Technical Report

Pattern Recognition and Image Processing Group
Institute of Computer Aided Automation
Vienna University of Technology
Favoritenstr. 9/1832
A-1040 Vienna AUSTRIA
Phone: +43 (1) 58801-18351
Fax: +43 (1) 58801-18392
E-mail: brun@leri.univ-reims.fr, krw@prip.tuwien.ac.at
URL: http://www.prip.tuwien.ac.at/

PRIP-TR-63 June 9, 2000

The Construction of Pyramids with Combinatorial Maps

Luc Brun and Walter Kropatsch[1]

Abstract

This paper presents a new formalism for irregular pyramids based on combinatorial maps. This technical report continues the work begun with the TR-54 and TR-57 reports (see [15] and [6]).We provide in this technical report algorithms allowing efficient parallel or sequential implementation of combinatorial pyramids

[1]This Work was supported by the Austrian Science Foundation under S7002-MAT.

Contents

1 Introduction

Objects that are mapped into the image plane induce spatial relations among each other and between their parts. Geometrical measurements derived from a digital image are very sensitive to errors due to noise, discrete sampling and motion inaccuracies. However these structural and topological relations are inherent to the objects and their arrangement in the image and mostly do not depend on the particular imaging situation. This is the background of several recent contributions describing spatial/structural representations and transformations preserving existing topological relations in the image plane. Following list enumerates a few possibilities to preserve structural relations into a more abstract representation:

1. The simplest one uses coordinates as vertex attributes of an attributed relational graph. This immediate representation depends on the particular mapping geometry. For well controlled environments (e.g. geographic information systems) it is widely used due to its simplicity.

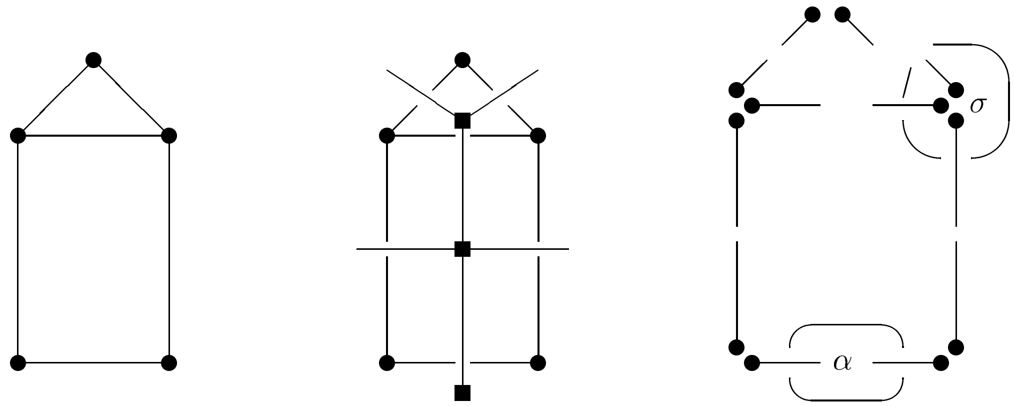
2. Another approach [17] considers local deformations of digital curves that preserve an implicitly given topology. The idea is that images showing the same topological arrangement of regions and curves can be transformed into each other. An interesting extension to higher dimension is presented by Fourey and Malgouyres [7].

3. A pair of plane[1] dual graphs is the base of an irregular graph pyramid built by repeated dual graph contractions [12]. It differs from the previous approach that the transformed data are reduced at each step by a factor which is the origin of its computational efficiency.

4. Topological and combinatorial maps have been investigated in [8] and [14]. There the embedding is determined by the local orientation of the structural elements. These works have been the basis of our two preceding technical reports [15, 6].

The rest of this report is structurate as follows: First we briefly recall the main results of our previous technical reports, in the sections 1.1, 1.2 and 1.3. Then, we present in Section 2 an implicit representation of combinatorial map

[1] A plane graph is an embedded planar graph. We purposely use the term 'plane' because two embeddings of the same planar graph need not be topologically isomorphic.



(a) A plane graph (b) decomposed along dual edges (c) combinatorial map

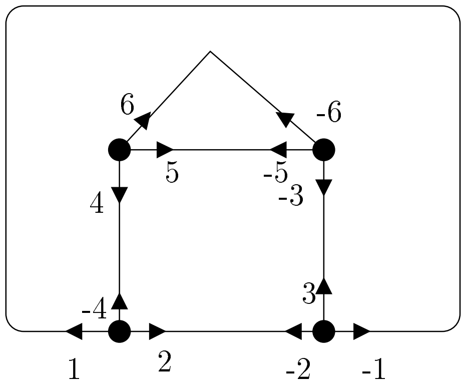
Figure 1: From a plane graph to a combinatorial map

pyramids defined by a sequence of contractions, or a sequence of removals. An explicit representation of combinatorial maps pyramid is also proposed. In Section 3 we extend this new encoding of a combinatorial map pyramid by allowing contractions and removal operations during the construction of the pyramid. Note that one step of the construction of the combinatorial map pyramid is encoded by kernels (see [6] and Section 1.3) and thus encodes only one type of operation.

1.1 Combinatorial Maps

A combinatorial map may be seen as a planar graph encoding explicitly the orientation of edges around a given vertex. Thus all graph definitions used in irregular pyramids [13] such as end vertices, self loops, or degrees may be retrieved easily.

Figure 1 demonstrates the derivation of a combinatorial map from a plane graph. First edges are split where their dual edges cross (see Figure 1-b). That decomposes the graph into connected parts of half-edges that surround each vertex. These half edges are called *darts* and have their origin at the vertex they are attached to. The fact that two half-edges (darts) stem from the same edge is recorded in the **reverse permutation** α. A second permutation σ, called the **successor permutation**, defines the (local) arrangement of darts around a vertex. Counterclockwise ordering is assumed here. Figure 2 gives a slightly enhanced example of combinatorial map with 12 darts. The symbols $\alpha^*(d)$ and $\sigma^*(d)$ stand, respectively, for the α and σ orbits of



$$\sigma = (1, 2, -4)(-2, -1, 3)(-3, -6, -5)(4, 5, 6)$$

Figure 2: *The permutation* σ

the dart d. More generally, if d is a dart and π a permutation we will denote the π-orbit of d by $\pi^*(d)$. The cardinal of this orbit will be denoted $|\pi^*(d)|$.

A **combinatorial map** G is the triplet $G = (\mathcal{D}, \sigma, \alpha)$, where $\mathcal{D}$ is the set of darts and σ, α are two permutations defined on $\mathcal{D}$ such that α is an involution, e.g. satisfying

$$\forall d \in \mathcal{D} \quad \alpha^2(d) = d$$

If the darts are encoded by positive and negative integers, the permutation α can be implicitly encoded by $\alpha(d) = -d$ (see Figure 2). In the following, we will use alternatively both notations, the notation $\alpha(d) = -d$ will be often use for practical results linked to the implementation of our model. Indeed, if the permutation α is implicitly encoded, the combinatorial map may be implemented by a basic array of integers encoding the permutation σ, which looks as follows for Fig. 2:

d	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5	6
$\sigma(d)$	-5	-3	1	-6	-1	3		2	-4	-2	5	6	4

Following concepts from graph theory that are needed later for structure preserving operations can be expressed in terms of combinatorial maps: self-loop, duality, and bridge. An edge $\alpha^*(d)$ is called a self loop, iff: $-d \in \sigma^*(d)$. Or, if the two endpoints of an edge are the same vertex.

A face of a planar graph is defined by the set of edges which surround it. Using a combinatorial map, one dart per edge is sufficient to encode a face, since for each dart the involution α allows us to retrieve the other dart defining the edge. Moreover, the ordered sequence of darts around a vertex encoded by permutation σ induce an order in the sequence of faces encountered when turning around a face. This order is encoded thanks to the permutation $\varphi = \sigma \circ \alpha$: Given a combinatorial map $G = (\boldsymbol{\mathcal{D}}, \sigma, \alpha)$, the combinatorial map $\overline{G} = (\boldsymbol{\mathcal{D}}, \varphi, \alpha)$ is called **dual combinatorial map** of G. The orbits of φ encode the faces of G. Note that the function φ is a permutation, since it is the composition of two permutations on the same set. Using a clockwise orientation for permutation σ all the faces of the combinatorial map except one are counter-clockwise oriented. The clockwise oriented face is called the infinite face. The dual map of Fig. 2 is given as follows:

d	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5	6
$\varphi(d)$	4	6	5	-2	-4	2		3	-1	-6	1	-3	-5

The connectivity of a graph (or a subgraph representing an object) is an essential structural property. Since our goal is to successively remove unnecessary parts the connectivity can be lost by these operations. Before disconnecting a graph into two components these two components will be connected by a single edge which is called a **bridge** which can be characterized by

$$\alpha(d) \in \varphi^*(d)$$

1.2 Contraction and Removal

In order to preserve the number of connected components of the original combinatorial map bridges must be excluded from removal operations. Using this restriction, the removal operation may be expressed as the definition of a sub combinatorial map without the removed edges. A formal definition of the removal operation written in terms of modifications of permutation σ is given in [15].

Given a partition of an image, merging two regions may be considered in two different ways: First we can consider that the two regions are merged by removing one of their common boundaries. This operation is encoded in our combinatorial map formalism by the edge removal. Secondly, we can also consider that the two regions are merged by identifying the two regions and removing one of their common boundaries. This dual point of view is encoded in our formalism by the contraction operation.

Using the duality we define the **contraction** of dart d of a given combinatorial map $G = (\boldsymbol{\mathcal{D}}, \sigma, \alpha)$ which is not a self loop. The result is the following graph

$$G' = G/\alpha^*(d) = \overline{\overline{G} \setminus \alpha^*(d)}$$

Note that this operation is well defined since d is a self-loop in G iff it is a bridge in $\overline{G}$.

Note that, under the same hypothesis, we have:

$$\overline{\overline{G}/\alpha^*(d)} = G \setminus \alpha^*(d)$$

Thus the two dual points of view on merging regions are performed by two dual operations on the combinatorial map and its dual. Thus many particular cases of one operation may be retrieved thanks to the particular cases of the other. For example, since bridges are forbidden for removal operation the dual of a bridge, i.e. a self-loop, is forbidden for contraction.

1.3 Equivalent Contraction Kernels

The concept of a tree and of a forest are used to define a contraction kernel that collects a set of darts that can be contracted independently of each other without destroying the connectivity structure of the graph. A sequence of merging segments of a partition may be encoded by a sequence of contractions of the combinatorial map encoding the partition. Since the contraction operation is forbidden for self-loops the set of darts involved in such a sequence of contractions must not contain a circuit. Thus the set of edges involved in such a contraction may be encoded by a tree which is a sub-map of the combinatorial map $G = (\boldsymbol{\mathcal{D}}, \sigma, \alpha)$ with only one φ'-orbit. The only dual face of a tree is the background face.

More generally, if we contract a set of vertices into a given set of surviving vertices, the set of darts involved in such contractions may be encoded by a forest $F = (\boldsymbol{\mathcal{D}}_1, \ldots, \boldsymbol{\mathcal{D}}_n)$ which is a collection of non-overlapping trees spanning the given combinatorial map $G = (\boldsymbol{\mathcal{D}}, \sigma, \alpha)$.

The forest $K = (\boldsymbol{\mathcal{D}}_1, \ldots, \boldsymbol{\mathcal{D}}_n)$ of G will be called a **contraction kernel** iff:

$$\mathcal{SD} = \boldsymbol{\mathcal{D}} - \bigcup_{i=1}^{n} \boldsymbol{\mathcal{D}}_i \neq \emptyset$$

The set $\mathcal{SD}$ is called the set of surviving darts.

