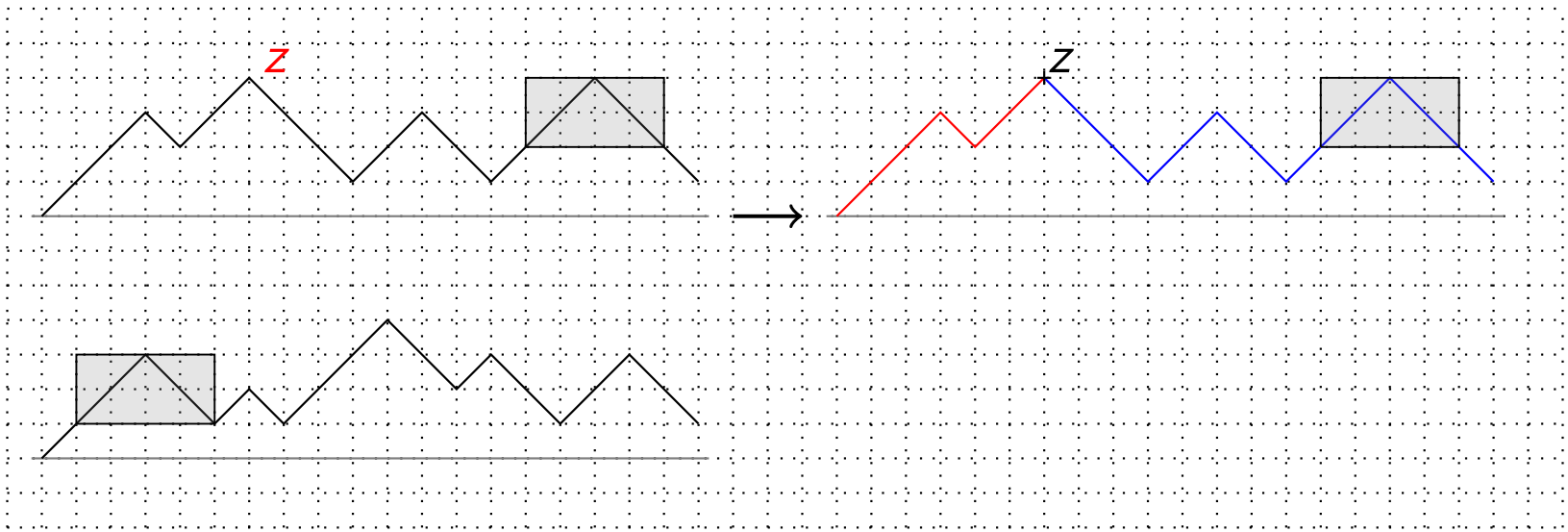
The constructive algorithm: Second step - case 2

φ contains at least one marked forbidden pattern



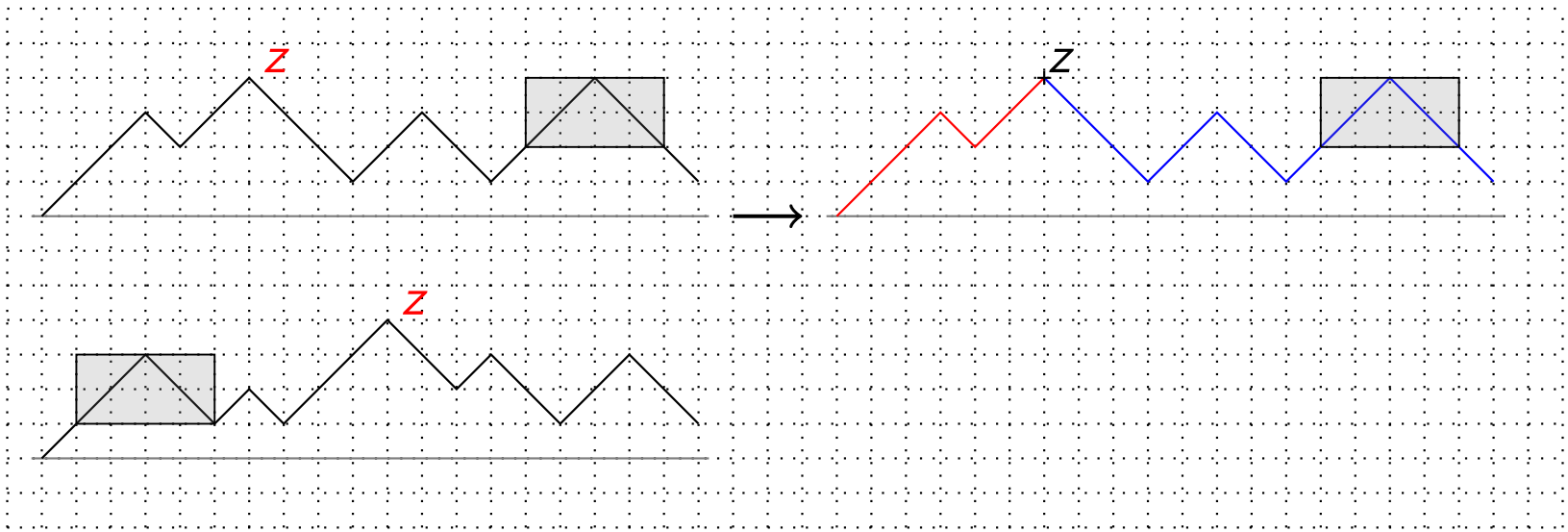
The constructive algorithm: Second step - case 2

φ contains at least one marked forbidden pattern



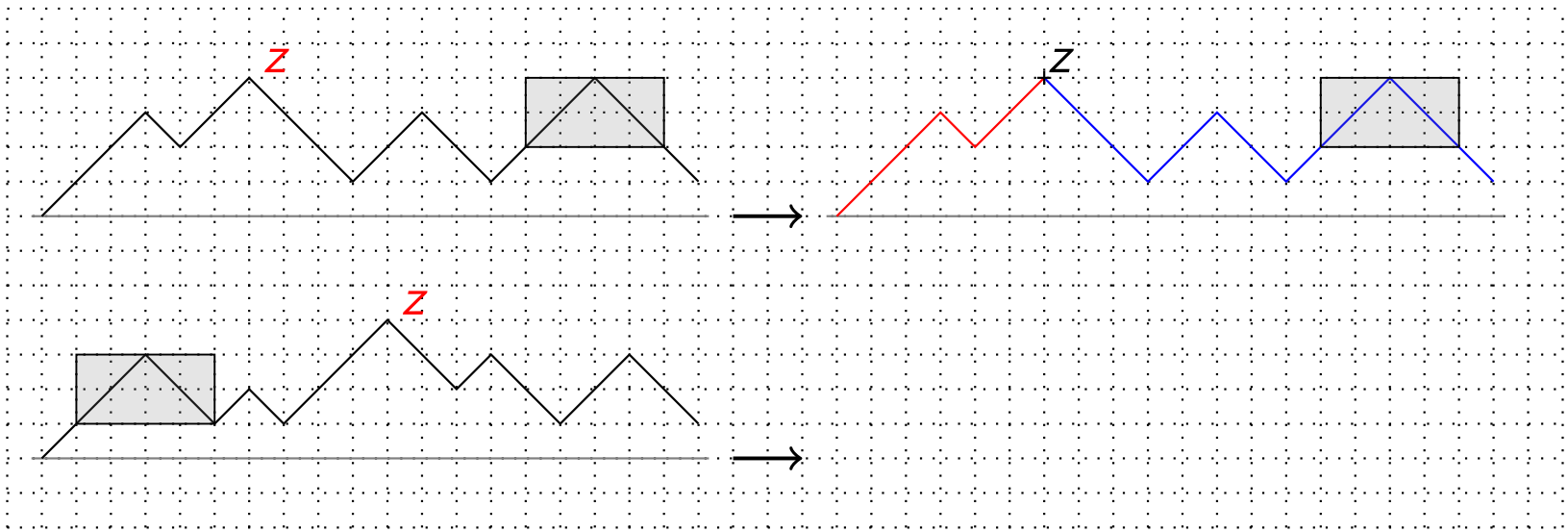
The constructive algorithm: Second step - case 2