We can apply successively two (and more) contraction kernels K_{01} and K_{12} to a given combinatorial map G_0: $G_1 = G_0/K_{01}$ and $G_2 = G_1/K_{12}$. The same result can be achieved by applying a bigger kernel only once: $G_2 = G_0/K_{02}$. Conversely, a contraction kernel may be decomposed into two smaller ones. The successive application of the resulting contraction kernels is equivalent to the application of the initial one. Different contraction kernels on the same combinatorial map G_0 may be related by **inclusion**, successive kernels give rise to **predecessor** and **successor** relations which allow us to formulate the above mentioned equivalences:

Inclusion of Contraction Kernels: Let us consider two different contraction kernels K_{01} and K_{02} defined on a combinatorial map G_0. We will say that the contraction kernel K_{02} includes K_{01} iff $K_{01} \subset K_{02}$. In this case each connected component, a tree $\mathcal{T}_1$ of K_{01} is included in exactly one connected component, a tree $\mathcal{T}_2$ of K_{02}:

$$\forall \mathcal{T}_1 \in \mathcal{CC}(K_{01}) \exists! \ \mathcal{T}_2 \in \mathcal{CC}(K_{02}) \text{ s.t. } \mathcal{T}_1 \subset \mathcal{T}_2.$$

Predecessor and Successor Kernels Given a combinatorial map $G_0 = (\boldsymbol{\mathcal{D}}, \sigma, \alpha)$, a contraction kernel K_{01} of G_0 and the contracted combinatorial map $G_1 = G_0/K_{01}$. If K_{12} is a contraction kernel of G_1 then we say that K_{01} is the predecessor of K_{12}, or that K_{12} is the successor of K_{01}. This relation will be denoted $K_{01} \prec K_{12}$.

The successive application of K_{01} and K_{12} forms a new operator on G_0 denoted by $K_{12} \circ K_{01}$.

Based on these two definitions two theorems could be formulated in TR-57 [6] that relate composition and decomposition of contraction kernels:

Theorem 4 in [6] derives inclusion kernels from successor kernels:

$$K_{01} \prec K_{12} \Longrightarrow K_{01} \subset K_{02} = K_{01} \cup K_{12}$$

$$\text{with } \left(G_0/K_{01}\right)/K_{12} = G_0/K_{02}.$$

The kernel K_{02} combines kernel K_{01} with the subtrees of K_{12} such that that the result of contracting G_0 with K_{02} is the same as if G_0 is contracted with K_{01} and with K_{02} in succession.

Theorem 6 in [6] derives successor kernels from inclusion kernels:

$$K_{01} \subset K_{02} \Longrightarrow K_{01} \prec K_{12} = K_{02} - K_{01}$$
$$\text{with } G_0/K_{02} = (G_0/K_{01})/K_{12}.$$

Given two contraction kernels K_{01}, K_{02} for G_0, K_{01} being included in K_{02}, the larger kernel K_{02} can be decomposed into K_{01} and the successor kernel K_{12} which can be used after contracting G_0 with K_{01} to yield the same result.

The definitions of connecting walk and the application $follow$ from TR-57 [6] are adapted here to clearly identify the pyramid levels of both the input and the output elements.

Definition 1 Connecting walk

Given an initial connected combinatorial map $G_0 = (\boldsymbol{\mathcal{D}}, \sigma, \alpha)$ and a contraction kernel K_{ij} we associate to each dart d of $\mathcal{SD}_j$ a **connecting walk** *$CW_{ij}(d)$ defined on $\mathcal{SD}_i$ by:*

$$CW_{ij}(d) = (d, \varphi_i(d), \ldots, \varphi_i^{n-1}(d)) \text{ with } n = Min\{p \in \mathbb{N}^* \mid \varphi_i^p(d) \in \mathcal{SD}_j\}$$

Definition 2 Function follow *Given an initial connected combinatorial map $G_0 = (\boldsymbol{\mathcal{D}}, \sigma, \alpha)$ and a contraction kernel K_{ij} the application $follow_{ij}(d)$ relates the dart $d \in \mathcal{SD}_j$ with its successor in $\mathcal{SD}_j$ through the connecting walk $CW_{ij}(d) \subset \mathcal{SD}_i$:*

$$follow_{ij}(d) = \varphi_i^n(d) \text{ with } n = Min\{p \in \mathbb{N}^* \mid \varphi_i^p(d) \in \mathcal{SD}_j\}$$

We have shown in TR-57 [6] that the set of connecting walks defined by an initial combinatorial map and a contraction kernel K_{ij} may be structured into a combinatorial map GC_{ij} such that GC_{ij} is isomorph to $G_j = G_i/K_{ij}$.

2 Coding All Contractions of a Pyramid

We will study in this section an encoding of a sequence of contractions defined by a sequence of successor or inclusion kernels. The basic idea of this encoding is to store for each dart the index of the contraction kernel which encloses it. We will show that this information is sufficient to retrieve all the contraction kernels and all the contracted combinatorial maps.

2.1 Coding the life time of a dart

Given a sequence of successive contraction kernels $K_{01} \prec K_{12} \prec \ldots \prec K_{n-1,n}$ we can consider the sequence of inclusion kernels $K_{01} \subset K_{01} \cup K_{12} \subset \ldots \subset \cup_{i=1}^{n} K_{i-1,i}$ which provides the same series of contracted combinatorial maps (see Section 1.3 and [6]). We can thus, without loss of generality, restrict our study to inclusion kernels. In this last case, all the connecting walks are defined on the same initial combinatorial map.

Proposition 1 *Given a combinatorial map $G_0 = (\boldsymbol{\mathcal{D}}, \sigma, \alpha)$, and two contraction kernels K_{01} and K_{02}, $K_{01} \subset K_{02}$. The connecting walks of K_{02} include the connecting walks of K_{01} (see Def. 1):*

$$\forall d \in \mathcal{SD}_2 \quad CW_{01}(d) \subset CW_{02}(d)$$

Proof:

Both connecting walks are defined by:

$$\begin{array}{lcll} CW_{01}(d) & = & (d, \varphi_0(d), \ldots, \varphi_0^{n-1}(d)) & \text{with} \quad n = Min\{p \in \mathbb{N}^* \quad | \varphi_0^p(d) \in \mathcal{SD}_1\} \\ CW_{02}(d) & = & (d, \varphi_0(d), \ldots, \varphi_0^{m-1}(d)) & \text{with} \quad m = Min\{p \in \mathbb{N}^* \quad | \varphi_0^p(d) \in \mathcal{SD}_2\} \end{array}$$

Since $\mathcal{SD}_2 \subset \mathcal{SD}_1$ we have $m \geq n$ and thus $CW_{01}(d) \subset CW_{02}(d)$. □

This result is illustrated in Figure 3.

Proposition 2 *Given a combinatorial map $G_0 = (\boldsymbol{\mathcal{D}}, \sigma, \alpha)$, and two contraction kernels K_{01} and K_{02}, $K_{01} \subset K_{02}$. Each connecting walk of K_{01} is included in exactly one connecting walk of K_{02}:*

$$\forall d \in \mathcal{SD}_1 \quad \exists! d' \in \mathcal{SD}_2 \quad \textit{such that } CW_{01}(d) \subset CW_{02}(d')$$

Proof:

We know that, given a contraction kernel, each dart belongs to exactly one connecting walk (see [6]). Thus there exists a unique dart d' in $\mathcal{SD}_2$ such that $d \in CW_{02}(d')$.

Let us consider n such that $\varphi^n(d) = follow_{01}(d)$ (see Def. 2). The sequence $CW_{01}(d) = (d, \varphi(d), \ldots, \varphi^{n-1}(d))$ is included in $K_{01} \subset K_{02}$ by definition of a connecting walk. A sequence of φ-consecutive darts included in K_{02} is included in exactly one connecting walk of K_{02}:

$$CW_{01}(d) \subset CW_{02}(d')$$

□

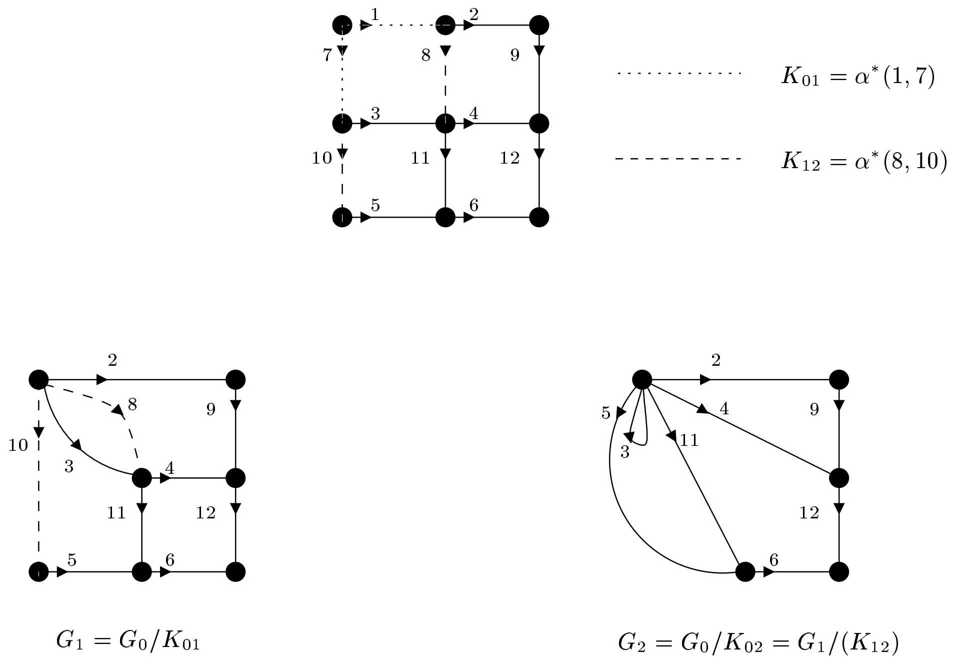


Figure 3: *This figure shows two included contraction kernels* K_{01} *and* K_{02}. *Each connecting walk of* K_{02} *defined by one dart in* $\mathcal{SD}_2 = \alpha^*(2,3,4,5,6,9,11,12)$ *includes the corresponding connecting walk of* K_{01}. *We have for example,* $CW_{01}(3) = 3$, *while* $CW_{02}(3) = 3, -8$

Proposition 3 *Given a combinatorial map $G_0 = (\boldsymbol{\mathcal{D}}, \sigma, \alpha)$, and two contraction kernels K_{01} and K_{02}, $K_{01} \subset K_{02}$. Each connecting walk of K_{02} is equal to a concatenation of connecting walks of K_{01}:*

$$\begin{array}{ll} \forall d \in \mathcal{SD}_2 & CW_{02}(d) = CW_{01}(d_1) \cdots CW_{01}(d_p) \\ \textit{with} & CW_{12}(d) = (d_1, \dots, d_p) \end{array}$$

Proof:

Given a dart $d \in \mathcal{SD}_2$, let us consider the ordered set

$$(d_1, \dots, d_n) = CW_{02}(d) \cap \mathcal{SD}_1$$

The order of the sequence $(d_1, \dots, d_n)$ is deduced from the order defined in $CW_{02}(d)$. Note that we have $d_1 = d$, since $d \in \mathcal{SD}_2 \subset \mathcal{SD}_1$ and d is the first dart of $CW_{02}(d)$.

Let us consider two cases:

- If $n = 1$, then the walk $CW_{02}(d) - \{d\}$ does not contain any surviving dart of $\mathcal{SD}_1$. Therefore, $\varphi(d) \in \mathcal{SD}_2$ and:

  $$CW_{02}(d) = CW_{01}(d) = (d)$$

  Moreover, we have in this case, $\varphi_1(d) = follow_{01}(d) = \varphi(d) \in \mathcal{SD}_2$. Thus $CW_{12}(d) = (d)$.

- If $n > 1$, each dart d_i is enclosed in $CW_{02}(d)$, thus we have by proposition 2:

  $$\forall \{d_1, \dots, d_n\} \in \mathcal{SD}_1 \quad CW_{01}(d_i) \subset CW_{02}(d)$$

  Moreover, by definition of a connecting walk, all darts contained in $CW_{02}(d)$, and thus all $(d_i)_{i \in \{1,\dots,n\}}$ belong to the same φ-orbit. Therefore, we have by construction of the ordered set $(d_1, \dots, d_n)$, $follow_{01}(d_i) = d_{i+1}$. Thus $CW_{01}(d_1) \cdots CW_{01}(d_n)$ is a walk of G_0 included in $CW_{02}(d)$ starting from d.

  If $follow_{01}(d_n)$ belongs to $\mathcal{SD}_1 - \mathcal{SD}_2$, we can find another dart d_{n+1} in $CW_{02}(d) \cap \mathcal{SD}_1$ which contradicts the definition of n. Therefore, $follow_{01}$ belongs to $\mathcal{SD}_2$ and:

  $$CW_{02}(d) = CW_{01}(d_1) \cdots CW_{01}(d_n)$$

  Moreover, since $d_{i+1} = follow_{01}(d_i) = \varphi_1(d_i) \in \mathcal{SD}_1 - \mathcal{SD}_2$, for each i in $\{1, \dots, n-1\}$, the sequence $(d_1, \dots, d_n)$ is included in $CW_{12}(d)$. Since we have by hypothesis $follow_{01}(d_n) = \varphi_1(d_n) \in \mathcal{SD}_2$, the connecting walk $CW_{12}(d)$ must stop at d_n and we have:

$$CW_{12}(d) = (d_1 \dots d_n)$$

□

Using the example given in Figure 3, we obtain for the dart -3: $CW_{12}(-3) = (-3, 8)$, while $CW_{02}(-3) = (-3, -7, 1, 8) = CW_{01}(-3) \cdot CW_{01}(8)$.

Proposition 4 *Given a combinatorial map $G_0 = (\boldsymbol{\mathcal{D}}, \sigma, \alpha)$, and a sequence of contraction kernels $K_{0,1}, \dots, K_{n-1,n}$. For each i in $\{1, \dots, n-1\}$, each connecting walk $CW_{j,i+1}(d)$ with $j < i$ and $d \in \mathcal{SD}_{i+1}$ is equal to a concatenation of connecting walks defined by $K_{j,i}$:*

$$\left.\begin{array}{lcl} \forall i & \in & \{1, \dots, n-1\} \\ \forall j & \in & \{0, \dots, i-1\} \end{array}\right\} \forall d \in \mathcal{SD}_i \; CW_{j,i+1}(d) = CW_{j,i}(b_1) \cdots CW_{j,i}(b_p)$$

with $CW_{i,i+1}(d) = (b_1, \dots, b_p)$.

Proof:

Given two indexes j and i fulfilling the above conditions and a dart d in $\mathcal{SD}_{i+1}$, let us consider the two sequences of darts:

$$\begin{array}{rcl} CW_{j,i+1}(d) & = & (d_1, \dots, d_n) \quad and \\ CW_{j,i}(b_1) \cdots CW_{j,i}(b_p) & = & (d'_1, \dots, d'_q) \end{array}$$

with $CW_{i,i+1}(d) = (b_1, \dots, b_p)$.

Since the first dart of a connecting walk is equal to the dart which defines it we must have $b_1 = d'_1 = d_1 = d$. Let us denote the connecting walks $CW_{j,i}(b_k)$ by:

$$CW_{j,i}(b_k) = b_{k,1} \dots, b_{k,p_k}$$

By definition of a connecting walk, $\varphi_i(b_k) = b_{k+1} = \varphi_j(b_{k,p_k})$ for each k in $\{1, \dots, p-1\}$. Moreover, for each dart $b_{k,j}$ in $CW_{j,i}(b_k)$, $b_{k,j+1} = \varphi_j(b_{k,j})$.

Therefore, $CW_{j,i}(b_1) \cdots CW_{j,i}(b_p)$ is a sequence of φ_j-successors. Moreover, by definition of connecting walks:

$$\begin{array}{llcl} \forall k \in \{2, \dots, p\} & b_k & \in & K_{i,i+1} \\ \forall k \in \{1, \dots, p\} & CW_{j,i}(b_k) - \{b_k\} & \subset & K_{j,i} \end{array}$$

Therefore:

$$(CW_{j,i}(b_1) - \{b_1\}) \cdot CW_{j,i}(b_2) \cdots CW_{j,i}(b_p) \subset K_{j,i} \cup K_{i,i+1} = K_{j,i+1} \quad (1)$$

Since $CW_{j,i+1}(d) - \{d\}$ is the maximal sequence of φ_j-successors included in $K_{j,i+1}$ and starting from $\varphi_j(d = b_1)$:

$$CW_{j,i}(b_1) \cdots CW_{j,i}(b_p) \subset CW_{j,i+1}(d)$$

Using our notations we have $b_{p,p_p} = d'_q$. Moreover, by definition of the connecting walk $CW_{i,i+1}(d)$: $\varphi_i(b_p) = \varphi_j(d'_q) \in \mathcal{SD}_{i+1}$. Using equation 1, $\varphi_j(d'_q)$ is the first dart of the sequence of φ_j-successors starting from d which belongs to $\mathcal{SD}_{i+1}$. Therefore, by definition of a connecting walk $n = q$ and:

$$CW_{j,i+1}(d) = CW_{j,i}(b_1) \cdots CW_{j,i}(b_p)$$

□

Definition 3 Pyramid Construction Plan
Given a combinatorial map $G_0 = (\boldsymbol{\mathcal{D}}, \sigma, \alpha)$, and a sequence of inclusion kernels $K_{01} \subset K_{02} \ldots \subset K_{0n}$, the pyramid construction plan $\mathcal{LP} = (G_0, level)$, associated to this sequence of contractions of G_0, is defined by G_0 and a function level:

$$level \left(\begin{array}{rcl} \boldsymbol{\mathcal{D}} & \to & \{1, \ldots, n+1\} \\ d & \mapsto & \max\{i \mid d \in \mathcal{SD}_{i-1}\} \end{array} \right)$$

Note that since each set of surviving darts $\mathcal{SD}_i, i \in \{1, \ldots, n\}$ is symmetric with respect to α, the function level must satisfy the following property:

$$\forall d \in \boldsymbol{\mathcal{D}} \quad level(\alpha(d)) = level(d)$$

Therefore, if the pyramid construction plan is implemented with an implicit encoding of the involution α by the sign, only the level of positive darts needs to be stored.

Proposition 5 *Given a combinatorial map $G_0 = (\boldsymbol{\mathcal{D}}, \sigma, \alpha)$, and a pyramid construction plan $\mathcal{LP} = (G_0, level)$ defined by n contraction kernels, each dart of level $i \leq n$ belongs to $K_{i-1,i}$:*

$$K_{i-1,i} = \{d \in \boldsymbol{\mathcal{D}} \quad | \quad level(d) = i\}$$

Proof:

Let us consider d in $\boldsymbol{\mathcal{D}}$ such that $level(d) = i \leq n$. By definition of the function $level$, d belongs to $\mathcal{SD}_{i-1}$ and $d \notin \mathcal{SD}_i$. Since $\mathcal{SD}_i = \mathcal{SD}_{i-1} - K_{i-1,i}$ d must belong to $K_{i-1,i}$.

Conversely, if d belongs to $K_{i-1,i}$ we have, $d \in \mathcal{SD}_{i-1}$ and $d \notin \mathcal{SD}_i$. Moreover, since $\mathcal{SD}_k \subset \mathcal{SD}_i$ for each k greater than i, $d \notin \mathcal{SD}_k$ for $k \geq i$. We thus obtain $level(d) = i$. □

Corollary 1 *Given a combinatorial map $G_0 = (\boldsymbol{\mathcal{D}}, \sigma, \alpha)$ and the pyramid construction plan $\mathcal{LP} = (G_0, level)$. Each contraction kernel K_{0i} is equal to the darts having a level less than or equal to i:*

$$\forall i \in \{1, \ldots, n\} \quad K_{0i} = \{d \in \boldsymbol{\mathcal{D}} \quad | \quad level(d) \leq i\}$$

Proof:

We have for each level i in $\{1, \ldots, n\}$:

$$K_{0i} = \bigcup_{j=1}^{i} K_{j-1,j}$$

Using proposition 5 we obtain:

$$K_{0i} = \{d \in \boldsymbol{\mathcal{D}} \quad | \quad level(d) \leq i\}$$

□

Corollary 2 *Given a combinatorial map $G_0 = (\boldsymbol{\mathcal{D}}, \sigma, \alpha)$ and the pyramid construction plan, $\mathcal{LP} = (G_0, level)$ defined by n contraction kernels. The surviving darts of the i^{th} contraction kernel have a level strictly greater than i:*

$$\forall i \in \{1, \ldots, n\} \quad \mathcal{SD}_i = \{d \in \boldsymbol{\mathcal{D}} \quad | \quad level(d) > i\}$$

Proof:

The surviving darts of level i are defined by:

$$\mathcal{SD}_i = \boldsymbol{\mathcal{D}} - K_{0i}$$

Since $K_{0i} = \{d \in \boldsymbol{\mathcal{D}} \quad | \quad level(d) \leq i\}$ (see corollary 1) we have:

$$\mathcal{SD}_i = \{d \in \boldsymbol{\mathcal{D}} \quad | \quad level(d) > i\}$$

□

Remark 1 *Given a pyramid construction plan $\mathcal{LP} = (G_0, level)$ defined by an initial combinatorial map G_0 and n inclusion kernels, a dart $d \in \mathcal{D}$ such that $level(d) = n + 1$ belongs to $\mathcal{SD}_n$. Therefore, this dart is not contracted during the sequence of contractions generating the pyramid.*

Proposition 5, Corollaries 1 and 2 show that the function level allows us to retrieve the different contraction kernels and their associated surviving darts. The permutation α being the same for all contracted combinatorial maps, a given contracted map $G_i = (\mathcal{SD}_i, \sigma_i, \alpha)$ will be completely determined if we can define the permutation σ_i from the function level. Propositions below show that Algorithm 1 allows us to retrieve the different permutations σ_i thanks to the implicit encoding of the contraction kernels K_{0i} by the function *level*. It can be considered the 'life time' of a dart in the sequence of contractions generating the pyramid.

```
dart survive_C(int i, dart d)
{
    if ( level(d) > i )
        return d;

    return survive_C(i,φ(d))
}
```

Algorithm 1: *The function $survive_C$ returns the first dart in $\mathcal{SD}_i$ encountered when turning around the face $\varphi^*(d)$*

Definition 4 survive$_C$ Stack *Given a combinatorial map $G_0 = (\mathcal{D}, \sigma, \alpha)$, and the pyramid construction plan, $\mathcal{LP} = (G_0, level)$. The ordered set $Stack_C(i, d)$ is the sequence of darts which will be passed as second argument of the recursive function $survive_C$ during a call to $survive_C(i, d)$.*

Remark 2 *Using the same notations and hypothesis as definition 4, the last dart of $Stack_C(i, d)$ is equal to $survive_C(i, d)$.*

Proposition 6 *Using the same notations and hypothesis as definition 4, the ordered set $Stack_C(i, \varphi(d))$ is equal to $CW_{0i}(d) - \{d\}$ concatenated with*

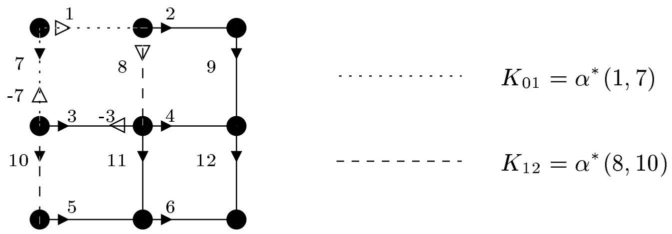


Figure 4: *An illustration of the algorithm* $survive_C$*. A call to* $survive_C(2,-7)$ *will induce the traversal of the darts* $-7, 1, 8$ *and* -3 *represented by empty arrows. Note that we have* $CW_{02}(-3) = -3, -7, 1, 8$. *and thus* $\varphi_2(-3) = \varphi(8) = -3$ *(see Figure 3)*

$follow_{0i}(d)$ *for each* i *in* $\{1, \ldots, n\}$ *and each* d *in* $\mathcal{SD}_i$*:*

$$\left.\begin{array}{lll}\forall & i & \in & \{1,\ldots,n\} \\ \forall & d & \in & \mathcal{SD}_i\end{array}\right\} \quad d \cdot Stack_C(i, \varphi(d)) = CW_{0i}(d) \cdot follow_{0i}(d)$$

Proof:

- If $CW_{0i}(d) = (d)$, then $\varphi(d)$ belongs to $\mathcal{SD}_i$. In this case $level(\varphi(d))$ is strictly greater than i (see Corollary 2) and $survive_C(i, \varphi(d))$ is equal to $\varphi(d)$. Therefore:

$$Stack_C(i, \varphi(d)) = (\varphi(d))$$

  with $\varphi(d) = follow_{0i}(d)$.

- Let $CW_{0i}(d) = (d, d_1, \ldots, d_p)$ with $p \geq 1$. Suppose that the series $Stack_C(i, \varphi(d))$ and $CW_{0i}(d) - \{d\}$ are equal until a given index j:

$$Stack_C(i, \varphi(d)) = (d_1, \ldots, d_j, \ldots)$$

  We have by definition of a connecting walk $d_k = \varphi^k(d)$ for each k in $\{1, \ldots, p\}$. Thus $d_1 = \varphi(d)$ belongs simultaneously to $Stack_C(i, \varphi(d))$ and $CW_{0i}(d) - \{d\}$ and the property is true for $j = 1$. If the property is true until a given rank $j < p$, we have: $d_{j+1} = \varphi(d_j)$ and $d_j \in K_i$. Therefore, the level of d_j is less than i (see corollary 1) and $survive_C(i, d_j) = survive_C(i, d_{j+1})$. Thus d_{j+1} belongs to $Stack_C(i, \varphi(d))$

and follows d_j in this order. The property is thus true until the rank $j+1$.