φ contains at least one marked forbidden pattern



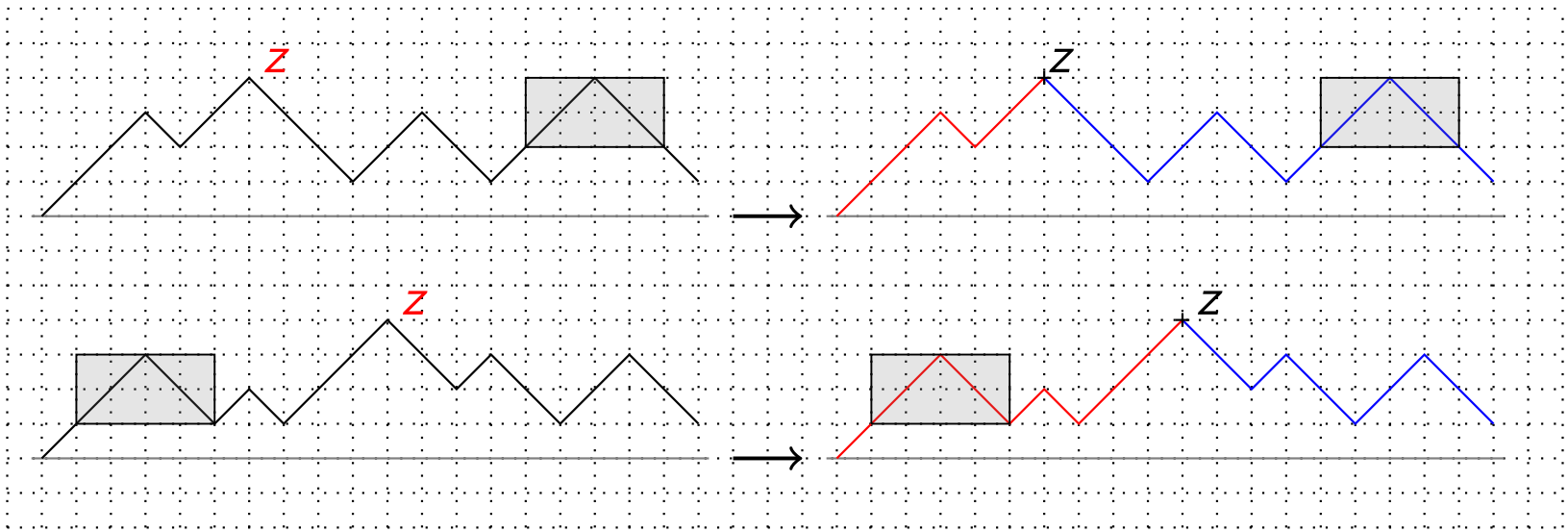
The constructive algorithm: Second step - case 2

φ contains at least one marked forbidden pattern



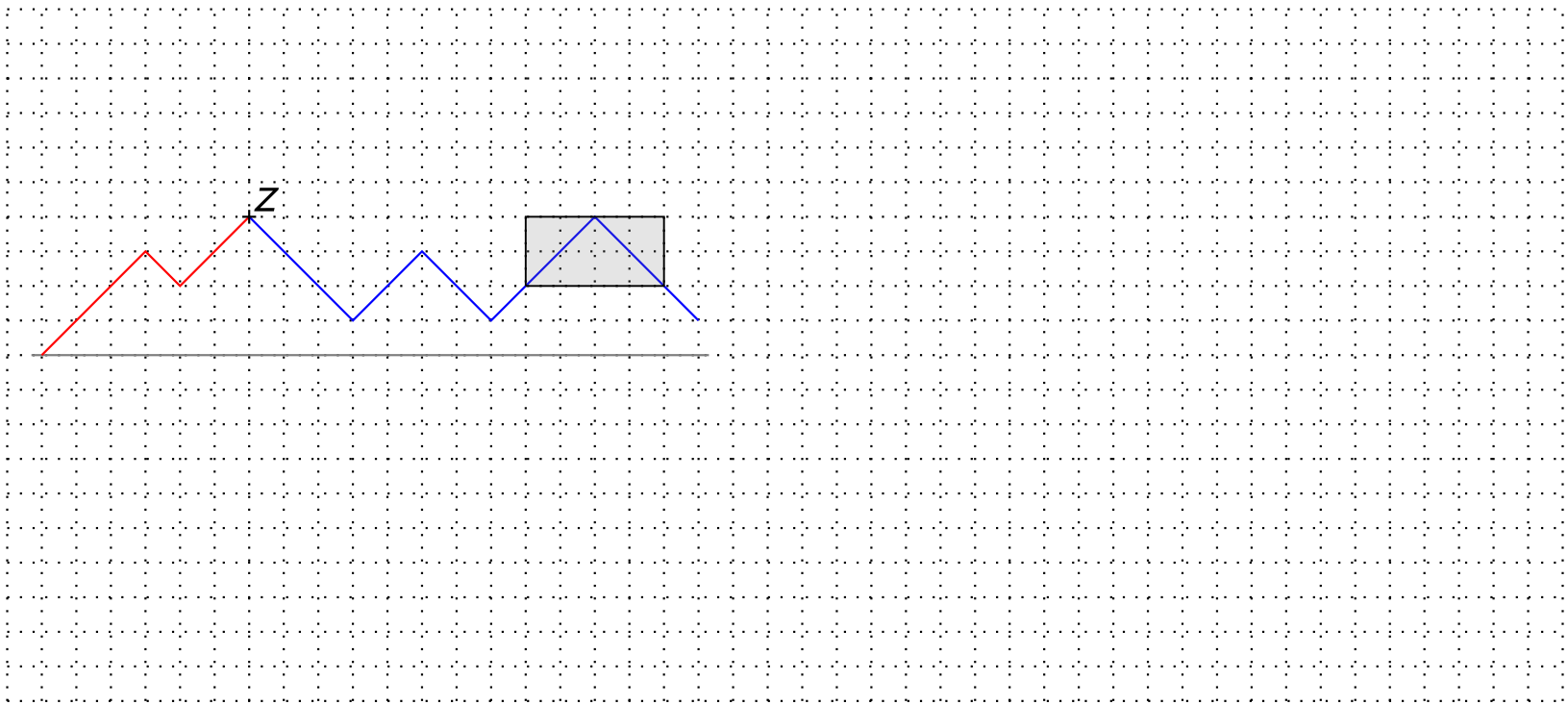
The constructive algorithm: Second step - case 2

φ contains at least one marked forbidden pattern



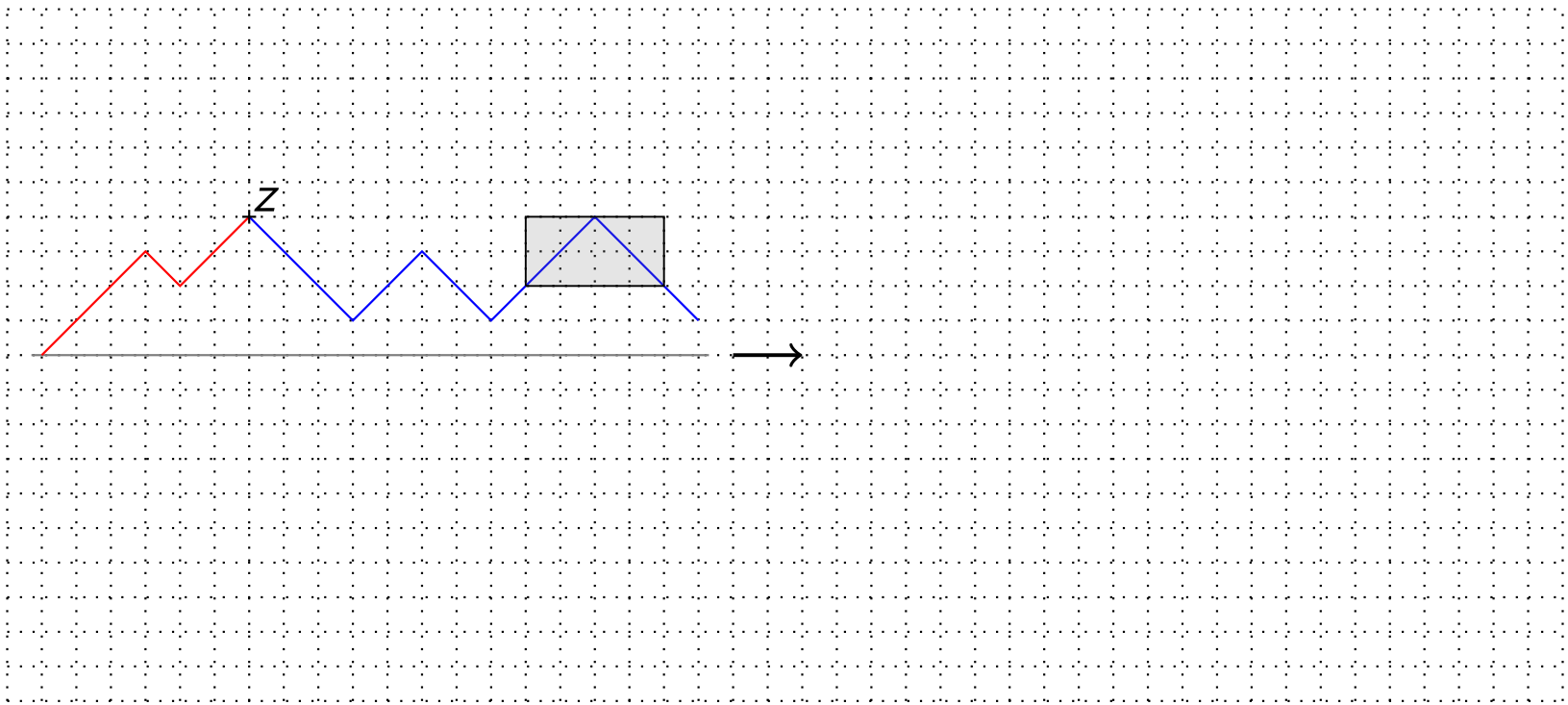
The constructive algorithm: Second step - case 2