The sequence $(d_1, \ldots, d_p)$ is thus included in $Stack_C(i, \varphi(d))$. Moreover, by definition of a connecting walk $\varphi(d_p) = follow_{0i}(d) \in \mathcal{SD}_i$. Therefore, the level of $\varphi(d_p)$ is strictly greater than i and $survive_C(i, d_p) = survive_C(i, \varphi(d_p)) = \varphi(d_p) = follow_{0i}(d)$. Thus the sequence of recursive calls to the function $survive_C$ stops on $\varphi(d_p)$ and $Stack_C(i, \varphi(d))$ is equal to:

$$Stack_C(i, \varphi(d)) = (d_1, \ldots, d_p, follow_{0i}(d))$$

□

Corollary 3 *Given a combinatorial map $G_0 = (\mathcal{D}, \sigma, \alpha)$ and a pyramid construction plan, $\mathcal{LP} = (G_0, level)$ defined by n inclusion kernels. The application $follow_{0i}$ defined by G_0 and the contraction kernel K_{0i} may be retrieved with the function $survive_C$ by using the following equation:*

$$\left.\begin{array}{ccc} \forall & i & \in & \{1, \ldots, n\} \\ \forall & d & \in & \mathcal{SD}_i \end{array}\right\} \quad follow_{0i}(d) = survive_C(i, \varphi(d))$$

Proof:

See proposition 6 and remark 2. □

A combinatorial map $G = (\mathcal{D}, \sigma, \alpha)$ is explicitly encoded by its set of darts $\mathcal{D}$ and the two permutations σ and α. Corollary 3 shows us that the function $follow_{0i}$, and thus σ_i may be retrieved thanks to the algorithm $survive_C$. However, the pyramid construction plan remains implicitly defined by function *level*. One idea to obtain a more explicit form of the pyramid construction plan is to store the results of the function $survive_C$ in a function Σ_C (see Table 1) which then encodes explicitly the pyramid construction plan:

Definition 5 Application Σ_C

Given a combinatorial map $G_0 = (\mathcal{D}, \sigma, \alpha)$, and a pyramid construction plan, $\mathcal{LP} = (G_0, level)$ defined by n inclusion kernels. The application Σ_C from $\{1, \ldots, n\} \times \mathcal{D}$ to $\mathcal{D}$ is defined by:

$$\Sigma_C \left(\begin{array}{ccc} \{1, \ldots, n\} \times \mathcal{D} & \rightarrow & \mathcal{D} \\ (i, d) & \mapsto & survive(i, \sigma(d)) \end{array} \right.$$

Definition 6 Restoration of the pyramid construction plan

Given a combinatorial map $G_0 = (\boldsymbol{\mathcal{D}}, \sigma, \alpha)$*, and a pyramid construction plan* $\mathcal{LP} = (G_0, level)$ *defined by a sequence of n inclusion kernels, the restoration* p_i *is an application from* $\mathcal{SD}_i$ *to* $\{1, \ldots, n\} \times \boldsymbol{\mathcal{D}}$ *which associates to each dart d in* $\mathcal{SD}_i$ *the couple* (i, d)*:*

$$p_i \left(\begin{array}{lll} \mathcal{SD}_i & \rightarrow & \{1, \ldots, n\} \times \boldsymbol{\mathcal{D}} \\ d & \mapsto & (i, d) \end{array} \right.$$

Proposition 7 *Given a combinatorial map* $G_0 = (\boldsymbol{\mathcal{D}}, \sigma, \alpha)$*, a pyramid construction plan,* $\mathcal{LP} = (G_0, level)$ *defined by n inclusion kernels and the restoration* $(p_1, \ldots, p_n)$*.*

The permutation σ_i *of* $G_i = (\mathcal{SD}_i, \sigma_i, \alpha)$ *is equal to* Σ_C *composed with* p_i*.*

$$\forall i \in \{1, \ldots, n\} \quad \Sigma_C \circ p_i = \sigma_i$$

Proof:

Given an index i in $\{1, \ldots, n\}$ and a dart d in $\mathcal{SD}_i$, the application $\Sigma_C \circ p_i$ maps d into $survive_C(i, \sigma(d))$. Thus, using Corollary 3 we have:

$$\Sigma_C \circ p_i(d) = survive(i, \sigma(d)) = survive(i, \varphi(\alpha(d))) = follow_{0i}(\alpha(d))$$

Using the isomorphism between the connecting walk map and the contracted one (see [6]) we have: $\sigma_i = follow_i \circ \alpha$. Therefore:

$$\forall i \in \{1, \ldots, n\} \quad \forall d \in \mathcal{SD}_i \quad \Sigma_C \circ p_i(d) = \sigma_i(d)$$

□

Thus, using the function *level* and the algorithm $survive_C$ we can retrieve, all the contraction kernels and all the contracted combinatorial maps defined by these kernels. If the result of the function $survive_C(i, \sigma(d))$ is stored in Σ_C for each level i and for each dart d in $\mathcal{SD}_i$, the implicit definition of the pyramid construction plan becomes explicit and it can be denoted by: $\mathcal{LP} = (G_0, level) = (\boldsymbol{\mathcal{D}}, \Sigma_C, level, \alpha)$ (see Table 1).

If the sequence of contractions is defined by successive contraction kernels, instead of inclusion kernels, the different contraction kernels may be retrieved from the function *level* by the proposition 5. This proposition may be used as a consequence of Definition 3 if the pyramid is defined by inclusion

d	$max_i + 1$	$\Sigma_C(i,d)$		
dart	level	0	1	2
1	1	7		
-1	1	8		
2	3	-1	10	5
-2	3	9	9	9
3	3	-7	8	-3
-3	3	11	11	11
4	3	-8	-8	2
-4	3	12	12	12
5	3	-10	-10	3
-5	3	6	6	6
6	3	-11	-11	-11
-6	3	-12	-12	-12
7	1	1		
-7	1	10		
8	2	2	2	
-8	2	-3	-3	
9	3	-2	-2	-2
-9	3	-4	-4	-4
10	2	3	3	
-10	2	5	5	
11	3	4	4	4
-11	3	-5	-5	-5
12	3	-9	-9	-9
-12	3	-6	-6	-6

Table 1: This table represents a possible implementation of the function Σ_C by a bi-dimensional array with lines of variable size. Note that in this case the function *level* encode simultaneously the level on which a dart is contracted and the size of its associated line. The different values of $\Sigma_C(i,d)$ given in this table correspond to the contractions defined in Figure 3

kernels or as a definition of the function *level* if the pyramid is defined by successive kernels. However, note that in this case, the uncontracted darts which belongs to $\boldsymbol{\mathcal{D}} - K_{0n}$ must be labeled with $n+1$.

The basic idea of a parallel implementation of the function $survive_C$ is to run the algorithm $survive_C$ concurrently on $|\boldsymbol{\mathcal{D}}|$ processors. If we suppose that we have an ideal CREW PRAM (Concurrent Read Exclusive Write Parallel Random Access Machine, [10]) the parallel algorithm consists to initialize a linear array *survive* of darts to the identity and to determine for each dart its next surviving dart within the face (see algorithm 2). This computation being performed concurrently.

```
void set_next_survivor(int i)
{
    For each d in 𝓓 do in parallel
        survive[d] = d;
    For each d in 𝓓 do in parallel
        get_survive(i,d);
}
```

Algorithm 2: *The algorithm set_next_survivor computes the next survivor of each dart. The function get_survive is described in Algorithm 3*

```
void get_survive(int i, dart d)
{
    while(level(survive[d]) ≤ i )
    {
        survive[d]=survive[φ(d)];

    }
}
```

Algorithm 3: *The algorithm get_survive is attached to one dart and computes its next survivor*

If we suppose that a face is defined by $\varphi^*(1) = (1, 2, 3, 4, 5, 6, 7, 8, 9)$, and that the surviving darts of this faces are 2 and 5, the algorithm get_survive will produce the following steps:

$$
survive : \left(\begin{array}{c|c|ccc|cccc}
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\
2 & 2 & 4 & 5 & 5 & 7 & 8 & 9 & 1 \\
2 & 2 & 5 & 5 & 5 & 8 & 9 & 1 & 2 \\
2 & 2 & 5 & 5 & 5 & 9 & 1 & 2 & 2 \\
2 & 2 & 5 & 5 & 5 & 1 & 2 & 2 & 2 \\
2 & 2 & 5 & 5 & 5 & 2 & 2 & 2 & 2
\end{array} \right.
$$

Using a PRAM model, each elementary operation is performed synchronously. Therefore, the number of elementary steps of each algorithm $get\_survive(i, d)$ is equal to the cyclic distance between d and its associated surviving dart . If we denote by D the maximum of these distances, the parallel algorithm will terminate after D steps. Therefore, worst case parallel complexity of our algorithm is linear in the cyclic max-distance between surviving darts. Moreover, using Brent's Lemma [2] our algorithm may be executed on a PRAM machine with p processors in:

$$t(p) \leq D + \frac{|\boldsymbol{\mathcal{D}}| + |\mathcal{SD}_i| - D}{p} \text{ steps.}$$

2.2 Adaptations from removal kernels

Since a single contraction of a combinatorial map does not change the number of (dual) faces also a sequence of contractions cannot remove a face. Thus using contraction kernels solely, at least one dart must survive in each face (see [6]). Therefore, we cannot contract a complex combinatorial map into a self-loop by contractions solely. Consequently also contractions of the dual combinatorial map, i.e. removals, must be considered. The reduction of the initial combinatorial map to a self-loop or to a combinatorial map with less faces than the original needs both contractions and removals. In the following we study some properties of the removal operation.

Definition 7 Removal Kernel
Given a combinatorial map G , a removal kernel is a contraction kernel of $\overline{G}$.

Note that a contraction operation on $\overline{G}$ is equivalent to a removal operation on G and vice-versa [15]. Moreover, a removal kernel is a forest of

$\overline{G}$. Therefore if K is a contraction kernel of G, its dual $\overline{K}$ is not necessarily a contraction kernel of the dual map $\overline{G}$. Thus, if we talk about a *removal kernel* below, we mean an inclusion kernel defined in the dual combinatorial map $\overline{G}$, and not the dual of an inclusion kernel of G.

Since a dual inclusion kernel is defined in the dual combinatorial map $\overline{G}$, the permutations σ and φ have to change their roles. Therefore, the definition of a connecting walk has to be changed accordingly(see Definition 1):

Definition 8 Dual connecting Walk

Given an initial connected combinatorial map $G_0 = (\mathcal{D}, \sigma, \alpha)$ and a removal kernel K_{ij} defined on $\overline{G_0}$, we associate to each dart d of $\mathcal{SD}_j$ a **dual connecting walk** *$DCW_{ij}(d)$ defined on $\mathcal{SD}_i$ by:*

$$DCW_{ij}(d) = (d, \sigma_i(d), \ldots, \sigma_i^{n-1}(d)) \text{ with } n = Min\{p \in \mathbb{N}^* \mid \sigma_i^p(d) \in \mathcal{SD}_j\}$$

A removal kernel is simply a contraction kernel defined in the dual combinatorial map. Therefore, given a combinatorial map G, all the properties in G shown in TR-57 [6] for inclusion or successor kernels remain valid in $\overline{G}$ for inclusion or successor removal kernels defined in $\overline{G}$.

One such property of a removal is the preservation of structure. We want that any surviving part remains connected or disconnected after removal. It has been shown in [11] that parallel edges or self-loops can be removed without destroying the structure if the enclosed face has a degree less than three. This criterion generates automatically removal kernels that 'clean' the original map from redundant parallel edges and self-loops.

A pyramid construction plan may be defined by a sequence of contractions or by a sequence of removals. Definition 3, Proposition 5 and Corollary 1 and 2, remain valid in both cases.

However, if we use a sequence of removals, the function $survive_C$ has to be adapted in order to traverse the dual connecting walks (see Algorithm 4). Moreover, it can be easily shown that the function $survive_R$ defined by Algorithm 4 is a transcription of the algorithm $survive_C$ in the dual graph. Therefore, all the results given in the previous section may be adapted and are given here without demonstration.

Definition 9 survive$_R$ Stack

Given a combinatorial map $G_0 = (\mathcal{D}, \sigma, \alpha)$, a sequence of removal inclusion kernels[2] $K_{01} \subset K_{02} \ldots \subset K_{0n}$ and their associated pyramid construction

[2]Note that K_{0i} must be a spanning forest of $\overline{G_0} = (\mathcal{D}, \varphi, \alpha)$.

```
dart survive_R(int i, dart d)
{
    if ( level(d) > i )
        return d;

    return survive_DC(i,σ(d))
}
```

Algorithm 4: *The function survive for a removal kernel*

plan, $\mathcal{LP} = (G_0, level)$. The ordered set $Stack_{DC}(i,d)$ is the sequence of darts which will be passed as second argument of the recursive function $survive_R$ during a call to $survive_R(i,d)$.

Remark 3 *Using the same notations and hypothesis as definition 9, the last dart of $Stack_R(i,d)$ is equal to $survive_R(i,d)$.*

Proposition 8 *Using the same notations and hypothesis as definition 9, the ordered set $Stack_R(i,\sigma(d))$ is equal to $DCW_{0i}(d) - \{d\}$ concatenated with $\overline{follow_{0i}}(d)$ for each i in $\{1,\ldots,n\}$ and each d in $\mathcal{SD}_i$.*

$$\left.\begin{array}{ccc} \forall & i & \in & \{1,\ldots,n\} \\ \forall & d & \in & \mathcal{SD}_i \end{array}\right\} \quad d.Stack_R(i,\sigma(d)) = DCW_{0i}(d) \cdot \overline{follow_{0i}}(d))$$

Where $DCW_{0i}(d)$ is the connecting walk of d defined by K_{0i} on $\overline{G_0}$. And $\overline{follow_{0i}}$ is the application $follow_{0i}$ defined on $\overline{G_0}$ by the removal kernel K_{0i}.

Corollary 4 *Given a combinatorial map $G_0 = (\boldsymbol{\mathcal{D}}, \sigma, \alpha)$ and a pyramid construction plan $\mathcal{LP} = (G_0, level)$ defined by n removal kernels. The application $\overline{follow_{0i}}$ defined by $\overline{G_0}$ and the removal kernel K_{0i} is equal to:*

$$\forall i \in \{1,\ldots,n\} \quad \forall d \in \mathcal{SD}_i \quad \overline{follow_{0i}}(d) = survive_R(i,\sigma(d))$$

Definition 10 Application Σ_R

Given a combinatorial map $G_0 = (\boldsymbol{\mathcal{D}}, \sigma, \alpha)$ and a pyramid construction plan $\mathcal{LP} = (G_0, level)$ defined by n dual inclusion kernels. The application Σ_R from $\{1,\ldots,n\} \times \boldsymbol{\mathcal{D}}$ to $\boldsymbol{\mathcal{D}}$ is defined by:

$$\Sigma_R \left(\begin{array}{ccc} \{1,\ldots,n\} \times \boldsymbol{\mathcal{D}} & \rightarrow & \boldsymbol{\mathcal{D}} \\ (i,d) & \mapsto & survive_R(i,\sigma(d)) \end{array} \right.$$

Proposition 9 *Given a combinatorial map* $G_0 = (\boldsymbol{\mathcal{D}}, \sigma, \alpha)$ *and a pyramid construction plan* $\mathcal{LP} = (G_0, level)$ *defined by n dual inclusion kernels. The application* $\Sigma_R \circ p_i$ *is equal to the permutation* σ_i *of the* i^{th} *removed combinatorial map.*

$$\forall i \in \{1, \ldots, n\} \quad \Sigma_R \circ p_i = \sigma_i$$

Proof:

According to the definitions of application Σ_R and p_i the composition of both applications is equal to $survive_R(i, \sigma(d))$ for each i in $\{1, \ldots, n\}$ and each d in $\mathcal{SD}_i$:

$$\forall i \in \{1, \ldots, n\} \forall d \in \mathcal{SD}_i \quad \Sigma_R \circ p_i(d) = survive_R(i, \sigma(d))$$

Using Corollary 4:

$$\Sigma_R \circ p_i(d) = survive_R(i, \sigma(d)) = \overline{follow_i}(d)$$

Using the isomorphism between the dual connecting walk map and the removed one (see [6]) we have: $\sigma_i = \overline{follow_{0i}}$, therefore:

$$\forall i \in \{1, \ldots, n\} \forall d \in \mathcal{SD}_i \quad \Sigma_R \circ p_i(d) = \sigma_i(d)$$

□

3 Generalized Pyramid Construction Plans

In this section we consider a sequence of n successive kernels such that each kernel performs either contractions or removals of a set of darts of the current combinatorial map. According to the definition of a contraction kernel, the set of darts contracted or removed at level i, is a forest of G_{i-1} if $K_{i-1,i}$ is a contraction kernel and a forest of $\overline{G_{i-1}}$ if $K_{i-1,i}$ is a removal kernel. Therefore, the successive application of two successive kernels $K_{i-1,i}$ and $K_{i,i+1}$ is neither a forest of G_{i-1} nor of $\overline{G_{i-1}}$ if $K_{i-1,i}$ and $K_{i,i+1}$ do not perform the same type of contraction. Since we no longer consider contraction kernels $K_{i,j}$ with $j \neq i+1$ we simplify the notations by denoting $K_{i-1,i}$ by K_i.

Proposition 10 *Given a sequence of contraction or removal kernels* $(K_1, \ldots, K_n)$ *successively applied on the initial combinatorial map* $G_0 = (\boldsymbol{\mathcal{D}}, \sigma, \alpha)$*, the two following properties hold:*

$$\begin{cases} \forall i \in \{1, \ldots, n\} & \mathcal{SD}_i = \boldsymbol{\mathcal{D}} - \bigcup_{j=1}^{i} K_j \\ \forall (i,j) \in \{1, \ldots, n\}^2, i \neq j & K_i \cap K_j = \emptyset \end{cases}$$

Proof:

Let us demonstrate the first property by recurrence. We have, by definition of a contraction kernel $\mathcal{SD}_1 = \boldsymbol{\mathcal{D}} - K_1$. Note that, this property holds also if K_1 is a removal kernel. Let us suppose the property is true until a given rank $i < n$. Then we have by definition of a kernel:

$$\mathcal{SD}_i = \mathcal{SD}_{i-1} - K_i = \boldsymbol{\mathcal{D}} - \bigcup_{j=1}^{i-1} K_j - K_i = \boldsymbol{\mathcal{D}} - \bigcup_{j=1}^{i} K_j$$

Let us now suppose that we can find a dart d belonging to two kernels K_i and K_j with $i < j$. In order to be contracted by K_j, d must belong to $\mathcal{SD}_{j-1}$ with :

$$\mathcal{SD}_{j-1} = \boldsymbol{\mathcal{D}} - \bigcup_{k=1}^{j-1} K_k$$

Since $K_i \subset \bigcup_{k=1}^{j-1} K_k$ we obtain the desired contradiction. □

3.1 Connecting dart sequences

In the following we will have to distinguish two cases:

1. When two successive kernels K_i and K_{i+1} are both contraction kernels or both removal kernels.

2. When K_i is a contraction kernel and K_{i+1} a removal one or when K_i is a removal kernel and K_{i+1} a contraction kernel.

In order to simplify the notations, we will say that two successive kernels K_i and K_{i+1} have the same type in the first case and different types in the second one. More generally the type of a contraction kernel refers to the combinatorial map on which it is applied (the initial or the dual one). Figure 5 shows a sequence of combinatorial maps built by contraction and removal kernels applied alternatively.

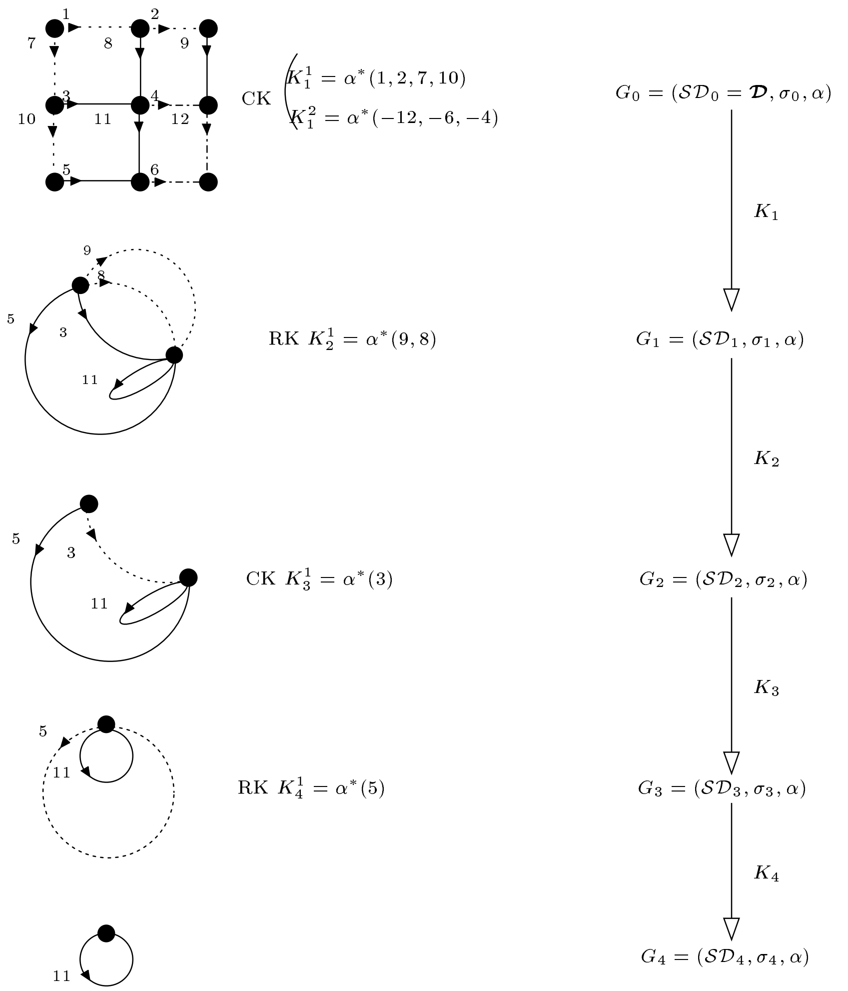


Figure 5: *The initial grid encoded by the initial combinatorial map* $G_0 = (\boldsymbol{\mathcal{D}}, \sigma, \alpha)$ *is successively contracted by* K_1*,* K_2*,* K_3 *and* K_4*. Kernels with even indices denote contraction kernels (CK) while odd indices denote removal kernels(RK).*

Definition 11 Connecting Dart Sequences

Given a combinatorial map $G_0 = (\mathcal{D}, \sigma, \alpha)$ and a sequence of contraction or removal kernels $K_1, K_2 \ldots, K_n$. The connecting dart sequence is defined by the following recursive construction:

$$\forall d \in \mathcal{D} \quad CDS_0(d) = d$$

For each level i in $\{1, \ldots, n\}$ and for each dart d in $\mathcal{SD}_i$

- *If K_i and K_{i-1} have the same type:*

$$CDS_i(d) = CDS_{i-1}(d_1) \cdots CDS_{i-1}(d_p)$$

- *If K_i and K_{i-1} have different types:*

$$CDS_i(d) = d_1 \cdot CDS^*_{i-1}(\alpha(d_1)) \cdots d_p \cdot CDS^*_{i-1}(\alpha(d_p))$$

Where $(d_1 \ldots d_p)$ is equal to $CW_{i-1,i}(d)$ if K_i is a contraction kernel and $DCW_{i-1,i}(d)$ if K_i is a removal kernel. The term $CDS^*_{i-1}(\alpha(d_j))$ denotes the connecting dart sequence $CDS_{i-1}(\alpha(d_j))$ without its first dart. The kernels $K_0 = \emptyset$ and K_1 have the same type by convention.

The set of connecting dart sequences associated to the kernels defined in Figure 5 are given in Tables 2, 3, 4 and 5 (see section A).

Proposition 11 *Given a combinatorial map $G_0 = (\mathcal{D}, \sigma, \alpha)$ and a sequence of contraction kernels $K_1, K_2 \ldots, K_n$. We can define a sequence of inclusion kernels $K_{01}, \ldots, K_{0n}$ with $K_{0i} = \cup_{j=1}^{i} K_i$ providing the same contracted combinatorial maps. Moreover, in this case the connecting dart sequences are equal to the connecting walks defined on the kernels $K_{01}, \ldots K_{0n}$:*

$$\forall i \in \{1, \ldots, n\} \quad \forall d \in \mathcal{SD}_i \quad CDS_i(d) = CW_{0i}(d)$$

Where $CW_{0i}(d)$ is defined by K_{0i} on G_0.