Cut and paste



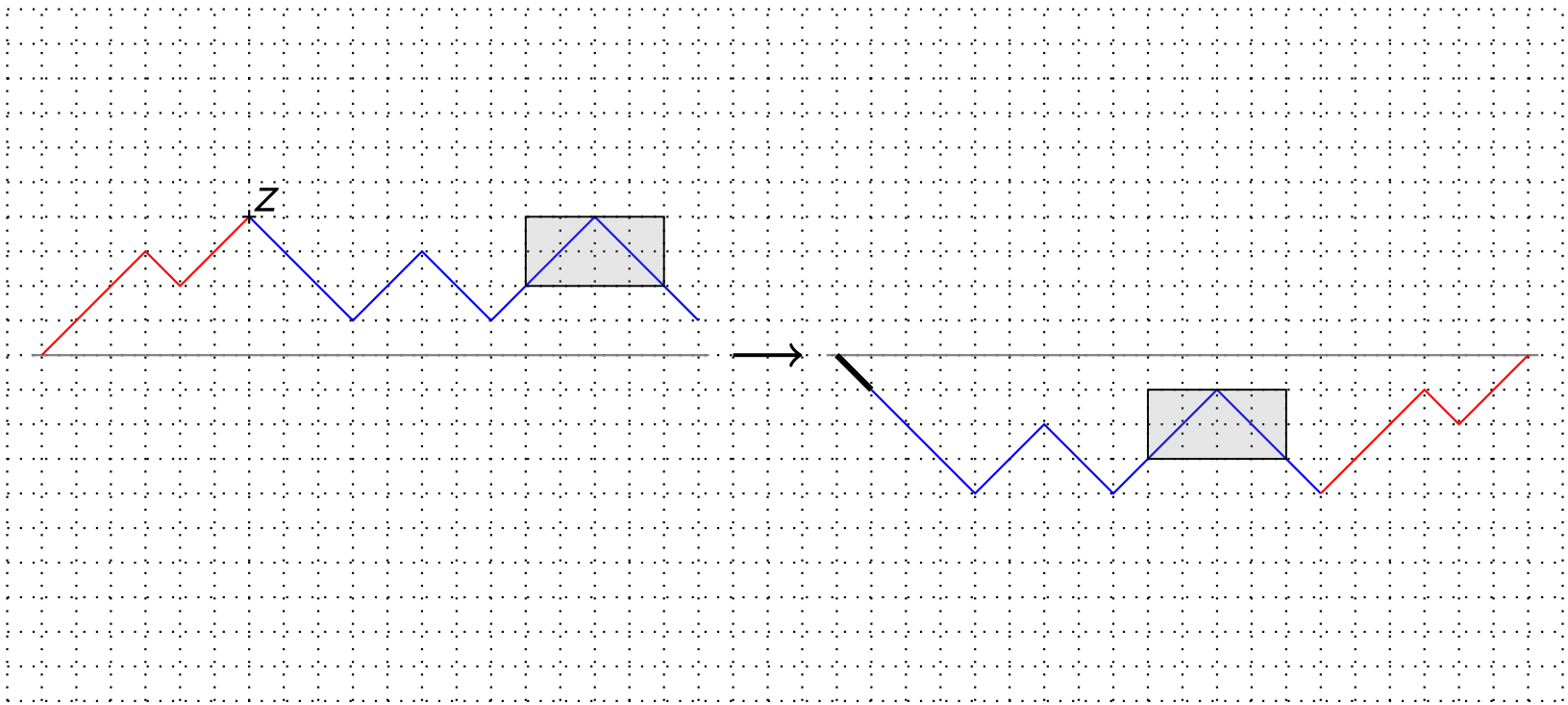
The constructive algorithm: Second step - case 2

Cut and paste



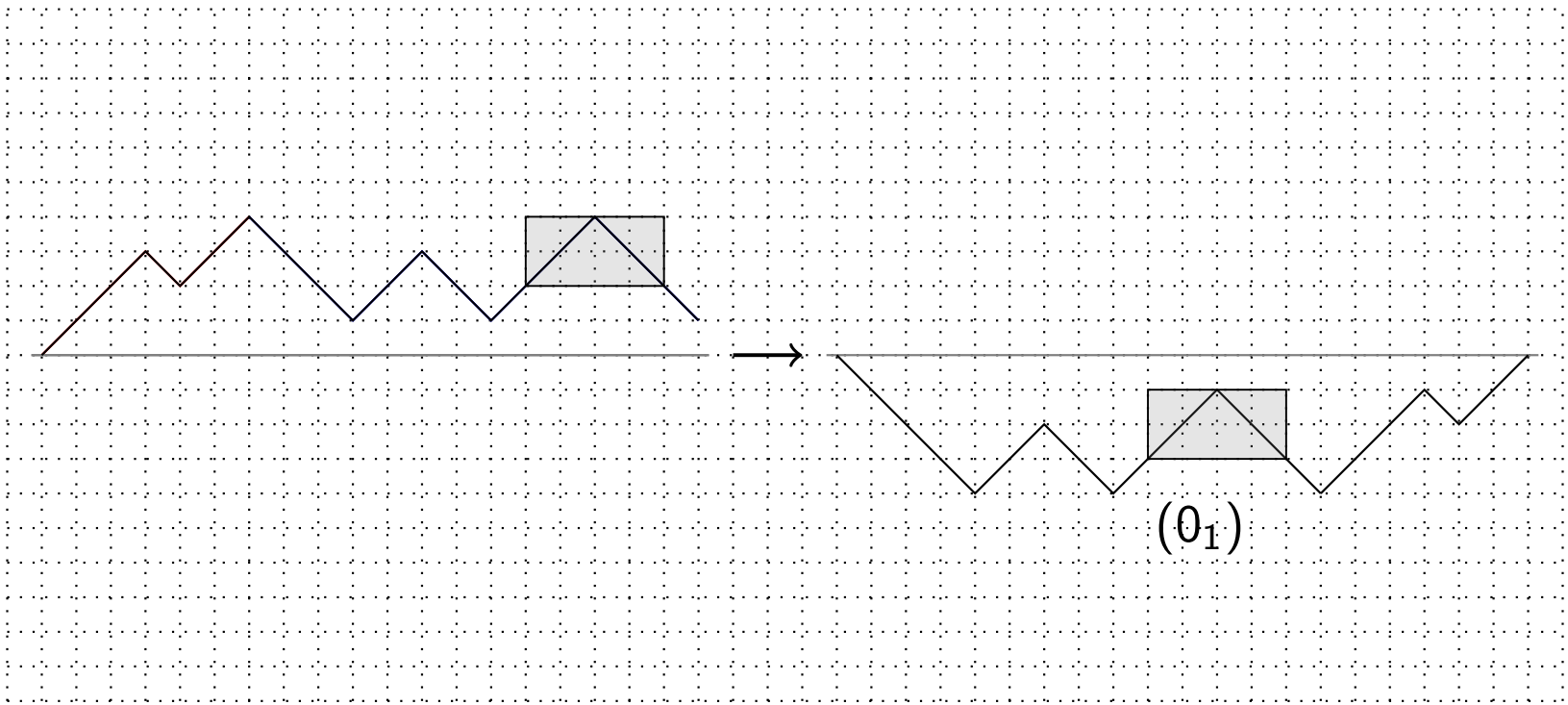
The constructive algorithm: Second step - case 2