Proof:

First note that, if all the contraction kernels have the same type, the connecting dart sequences are defined by:

$$\forall d \in \mathcal{SD}_i \quad CDS_i(d) = CDS_{i-1}(d_1), \cdots, CDS_{i-1}(d_p)$$

With $CW_i(d) = d_1, \ldots, d_p$ is the connecting walk of d defined on G_{i-1} by K_i.

The proposition is trivially true for $i = 1$. Indeed, in this case:

$$\forall d \in \mathcal{SD}_1 \quad \begin{cases} CDS_1(d) &= CDS_0(d_1)\cdots CDS(d_p) \\ CDS_1(d) &= (d_1 \ldots d_p) = CW_{01}(d) \end{cases}$$

Let us suppose that this proposition is true until a given rank i. Since K_i and K_{i+1} have the same type, we have for each dart d in $\mathcal{SD}_{i+1}$:

$$\begin{aligned} CDS_{i+1}(d) &= CDS_i(d'_1)\cdots CDS_i(d'_q) \\ &= CW_{0i}(d'_1)\cdots CW_{0i}(d'_q) \quad \text{(by our recurrence hypothesis)} \end{aligned}$$

With $CW'_{i+1}(d) = (d'_1 \ldots d'_q)$ denotes the connecting walk of d defined by K_{i+1} on G_i.

Moreover, we have by proposition 4:

$$CW_{0,i+1}(d) = CW_{0,i}(d'_1)\cdots CW_{0,i}(d'_q)$$

Therefore, $CDS_{i+1}(d) = CW_{0,i+1}(d)$ and the recurrence hypothesis holds until $i + 1$. $\square$

Remark 4 *Note that the demonstration of proposition 11 remains valid if all kernels are removal ones, therefore:*

Given a combinatorial map $G_0 = (\boldsymbol{\mathcal{D}}, \sigma, \alpha)$ and a sequence of removal kernels $K_1, K_2 \ldots, K_n$. We can define a sequence of removal inclusion kernels $K_{01}, \ldots, K_{0n}$ with $K_{0i} = \cup_{j=1}^{i} K_i$ providing the same combinatorial maps. In this case the connecting dart sequences are equal to the dual connecting walks defined on the kernels $K_{01}, \ldots K_{0n}$:

$$\forall i \in \{1, \ldots, n\} \quad \forall d \in \mathcal{SD}_i \quad CDS_i(d) = DCW_{0i}(d)$$

Where $DCW_{0i}(d)$ is defined by K_{0i} on $\overline{G_0}$.

In the following, we will have to consider connecting walks or dual connecting walks according to the type of the associated kernel. All the properties of connecting walks used below are common to connecting walks and dual connecting walks. In order to not overload the demonstrations, we will denote both connecting walks $CW_{i-1,i}$ and dual connecting walks $DCW_{i-1,i}$ by CW_i. The type of the connecting walk is then implicitly defined by the type of its associated kernel K_i.

Proposition 12 *Given a combinatorial map $G_0 = (\boldsymbol{\mathcal{D}}, \sigma, \alpha)$ and a sequence of contraction or removal kernels $K_1, K_2 \ldots, K_n$. For any level i in $\{1, \ldots, n\}$, the first dart of $CDS_i(d)$ with d in $\mathcal{SD}_i$ is d:*

$$\left.\begin{array}{ccc} \forall & i & \in & \{1, \ldots, n\} \\ \forall & d & \in & \mathcal{SD}_i \end{array}\right\} \quad CDS_i(d) = (d, d_2, \ldots, d_p)$$

Proof:

The proposition is trivial for $i = 0$ (we have in this case $p = 0$). Let us suppose it is true until a given rank $i - 1 < n$ and let us consider a given dart d in $\mathcal{SD}_i$ with:

$$CW_i(d) = (d_1, \ldots, d_p)$$

Note that according to the definition of a connecting walk (see Def. 1) we have $d_1 = d$.

Then, if K_i and K_{i-1} have the same type:

$$CDS_i(d) = CDS_{i-1}(d_1) \cdots CDS_{i-1}(d_p)$$

Since d_1 is equal to d the proposition is true at rank i thanks to our recurrence hypothesis.

If K_i and K_{i-1} have a different type:

$$CDS_i(d) = d_1 \cdot CDS^*_{i-1}(\alpha(d_1)) \cdots d_p \cdot CDS^*_{i-1}(\alpha(d_p))$$

Like previously, we have $d_1 = d$ by definition of a connecting walk, and the first dart of $CDS_i(d)$ is thus equal to d. $\square$

Proposition 13 *Given a combinatorial map $G_0 = (\boldsymbol{\mathcal{D}}, \sigma, \alpha)$ and a sequence of contraction or removal kernels $K_1, K_2, \ldots, K_n$. For any level i in $\{1, \ldots, n\}$ and for any connecting dart sequence $CDS_i(d)$ with d in $\mathcal{SD}_{i-1}$, the sequence $CDS^*_i(d)$ is included in $\cup_{j=0}^{i} K_j$:*

$$\forall i \in \{0, \ldots, n\} \quad \forall d \in \mathcal{SD}_i \quad CDS^*_i(d) \subset \bigcup_{j=0}^{i} K_j$$

Proof:

If $i = 0$, $CDS_0(d) = d$ therefore:

$$\forall d \in \boldsymbol{\mathcal{D}} \quad CDS^*_0(d) = \emptyset \subset K_0 = \emptyset$$

The proposition is thus trivial for $i = 0$. Let us suppose it is true until a given rank $i - 1$.

Let us consider a given d in $\mathcal{SD}_i$ such that:

$$CW_i(d) = (d_1, \ldots, d_p)$$

with $d_1 = d$ and $(d_2, \ldots, d_p) \subset K_i$.

- If K_i and K_{i-1} have the same type:

  $$CDS_i(d) = CDS_{i-1}(d_1) \cdots CDS_{i-1}(d_p)$$

  Using our recurrence hypothesis,

  $$\forall j \in \{1, \ldots, p\} \quad CDS^*_{i-1}(d_j) \subset \bigcup_{k=0}^{i-1} K_k$$

  and the fact that all darts $d_j, 1 < j \leq p$, of the connecting walk $CW_i(d)$ belong to K_i we have:

  $$\begin{array}{lll} & CDS^*_{i-1}(d) & \subset \bigcup_{k=0}^{i-1} K_k \\ \forall j \in \{2, \ldots, p\} & CDS_{i-1}(d_j) & \subset \bigcup_{k=0}^{i} K_k \end{array}$$

  Therefore,

  $$CDS^*_i(d) = CDS^*_{i-1}(d) \cdot CDS_{i-1}(d_2) \cdots CDS_{i-1}(d_p) \subset \bigcup_{k=0}^{i} K_k$$

- If K_i and K_{i-1} have not the same type:

  $$CDS_i(d) = d_1 \cdot CDS^*_{i-1}(\alpha(d_1)) \cdots d_p \cdot CDS^*_{i-1}(\alpha(d_p))$$

  Like previously, using our recurrence hypothesis:

  $$\begin{array}{lll} & CDS^*_{i-1}(\alpha(d)) & \subset \bigcup_{k=0}^{i-1} K_k \\ \forall j \in \{2, \ldots, p\} & d_j \cdot CDS^*_{i-1}(\alpha(d_j)) & \subset \bigcup_{k=0}^{i} K_k \end{array}$$

  Therefore,

  $$CDS^*_i(d) = CDS^*_{i-1}(\alpha(d_1)) \cdots d_p \cdot CDS^*_{i-1}(\alpha(d_p)) \subset \bigcup_{k=0}^{i} K_k$$

□

Proposition 14 *Given a combinatorial map $G_0 = (\boldsymbol{\mathcal{D}}, \sigma, \alpha)$ and a sequence of contraction or removal kernels $K_1, K_2 \dots, K_n$. Any connecting dart sequence at level i, with $i \in \{1, \dots, n\}$ is not shorter than the corresponding connecting walk:*

$$\forall i \in \{1, \dots, n\} \quad \forall d \in \mathcal{SD}_i \quad |CDS_i(d)| \geq |CW_i(d)|$$

Proof:

Given an index i in $\{1, \dots, n\}$, and a dart d such that:

$$CW_i(d) = (d_1, \dots, d_p)$$

If K_i and K_{i-1} have the same type,

$$CDS_i(d) = CDS_{i-1}(d_1) \cdots CDS_{i-1}(d_p)$$

Since each connecting dart sequence $CDS_{i-1}(d_j)$ with j in $\{1, \dots, p\}$ contains at least d_j (see proposition 12), we have $|CDS_i(d)| \geq |CW_i(d)|$. If K_i and K_{i-1} have not the same type, we have:

$$CDS_i(d) = d_1 \cdot CDS^*_{i-1}(\alpha(d_1)) \dots d_p \cdot CDS^*_{i-1}(\alpha(d_p))$$

The proposition is then trivial. □

Proposition 15 *Given a combinatorial map $G_0 = (\boldsymbol{\mathcal{D}}, \sigma, \alpha)$ and a sequence of contraction or removal kernels $K_1, K_2 \dots, K_n$. If a connecting dart sequence $CDS_i(d)$ is reduced to d then $\varphi_i(d) = \varphi(d)$ if K_i is a contraction kernel, and $\sigma_i(d) = \sigma(d)$ if K_i is a removal kernel:*

$$CDS_i(d) = (d) \Longrightarrow \begin{cases} \varphi_i(d) &= \varphi(d) \quad \textit{If } K_i \textit{ is a contraction kernel} \\ \sigma_i(d) &= \sigma(d) \quad \textit{If } K_i \textit{ is a removal kernel} \end{cases}$$

Proof:

If $i = 1$, $CDS_1(d) = CW_1(d)$ for any d in $\mathcal{SD}_1$. Then if the cardinal of $CDS_1(d)$ is reduced to 1, the cardinal of $CW_1(d)$ must also be equal to 1 (see proposition 14). We have thus in this case $CDS_1(d) = CW_1(d) = (d)$. Therefore, using the definition of a connecting walk:

- $\varphi_1(d) = \varphi(d)$ if K_1 is a contraction kernel.
- $\sigma_1(d) = \sigma(d)$ if K_1 is a removal kernel.

The property is thus true at rank 1 for any dart in $\mathcal{SD}_1$. Let us suppose the property is true until rank $i-1 < n$. Then, if $CDS_i(d) = (d)$, we have as previously, $CW_i(d) = (d)$. Therefore:

$$\begin{array}{rcll} \varphi_i(d) & = & \varphi_{i-1}(d) & \text{if } K_i \text{ is a contraction kernel} \\ \sigma_i(d) & = & \sigma_{i-1}(d) & \text{if } K_i \text{ is a removal kernel} \end{array} \tag{2}$$

Moreover, if K_i and K_{i-1} have the same type:

$$CDS_i(d) = CDS_{i-1}(d) = (d)$$

using the recurrence hypothesis:

- $\varphi_i(d) = \varphi_{i-1}(d) = \varphi(d)$ if K_i (and thus K_{i-1}) is a contraction kernel.
- $\sigma_i(d) = \sigma_{i-1}(d) = \sigma(d)$ if K_i and K_{i-1} are removal kernels.

If K_i and K_{i-1} have not the same type,

$$CDS_i(d) = d \cdot CDS^*_{i-1}(\alpha(d)) = (d)$$

We have thus $CDS_{i-1}(\alpha(d)) = \alpha(d)$, and using our recurrence hypothesis and equation (2):

- If K_i is a contraction kernel and K_{i-1} a removal one:

$$\varphi_i(d) = \varphi_{i-1}(d) = \sigma_{i-1}(\alpha(d)) = \sigma(\alpha(d)) = \varphi(d)$$

- If K_i is a removal kernel and K_{i-1} a contraction kernel.

$$\sigma_i(d) = \sigma_{i-1}(d) = \varphi_{i-1}(\alpha(d)) = \varphi(\alpha(d)) = \sigma(d)$$

□

Proposition 16 *Given a combinatorial map $G_0 = (\boldsymbol{\mathcal{D}}, \sigma, \alpha)$ and a sequence of contraction or removal kernels $K_1, K_2 \ldots, K_n$. For any level i in $\{1, \ldots, n\}$, the second dart of $CDS_i(d)$ with d in $\mathcal{SD}_i$ is, when it exists, equal to $\varphi(d)$ if K_i is a contraction kernel and $\sigma(d)$ if K_i is a removal kernel:*

$$\forall i \in \{1, \ldots, n\} \quad \forall d \in \mathcal{SD}_i \quad | \quad CDS_i(d) = (d, d_1, \ldots, d_p) \textit{ with } p > 0$$

$$d_1 = \begin{cases} \varphi(d) & \textit{If } K_i \textit{ is a contraction kernel} \\ \sigma(d) & \textit{If } K_i \textit{ is a removal kernel} \end{cases}$$

Proof:

This proposition cannot be applied to $i = 0$, since at this level all connecting dart sequences are reduced to a singleton. If $i = 1$ we have for each d in $\mathcal{SD}_1$:

$$CDS_1(d) = CW_1(d) = (d, d_1 \ldots d_p) \text{ with } p \geq 0$$

If $p > 0$ for one $d \in \mathcal{SD}_1$, we have by definition of a connecting walk:

$$d_1 = \begin{cases} \varphi(d) & \text{If } K_1 \text{ is a contraction kernel} \\ \sigma(d) & \text{If } K_1 \text{ is a removal kernel} \end{cases}$$

The proposition is thus true for $i = 1$, let us suppose it is true until a given rank $i - 1 < n$.

Let us consider a given dart d in $\mathcal{SD}_i$ such that:

$$CW_i(d) = (d, d'_1, \ldots, d'_q)$$

Moreover, we suppose that $|CDS_i(d)| > 1$ to fulfill the requirements of the proposition.

- If K_i and K_{i-1} have the same type

  $$CDS_i(d) = CDS_{i-1}(d) \cdot CDS_{i-1}(d'_1) \cdots CDS_{i-1}(d'_q)$$

  If $|CDS_{i-1}(d)| > 1$ we can apply the recursive hypothesis. If not, we have $CDS_{i-1}(d) = (d)$. In this case we must have $q > 0$ since $q = 0$ and $CDS_{i-1}(d) = (d)$ implies that $CDS_i(d) = d$ which is refused by hypothesis. Therefore, d'_1 exists and $d_1 = d'_1$. Using proposition 15:

  - If K_i is a contraction kernel,

    $$d_1 = d'_1 = \varphi_{i-1}(d) = \varphi(d)$$

– If K_i is a removal kernel,

$$d_1 = d'_1 = \sigma_{i-1}(d) = \sigma(d)$$

- If K_i and K_{i-1} have not the same type:

$$CDS_i(d) = d \cdot CDS^*_{i-1}(\alpha(d)) \cdot d'_1 \cdot CDS^*_{i-1}(\alpha(d'_1)) \cdots d'_q \cdot CDS^*_{i-1}(\alpha(d'_q))$$

Let us denote:

$$CDS_{i-1}(\alpha(d)) = (\alpha(d), b_1, \ldots, b_r)$$

– If $|r| \geq 1$ the second dart d_1 of $CDS_i(d)$ is equal to b_1. Then using our recurrence hypothesis:

  * If K_i is a contraction kernel, then K_{i-1} is a removal kernel and:
  $$d_1 = b_1 = \sigma(\alpha(d)) = \varphi(d)$$
  * If K_i is a removal kernel, K_{i-1} is a contraction kernel and:
  $$d_1 = b_1 = \varphi(\alpha(d)) = \sigma(d)$$

– If $CDS_{i-1}(\alpha(d)) = (\alpha(d))$, then $d_1 = d'_1$, moreover:

  * If K_i is a contraction kernel, K_{i-1} is a removal kernel. Therefore, using proposition 15:
  $$\sigma_{i-1}(\alpha(d)) = \sigma(\alpha(d)) = \varphi(d)$$
  and:
  $$d_1 = d'_1 = \varphi_{i-1}(d) = \sigma_{i-1}(\alpha(d)) = \varphi(d)$$
  * If K_i is a removal kernel, K_{i-1} is a contraction kernel. Therefore $\varphi_{i-1}(\alpha(d)) = \varphi(\alpha(d)) = \sigma(d)$ and:
  $$d_1 = d'_1 = \sigma_{i-1}(d) = \varphi_{i-1}(\alpha(d)) = \sigma(d)$$

□

Proposition 17 *Given a combinatorial map* $G_0 = (\boldsymbol{\mathcal{D}}, \sigma, \alpha)$ *and a sequence of contraction or removal kernels* $K_1, K_2 \dots, K_n$. *The connecting dart sequence of any dart* $d \in \mathcal{SD}_i$ *defined at level* $i \in \{1, \dots, n\}$*:*

$$CDS_i(d) = (d_1, \dots, d_p),\ p > 1$$

satisfies the following property:

- *If* K_i *is a contraction kernel:*

$$\varphi_i(d) = \begin{cases} \varphi(d_p) & \text{if } d_p \text{ is contracted} \\ \sigma(d_p) & \text{if } d_p \text{ is removed} \end{cases}$$

- *If* K_i *is a removal kernel:*

$$\sigma_i(d) = \begin{cases} \varphi(d_p) & \text{if } d_p \text{ is contracted} \\ \sigma(d_p) & \text{if } d_p \text{ is removed} \end{cases}$$

Proof:

Let us show this proposition by recurrence.

1. If $i = 1$, the connecting dart sequences are equal to the connecting walks. Thus:

   $$\forall d \in \mathcal{SD}_1 \quad CDS_1(d) = CW_1(d) = (d_1, \dots, d_p)$$

   Using , the definition of connecting walks:

   (a) If K_1 is a contraction kernel, d_p is contracted and: $\varphi_1(d) = \varphi(d_p)$

   (b) If K_1 is a removal kernel, d_p is removed and: $\sigma_1(d) = \sigma(d_p)$

   The proposition is thus true for $i = 1$.

2. Let us suppose that it holds until a given rank $i - 1$.

   (a) If K_i and K_{i-1} have the same type, then:

   $$CDS_i(d) = CDS_{i-1}(d'_1) \cdots CDS_{i-1}(d'_q) = (d_1, \dots, d_p)$$

   with $CW_i(d) = (d'_1, \dots, d'_q)$.
   Then, the last dart of $CDS_i(d)$ is the last dart of $CDS_{i-1}(d'_q)$ and we have, using the definition of connecting walks and the recurrence hypothesis:

i. If K_i is a contraction kernel,

$$\varphi_i(d) = \varphi_{i-1}(d'_q) = \begin{cases} \varphi(d_p) & \text{If } d_p \text{ is contracted} \\ \sigma(d_p) & \text{If } d_p \text{ is removed} \end{cases}$$

ii. If K_i is a removal kernel,

$$\sigma_i(d) = \sigma_{i-1}(d'_q) = \begin{cases} \varphi(d_p) & \text{If } d_p \text{ is contracted} \\ \sigma(d_p) & \text{If } d_p \text{ is removed} \end{cases}$$

(b) If K_i and K_{i-1} have not the same type:

$$\begin{aligned} CDS_i(d) &= d'_1 \cdot CDS^*_{i-1}(\alpha(d'_1)) \cdots d'_q \cdot CDS^*_{i-1}(\alpha(d'_q)) \\ &= (d_1, \ldots, d_p) \end{aligned}$$

with $CW_i(d) = (d'_1, \ldots, d'_q)$.

i. If $CDS_{i-1}(\alpha(d'_q)) = (\alpha(d'_q))$, then the last dart of $CDS_i(d)$ is d'_q which is contracted or removed at level i according to K_i.

A. If K_i is a contraction kernel, then K_{i-1} is a removal kernel and d'_q is contracted at level i. Then we have by definition of connecting walks:

$$\varphi_i(d) = \varphi_{i-1}(d'_q) = \sigma_{i-1}(\alpha(d'_q))$$

Moreover, using proposition 15, since K_{i-1} is a removal kernel:

$$\sigma_{i-1}(\alpha(d'_q)) = \sigma(\alpha(d'_q)) = \varphi(d'_q)$$

therefore, $\varphi_i(d) = \varphi(d'_q)$

B. If K_i is a removal kernel, then K_{i-1} is a contraction kernel, and d'_q is removed at level i. Using proposition 15:

$$\varphi_{i-1}(\alpha(d'_q)) = \varphi(\alpha(d'_q)) = \sigma(d'_q)$$

Using the definition of connecting walks, we have: $\sigma_i(d) = \sigma_{i-1}(d'_q) = \varphi_{i-1}(\alpha(d'_q))$. Thus:

$$\sigma_i(d) = \sigma(d'_q)$$

ii. If $|CDS_{i-1}(\alpha(d'_q))| > 1$, then the last dart of $CDS_i(d)$ is the last dart of $CDS_{i-1}(\alpha(d'_q))$. Therefore, using our recurrence hypothesis:

A. If K_i is a contraction kernel, then

$$\varphi_i(d) = \varphi_{i-1}(d'_q) = \sigma_{i-1}(\alpha(d'_q)) = \begin{cases} \varphi(d_p) & \text{If } d_p \text{ is contracted} \\ \sigma(d_p) & \text{If } d_p \text{ is removed} \end{cases}$$

B. If K_i is a removal kernel, then

$$\sigma_i(d) = \sigma_{i-1}(d'_q) = \varphi_{i-1}(\alpha(d'_q)) = \begin{cases} \varphi(d_p) & \text{If } d_p \text{ is contracted} \\ \sigma(d_p) & \text{If } d_p \text{ is removed} \end{cases}$$

□

Proposition 18 *Given a combinatorial map $G_0 = (\mathcal{D}, \sigma, \alpha)$ and a sequence of contraction or removal kernels $K_1, K_2 \dots, K_n$. For each i in $\{1, \dots, n\}$, each dart d in $\mathcal{D}$ belongs to exactly one connecting dart sequence defined at level i:*

$$\forall i \in \{1, \dots, n\} \quad \forall d \in \mathcal{D} \quad \exists! d' \in \mathcal{SD}_i \quad | \quad d \in CDS_i(d')$$

Proof:

Let us show this proposition by recurrence:

If $i = 0$, $CDS_0(d)$ is equal to d for each d in $\mathcal{D}$. The proposition is then trivial. Let us suppose that the proposition is true until a given rank $i - 1$.

If d is neither contracted nor removed until level $i + 1$, it is the first dart of its own connecting dart sequence until level i (see proposition 12) and the demonstration of the existence of a connecting dart sequence at level i containing d is then trivial. Moreover, in this case, if d is contained in two connecting dart sequences defined at level i, it must be the first dart of both connecting dart sequences (see proposition 13). The first dart of a connecting dart sequence being the one which defines it(see definition 11), the demonstration of the uniqueness of the connecting dart sequence containing d is trivial.

If d is contracted or removed at level i, there exists a dart d'' in $\mathcal{SD}_i$ such that $d \in CW_i(d'')$. Therefore, d must belong to $CDS_i(d'')$ (see definition 11). Moreover, according to the definition of a connecting dart sequence, the darts contracted at level i which belong to a connecting dart sequence, must belong to its associated connecting walk. Therefore, if d belongs to two connecting dart sequences $CDS_i(d'')$ and $CDS_i(b)$, it must also belong to the connecting walks $CW_i(d'')$ and $CW_i(b)$. Since the set of connecting walks defined at level i defines a partition of $\mathcal{SD}_{i-1}$ [6], we have $d'' = b$ and therefore, $CDS_i(d'') = CDS_i(b)$.

Let us now suppose that d has been contracted or removed before level i. Then d cannot be the first dart of any connecting walk defined at levels i or $i-1$. Let us now decompose the demonstration at rank i in two steps:

Existence: According to our recurrence hypothesis, there exists a unique dart d' in $\mathcal{SD}_{i-1}$ such that: $d \in CDS_{i-1}(d')$.

- If K_i and K_{i-1} have the same type there exists a unique dart $d'' \in \mathcal{SD}_i$ such that $d' \in CW_i(d'')$, with:

$$CDS_i(d'') = CDS_{i-1}(d_1) \cdots CDS_{i-1}(d_p)$$

since $d' \in CW_i(d'') = d_1, \ldots, d_p$, we have:

$$d \in CDS_{i-1}(d') \subset CDS_i(d'')$$

- If K_i and K_{i-1} have not the same type, there exists a unique dart d'' in $\mathcal{SD}_i$ such that $\alpha(d') \in CW_i(d'')$. Moreover, since d is not the first dart of $CDS_{i-1}(d')$:

$$\begin{array}{l} d \in CDS^*_{i-1}(d') \subset CDS_i(d'') \; with \\ CDS_i(d'') = d_1 \cdot CDS^*_{i-1}(\alpha(d_1) \cdots d_p \cdot CDS^*_{i-1}(\alpha(d_p) \end{array}$$

and $\alpha(d') \in CW_i(d'') = (d_1, \ldots, d_p)$.