Cut and paste



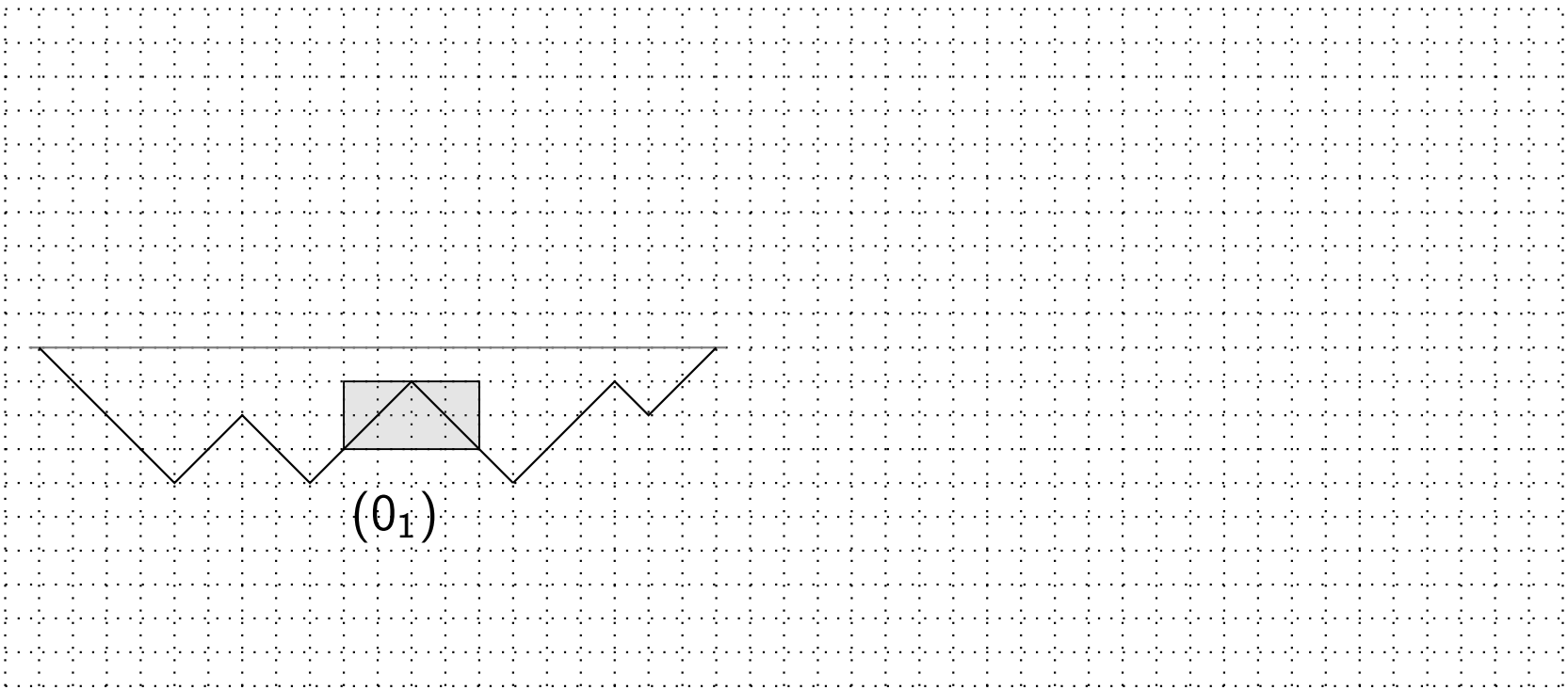
The constructive algorithm: Second step - case 2

Cut and paste



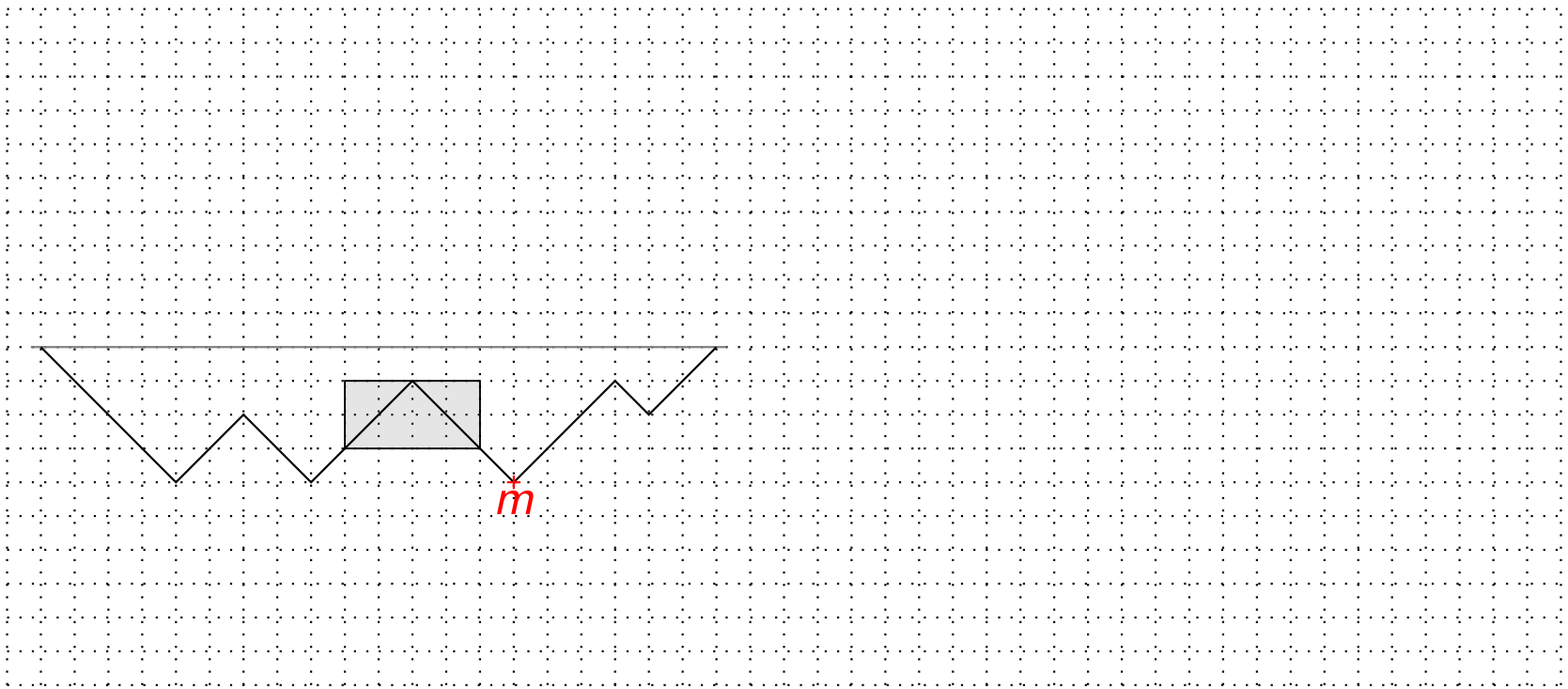
The constructive algorithm: Second step - case 2

Reverse of cut and paste



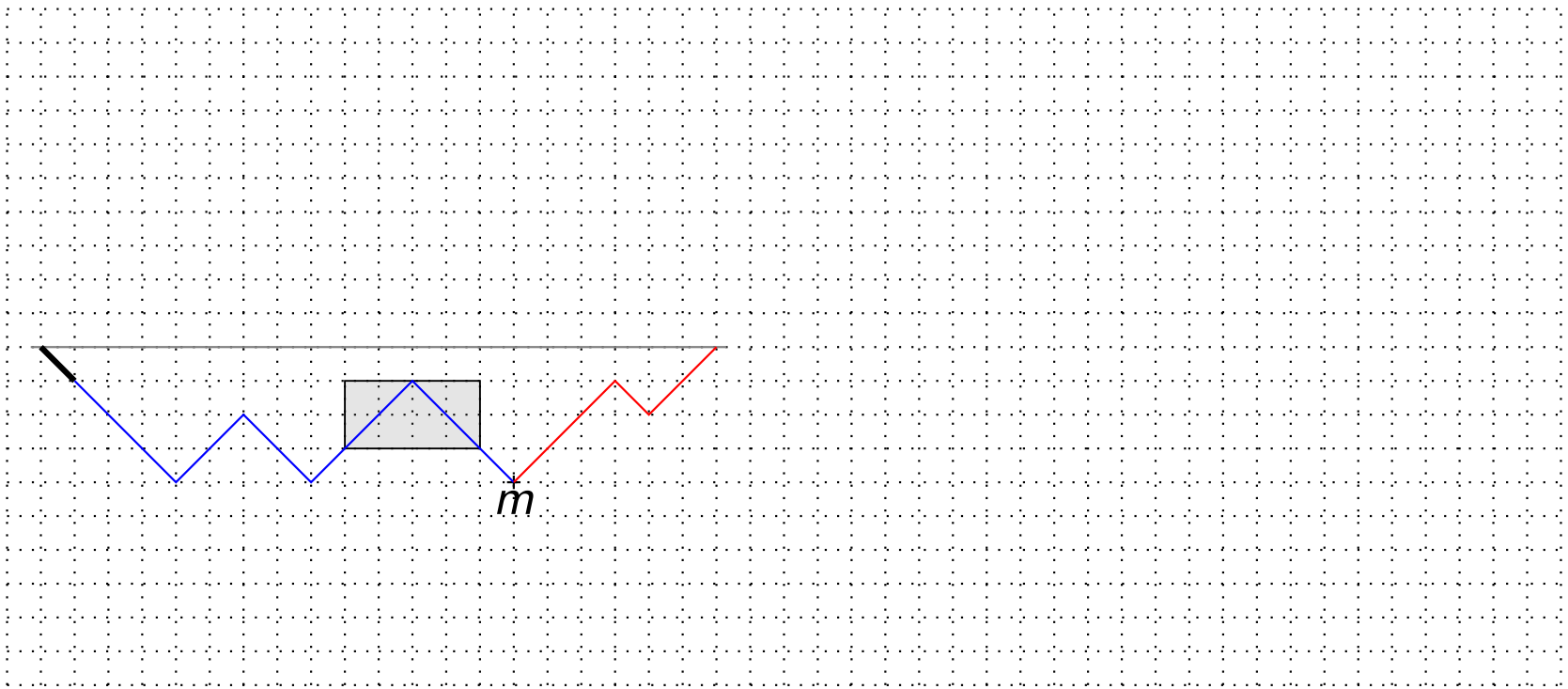
The constructive algorithm: Second step - case 2

Reverse of cut and paste



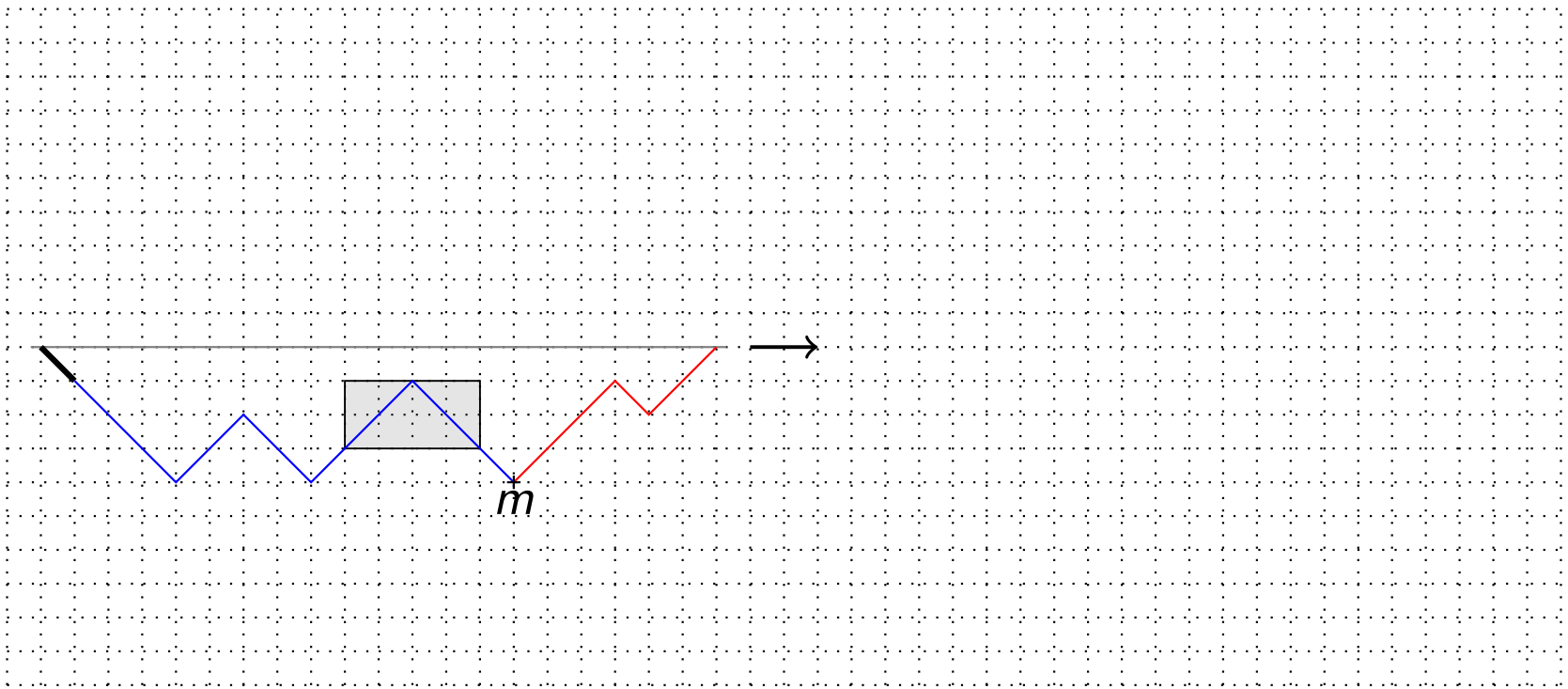
The constructive algorithm: Second step - case 2

Reverse of cut and paste



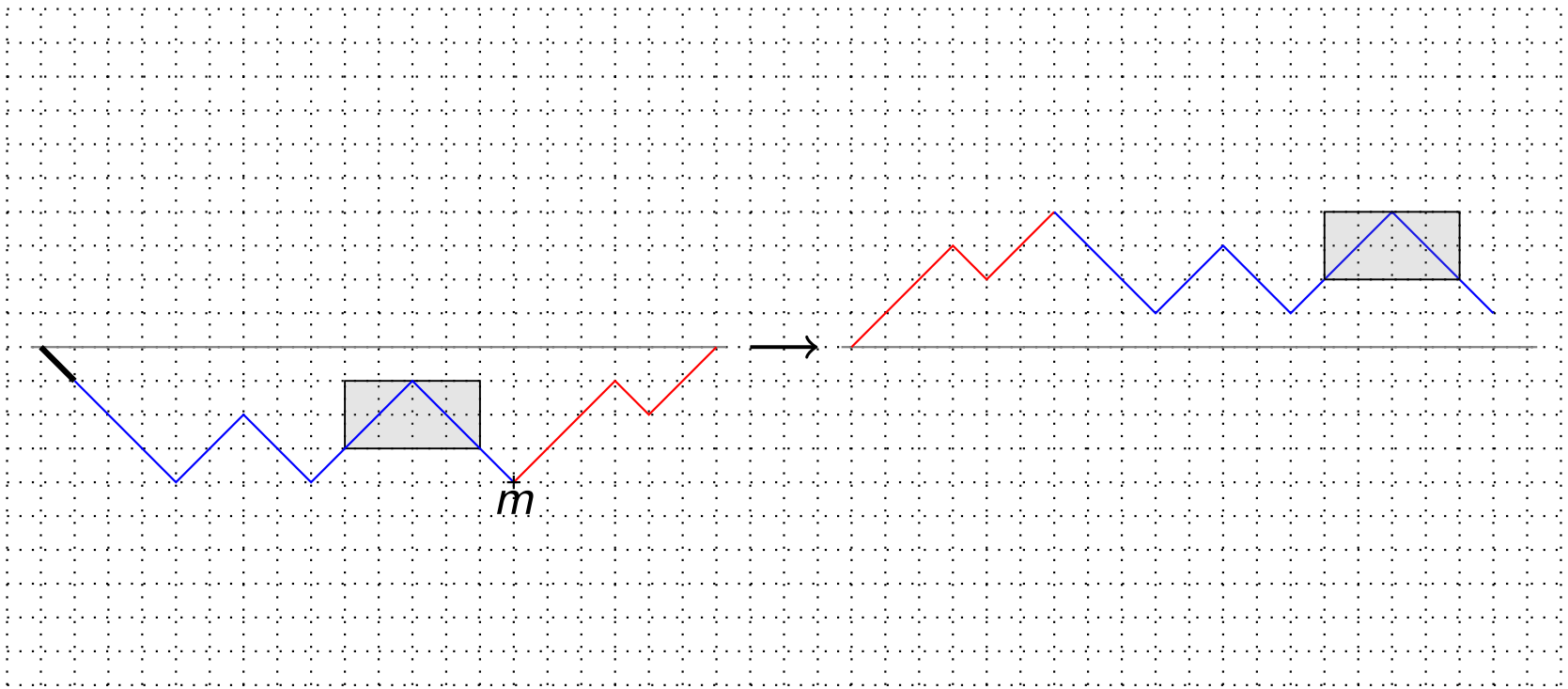
The constructive algorithm: Second step - case 2

Reverse of cut and paste



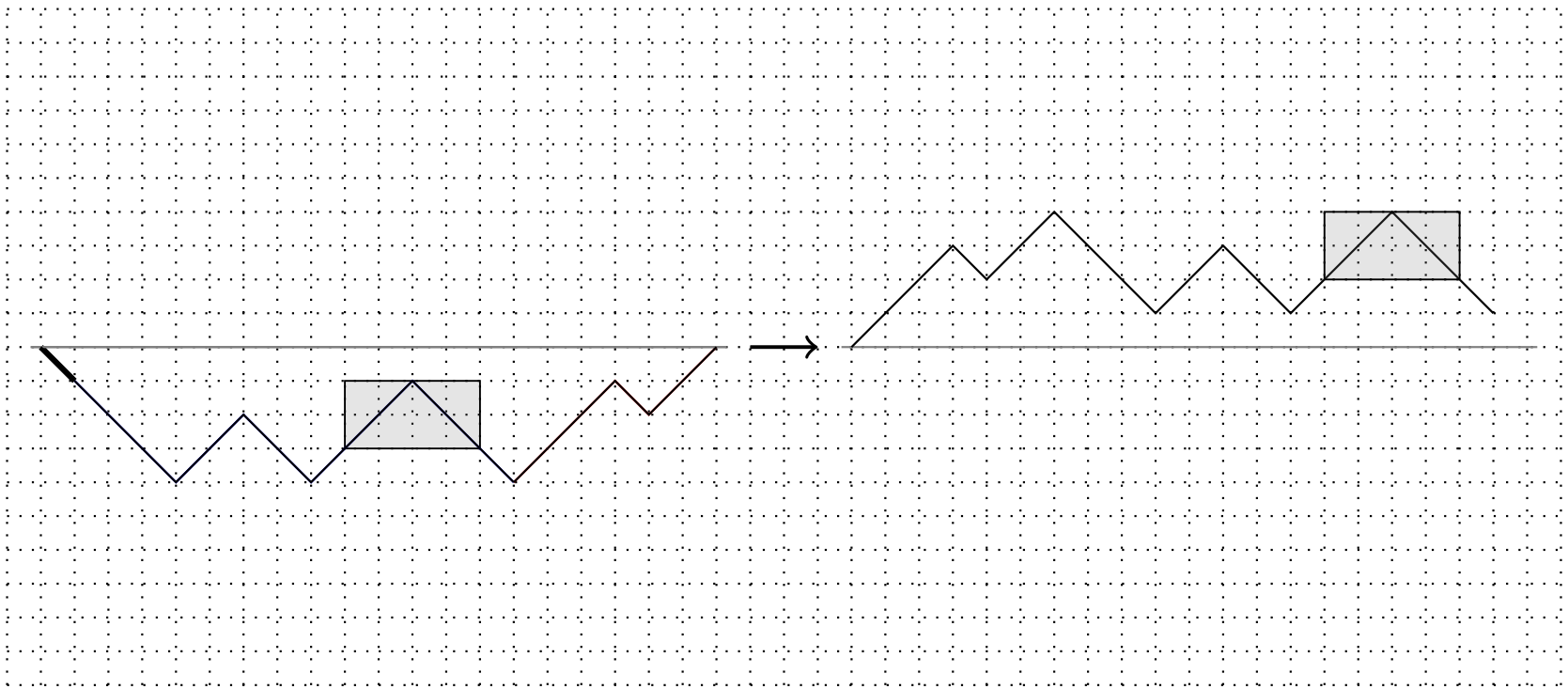
The constructive algorithm: Second step - case 2

Reverse of cut and paste

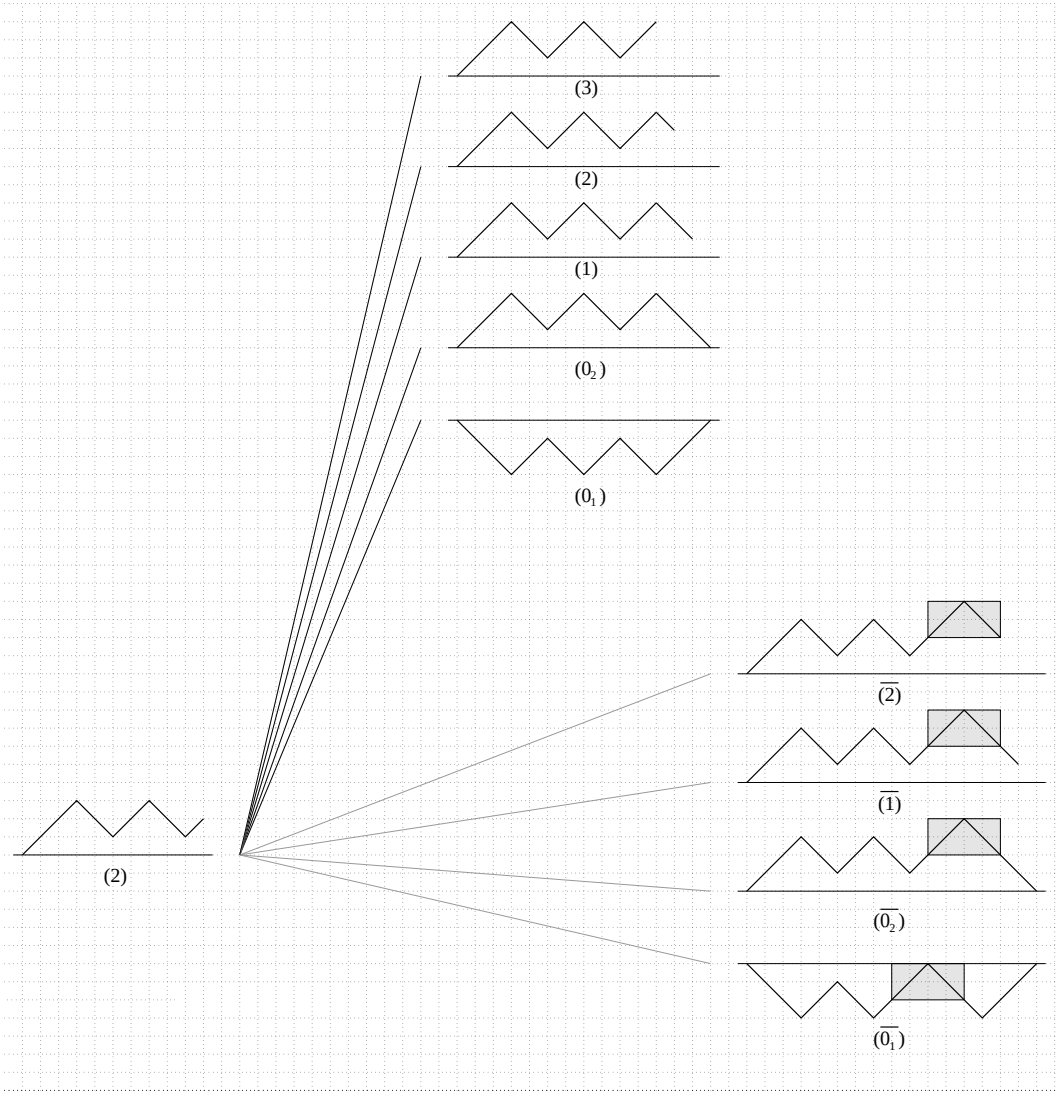


The constructive algorithm: Second step - case 2

Reverse of cut and paste

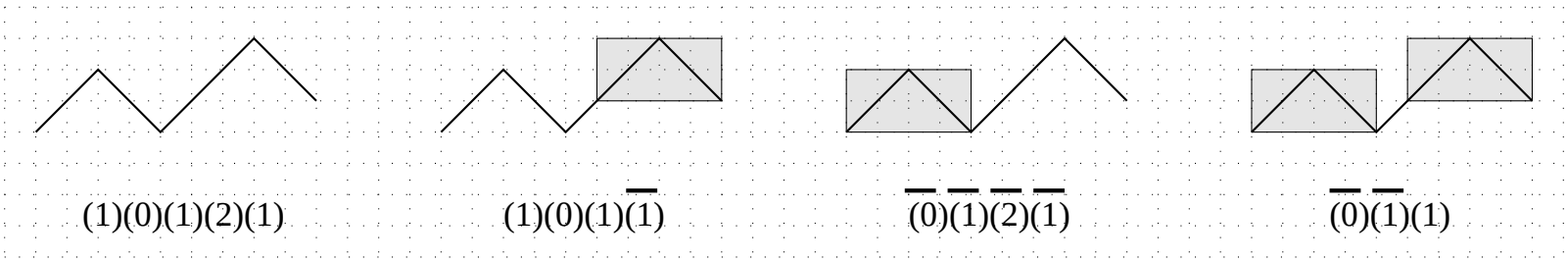


Example of the complete algorithm



Proof: idea

The described algorithm is a construction for the set $F^{[\mathfrak{p}]}$ according to the number of rise steps. This means that all the paths in F with n rise steps are obtained. Moreover, for each obtained path ψ in $F \setminus F^{[\mathfrak{p}]}$, having C forbidden patterns, with n rise steps and (k) as last label of the associated code, a path ψ' in $F \setminus F^{[\mathfrak{p}]}$ having C forbidden patterns, with n rise steps, and $(\overline{k})$ as last label of the associated code is also generated having the same shape as ψ but such that the last forbidden pattern is marked if it is not in ψ and vice-versa.



Enumeration

Let Z be the set of paths whose instances are coded by a sequence of labels in the generating tree ending by a non-marked zero, S be the set of paths whose instances are coded by a sequence of labels ending by a marked zero, N be the set of paths whose instances are coded by a sequence of labels ending by a non-marked $k \geq 1$ and M be the set of paths whose instances are coded by a sequence of labels ending by a marked $k \geq 1$.

Then $F^{[\mathfrak{p}]} = (Z \backslash S) \cup (N \backslash M)$.

$$Z(x,1) = \sum_{\omega \in Z} x^{n(\omega)} y^0 = 1 + 2xZ(x,1) + 2xN(x,1) + x^j S(x,1) + 2x^j M(x,1),$$

$$S(x,1) = \sum_{\omega \in S} x^{n(\omega)} y^0 = 2xS(x,1) + 2xM(x,1) + x^j Z(x,1) + 2x^j N(x,1),$$

$$N(x,y) = \sum_{\omega \in N} x^{n(\omega)} y^{h(\omega)} = xyZ(x,1) + \sum_{\omega \in N} \sum_{i=1}^{h(\omega)+1} x^{n(\omega)+1} y^i + \sum_{\omega \in M} \sum_{i=1}^{h(\omega)} x^{n(\omega)+j} y^i,$$

$$M(x,y) = \sum_{\omega \in M} x^{n(\omega)} y^{h(\omega)} = xyS(x,1) + \sum_{\omega \in M} \sum_{i=1}^{h(\omega)+1} x^{n(\omega)+1} y^i + \sum_{\omega \in N} \sum_{i=1}^{h(\omega)} x^{n(\omega)+j} y^i.$$

Enumeration

We obtain the generating function $F_j(x)$, $j > 1$, for the words $\omega \in F^{[\mathfrak{p}]}$ according to the number of ones:

$$F_j(x) = \frac{y_0(x) - x^{j-1}}{(1 - x^{j-1})(1 + x^j - 2xy_0(x))},$$

where

$$y_0(x) = \frac{x^j + 1 - \sqrt{(x^j + 1)^2 - 4x}}{2x}.$$

Enumeration

Let us remark that the generating function $F_j(x)$ depends only on the number of ones in the forbidden pattern. So, bifix-free Grand Dyck words represent an *invariant class* for the enumeration of the words which avoid anyone of such a shape path. This means that all the sets in F with n ones avoiding a forbidden pattern $\mathfrak{p} \in \mathcal{G}_j^b$ have the same cardinality, independently on the shape of $\mathfrak{p}$.

Example

For any pattern $\mathfrak{p}$ in $\mathcal{G}_4^b$, the first numbers of the sequence enumerating the binary words in $F^{[\mathfrak{p}]}$, according to the number of ones, are:
$1, 3, 10, 35, 125, 454, 1671, 6211, 23261, \cdots$ being

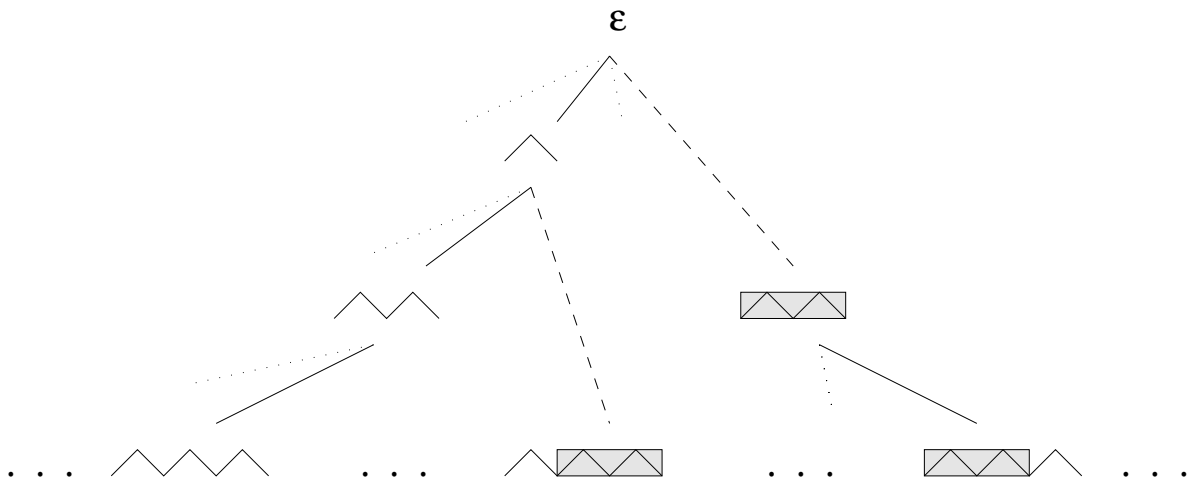
$$F_4(x) = \frac{1 - x^4 - \sqrt{x^8 + 2x^4 + 1 - 4x}}{2x(1 - x^3)\sqrt{x^8 + 2x^4 + 1 - 4x}}$$

the associated generating function.

Why *bifix-free* forbidden patterns?

The method works only for bifix-free forbidden patterns: if $\mathfrak{p}$ is not bifix-free, then the jumping and marked succession rule should generate paths containing forbidden patterns which could not be eliminated in the right way.

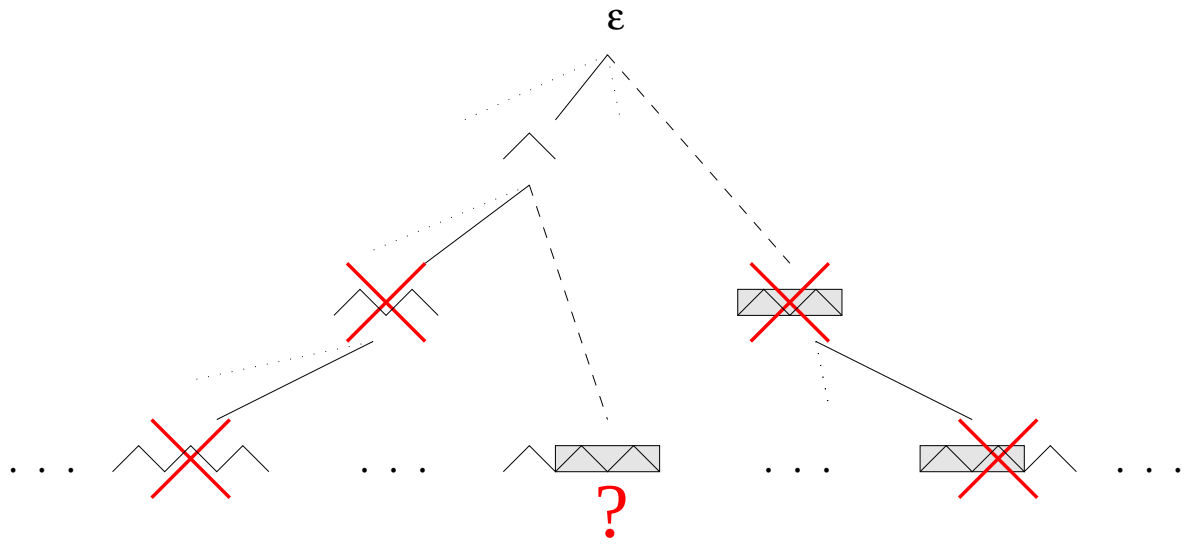
For example, when $\mathfrak{p} =$ 



Why *bifix-free* forbidden patterns?

The method works only for bifix-free forbidden patterns: if $\mathfrak{p}$ is not bifix-free, then the jumping and marked succession rule should generate paths containing forbidden patterns which could not be eliminated in the right way.

For example, when $\mathfrak{p} =$ 

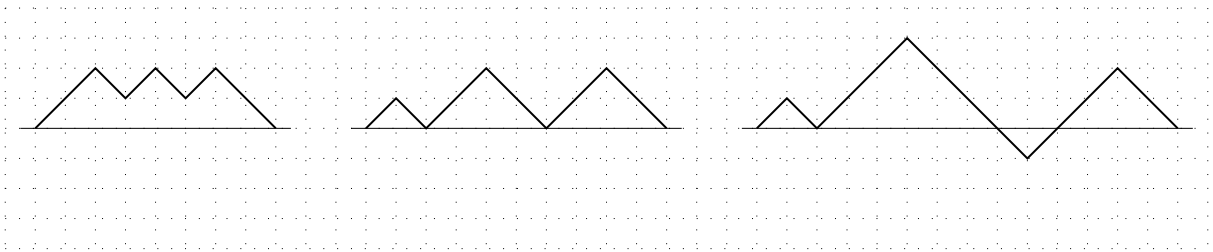


Cross-bifix-free patterns

In order to generalize our method we have to consider cross-bifix-free Grand Dyck words.

Two distinct bifix-free Grand Dyck words δ, δ' are said to be cross-bifix-free if and only if no prefix of δ is also a suffix of δ' and vice-versa.

A cross-bifix-free set is a set of words where any two words are cross-bifix-free.



Generalization

Given a set $\mathcal{P} = \{\mathfrak{p}_1, \mathfrak{p}_2, \cdots, \mathfrak{p}_m\}$ of cross-bifix-free Grand Dyck words, none included in any other, such that each $\mathfrak{p}_l$ having j_l as the number of its 1's, $1 \leq l \leq m$, then the growth of the set $F^{[\mathcal{P}]}$, according to the number of ones, can be described by the following jumping and marked succession rule:

$$
\begin{cases}
(0) & \\
(k) \overset{1}{\rightsquigarrow} (k+1)(k)\cdots(1)(0_2)(0_1) & k \geq 0 \\
(0) \overset{j_1}{\rightsquigarrow} (\overline{0_2}) & \\
(k) \overset{j_1}{\rightsquigarrow} (\overline{k})(\overline{k-1})\cdots(\overline{1})(\overline{0_2})(\overline{0_1}) & k \geq 1 \\
(0) \overset{j_2}{\rightsquigarrow} (\overline{0_2}) & \\
(k) \overset{j_2}{\rightsquigarrow} (\overline{k})(\overline{k-1})\cdots(\overline{1})(\overline{0_2})(\overline{0_1}) & k \geq 1 \\
\vdots & \\
(0) \overset{j_m}{\rightsquigarrow} (\overline{0_2}) & \\
(k) \overset{j_m}{\rightsquigarrow} (\overline{k})(\overline{k-1})\cdots(\overline{1})(\overline{0_2})(\overline{0_1}) & k \geq 1
\end{cases}
$$

Generalization

By using the previous notation, we have:

$$Z(x,1) = 1 + 2xZ(x,1) + 2xN(x,1) + (x^{j_1} + \cdots + x^{j_m})S(x,1) + 2(x^{j_1} + \cdots + x^{j_m})M(x,1),$$

$$S(x,1) = 2xS(x,1) + 2xM(x,1) + (x^{j_1} + \cdots + x^{j_m})Z(x,1) + 2(x^{j_1} + \cdots + x^{j_m})N(x,1),$$

$$N(x,y) = xyZ(x,1) + \sum_{\omega \in N} \sum_{i=1}^{h(\omega)+1} x^{n(\omega)+1} y^i + \sum_{\omega \in M} \sum_{i=1}^{h(\omega)} x^{n(\omega)+j_1} y^i + \cdots + \sum_{\omega \in M} \sum_{i=1}^{h(\omega)} x^{n(\omega)+j_m} y^i,$$

$$M(x,y) = xyS(x,1) + \sum_{\omega \in M} \sum_{i=1}^{h(\omega)+1} x^{n(\omega)+1} y^i + \sum_{\omega \in N} \sum_{i=1}^{h(\omega)} x^{n(\omega)+j_1} y^i + \cdots + \sum_{\omega \in N} \sum_{i=1}^{h(\omega)} x^{n(\omega)+j_m} y^i.$$

Generalization

We obtain the generating function $F_{j_1,\ldots,j_m}(x)$ for the words $\omega \in F^{[\mathcal{P}]}$ according to the number of ones:

$$F_{j_1,\ldots,j_m}(x) = \frac{y_0(x) - x^{j_1-1} - \cdots - x^{j_m-1}}{(1 - x^{j_1-1} - \cdots - x^{j_m-1})(1 + x^{j_1} + \cdots + x^{j_m} - 2xy_0(x))},$$

where

$$y_0(x) = \frac{1 + x^{j_1} + \cdots + x^{j_m} - \sqrt{(x^{j_1} + \cdots + x^{j_m} + 1)^2 - 4x}}{2x}.$$

Further developments

1. Study other forbidden patterns $\mathfrak{p}$
2. Expand the alphabet $\{0, 1\}$
3. Find a unified approach for pattern avoidance construction and enumeration

Further developments

1. Study other forbidden patterns $\mathfrak{p}$
2. Expand the alphabet $\{0, 1\}$
3. Find a unified approach for pattern avoidance construction and enumeration

Further developments

1. Study other forbidden patterns $\mathfrak{p}$
2. Expand the alphabet $\{0, 1\}$
3. Find a unified approach for pattern avoidance construction and enumeration

Thanks for your attention