Uniqueness: Let us suppose that there exist at least two darts in $\mathcal{SD}_i$ such that $d \in CDS_i(d') \cap CDS_i(d'')$

- If K_i and K_{i-1} have the same type, then:

$$\left\{ \begin{array}{lcl} CDS_i(d') & = & CDS_{i-1}(b_1) \cdots CDS_{i-1}(b_p) \\ CDS_i(d'') & = & CDS_{i-1}(b'_1) \cdots CDS_{i-1}(b'_q) \end{array} \right.$$

with

$$\left\{ \begin{array}{lcl} CW_i(d') & = & (b_1, \ldots, b_p) \\ CW_i(d'') & = & (b'_1, \ldots, b'_q) \end{array} \right.$$

If $d \in CDS_i(d') \cap CDS_i(d'')$, then there must exist at least two indices $i \in \{1, \ldots, p\}$ and $j \in \{1, \ldots, q\}$ respectively such that $d \in CDS_{i-1}(b_i) \cap CDS_{i-1}(b'_j)$. Using our recurrence hypothesis,

d can belong to only one connecting dart sequence at level $i-1$, therefore, $b_i = b'_j$. We have therefore one dart in $\mathcal{SD}_i$ which belongs to two different connecting walks $CW_i(d')$ and $CW_i(d'')$ which is impossible since the connecting walk at level i form a partition of $\mathcal{SD}_{i-1}$(see [6] proposition 6).

- If K_i and K_{i-1} have the not same type, then:

$$\begin{cases} CDS_i(d') &= b_1 \cdot CDS^*_{i-1}(\alpha(b_1)) \cdots b_p \cdot CDS^*_{i-1}(\alpha(b_p)) \\ CDS_i(d'') &= b'_1 \cdot CDS^*_{i-1}(\alpha(b'_1)) \cdots b'_q \cdot CDS^*_{i-1}(\alpha(b'_q)) \end{cases}$$

with $(b_1, \ldots, b_p)$ and $(b'_1, \ldots, b'_q)$ defined as previously.

Since d is contracted or removed before level i, it does not belong to $\{b_1, \ldots, b_p\} \cup \{b'_1, \ldots, b'_q\}$ which are contracted or removed at level i.

Therefore, if $d \in CDS_i(d') \cap CDS_i(d'')$, there must exist two indices, $i \in \{1, \ldots, p\}$ and $j \in \{1, \ldots, q\}$ such that:

$$\begin{array}{lcl} d & \in & CDS^*_{i-1}(\alpha(b_i)) \cap CDS^*_{i-1}(\alpha(b'_j)) \\ \Rightarrow d & \in & CDS_{i-1}(\alpha(b_i)) \cap CDS_{i-1}(\alpha(b'_j)) \end{array}$$

Using our recurrence hypothesis we must have: $\alpha(b_i) = \alpha(b'_j)$ and thus $b_i = b'_j$. We obtain the same contradiction as previously.

□

Using connecting walks, each connecting walk is included in one φ-orbit, or one σ-orbit if the kernel is a removal one. Moreover, each dart appears only once in each φ-orbit and σ-orbit. Thus each dart appears also only once in each connecting walk. The connecting dart sequence are not included in one φ-orbit nor in one $\sigma-$orbit. Therefore, we have to demonstrate that each dart appears at most once in each connecting dart sequence:

Proposition 19 *Given a combinatorial map $G_0 = (\boldsymbol{\mathcal{D}}, \sigma, \alpha)$, a sequence of contraction kernels or removal kernels $K_1, K_2 \ldots, K_n$, each dart d appears at most once in each connecting dart sequence.*

Proof:

This property is trivial for $i = 0$. Let us suppose that it holds until a given level $i-1$ and let us suppose that we can find a dart d in $\boldsymbol{\mathcal{D}}$ that appears at least twice in a given connecting dart sequence $CDS_i(d')$.

- If K_i and K_{i-1} have the same type, then:

$$CDS_i(d') = CDS_{i-1}(d_1) \cdots CDS_{i-1}(d_p)$$

with $CW_i(d') = d_1, \ldots, d_p$.

Using our recurrence hypothesis, d cannot appear twice in the same connecting dart sequence at level $i-1$. Therefore, it must exists, two different indices i and j in $\{1, \ldots, p\}$ such that $d \in CDS_{i-1}(d_i) \cap CDS_{i-1}(d_j)$. This last assertion contradicts the fact that each dart belongs to exactly one connecting dart sequence defined at level $i-1$ (see proposition 18).

- If K_i and K_{i-1} have not the same type, then:

$$CDS_i(d') = d_1 \cdot CDS^*_{i-1}(\alpha(d_1)), \ldots, d_p \cdot CDS^*_{i-1}(\alpha(d_p))$$

with $CW_i(d') = d_1, \ldots, d_p$.

The dart d cannot appear twice in a connecting walk(see [6]). Therefore, d appears at most once in $CW_i(d')$. Note that if d belongs to $CW_i(d')$ it must be contracted or removed at level i.

Moreover, if d belongs to one $CDS^*_{i-1}(\alpha(d_j))$ with $j \in \{1, \ldots, p\}$ then it must belongs to one K_l with $l < i$ (see proposition 13). Therefore, d cannot belong simultaneously to $CW_i(d)$ and to one $CDS^*_{i-1}(\alpha(d_j))$.

Therefore, d can appear twice in $CDS_i(d)$ only if there exist two different indices j and k such that:

$$d \in CDS^*_{i-1}(\alpha(d_j)) \cap CDS^*_{i-1}(\alpha(d_k))$$

Therefore, d must belong to $CDS_{i-1}(\alpha(d_j)) \cap CDS_{i-1}(\alpha(d_k))$. This last assertion contradicts proposition 18.

□

3.1.1 Traversing Connecting Dart Sequences

Proposition 17 allows us to compute the φ_i and σ_i successors of a given dart d at any level i by using its connecting dart sequence. Therefore, if we are able to design an algorithm which traverses the connecting dart sequences of any darts at any level we should be able to build the different contracted combinatorial maps. However, the traversal of a connecting dart sequence induces the determination of the relation which links two successive darts within a connecting dart sequence. Using connecting walks, this relation is given by the definition of a connecting walk. Using connecting dart sequences we have to build a constructive definition from the recursive one. The following proposition shows that the successor of a dart in a given connecting dart sequence remains the same for all levels once it is defined.

Proposition 20 *Given a combinatorial map $G_0 = (\boldsymbol{\mathcal{D}}, \sigma, \alpha)$ and a sequence of contraction or removal kernels $K_1, K_2 \ldots, K_n$. If a dart d belongs to a connecting dart sequence $CDS_i(d')$ and if d is neither the first nor the last dart of $CDS_i(d')$ then its successor within the connecting dart sequence will be the same in all connecting dart sequences which include d and which are defined at a level greater than i.*

Proof:

Let us consider the smallest index l of the contraction kernels such that there exists a connecting dart sequence $CDS_l(d')$ including d and such that d is neither the first nor the last dart of the connecting dart sequence. Note that using proposition 13, if d is not the first dart of $CDS_l(d)$ it must be contracted or removed before level l (see the paragraph below this proof). Let us suppose that $CDS_l(d') = (d_1, \ldots, d_p)$ and that d is one of the darts $\{d_2, \ldots, d_{p-1}\}$. Let us show that the successor of d remains the same in all connecting dart sequences containing d defined at a level greater than or equal to l. The proposition is trivial at level l, let us suppose it is true until a level $k < n$. We have thus a dart d' in $\mathcal{SD}_k$ such that $CDS_k(d') = (d_1, \ldots, d_p)$, $d = d_m$, $m \in \{2, \ldots, p-1\}$ and d_{m+1} is the successor of d from level l.

- If K_k and K_{k+1} have the same type, let us consider d'' in $\mathcal{SD}_{k+1}$ such that $d' \in CW_{k+1}(d'')$. Then:

  $$CDS_{k+1}(d'') = CDS_k(b_1) \cdots CDS_k(b_q)$$

  with $d' \in CW_{k+1}(d'') = (b_1, \ldots, b_q)$. The successor of d in $CDS_{k+1}(d'')$ is then the same as in $CDS_k(d')$. Moreover, d is neither the first nor the last dart of $CDS_{k+1}(d'')$ by construction.

- If K_k and K_{k+1} have not the same type, let us consider the dart d'' in $\mathcal{SD}_{k+1}$ such that $\alpha(d') \in CW_{k+1}(d'')$:

  $$CDS_{k+1}(d'') = b_1 \cdot CDS^*_k(\alpha(b_1)) \cdots b_q \cdot CDS^*_k(\alpha(b_q))$$

  with $\alpha(d') \in CW_{k+1}(d'') = (b_1, \dots, b_q)$. Since d is not the first nor the last dart of $CDS_k(d')$, its successor remains the same in $CDS_{k+1}(d'')$. Moreover, d is not the first nor the last dart of $CDS_{k+1}(d'')$.

In both cases, $CDS_{k+1}(d'')$ satisfy the recursive hypothesis. Moreover, using proposition 18, $CDS_{k+1}(d'')$ is the unique connecting dart sequence defined at level $k+1$ containing d. Therefore, our recursive hypothesis holds until level $k+1$. $\square$

Proposition 16 shows us that the successor of the first dart of a connecting dart sequence defined at one level depends on the type of contraction applied at this level. On the contrary proposition 20 shows us that the successors of the other darts do not depend on the type of the applied contraction.

Therefore, the successor of a dart d in the connecting dart sequence which contains it changes at each level according to the type of the associated kernel until d is contracted or removed. Then the successor of d in the connecting dart sequences which contain it remains the same for all levels greater than $level(d)$.

In the following we will determine the relationships between two successive darts within a connecting dart sequence.

Proposition 21 *Given a combinatorial map $G_0 = (\mathcal{D}, \sigma, \alpha)$, a dart d in $\mathcal{D}$ and a sequence of contraction or removal kernels $K_1, K_2 \dots, K_n$. If d is contracted or removed at level $l < n$, the set of levels I_d defined by:*

$$I_d = \{i \in \{l, \dots, n\} \mid \exists! d' \in \mathcal{SD}_i, CDS_i(d') = (d_1, \dots, d)\}$$

is either empty or is a contiguous interval of $\{l, \dots, n\}$ containing l.

Proof:

Let us suppose that I_d is non empty and let us show that if $k > l$ belongs to I_d then $k-1$ does.

If k belongs to I_d there exists one dart d' such that d is the last dart of $CDS_k(d')$. Let us denote by $d_1, \dots, d_p$ the connecting walk of d' defined at level k:

$$CW_k(d') = (d_1, \dots, d_p)$$

- If K_k and K_{k-1} have the same type, then:

$$CDS_k(d') = CDS_{k-1}(d_1) \cdots CDS_{k-1}(d_p)$$

  Therefore, $CDS_k(d')$ and $CDS_{k-1}(d_p)$ have the same last dart d and the recursive hypothesis holds at level $k-1$.

- If K_k and K_{k-1} have not the same type, then:

$$CDS_k(d') = d_1 \cdot CDS^*_{k-1}(\alpha(d_1)) \cdots d_p \cdot CDS^*_{k-1}(\alpha(d_p))$$

  If $|CDS_{k-1}(\alpha(d_p))| > 1$, the last dart of this connecting dart sequence must be equal to d and the recursive hypothesis holds at level $k-1$.

  If $|CDS_{k-1}(\alpha(d_p))| = 1$ then, since the last dart of $CDS_k(d')$ is equal to d, we must have $d_p = d$ and d is contracted or removed at level $k = l$. We have thus nothing to demonstrate since l is the smallest index contained in I_d.

Note that in both cases, the uniqueness of the connecting dart sequence containing d at level $k-1$ is insured by proposition 18. Moreover, the above verification by induction stops only for $k = l$. Therefore, the lower bound of I_d, must be equal to l if I_d is non empty. □

Using proposition 21, if I_d is non-empty it can be written as $\{level(d), \ldots, m\}$ where m denotes the upper bound of I_d. Moreover, using Proposition 17 we can determine both $\varphi_i(d^i)$ and $\sigma_i(d^i)$ from $\varphi(d)$ and $\sigma(d)$ for each dart d^i in $\mathcal{SD}_i$ such that the last dart $CDS_i(d^i)$ is d.

Corollary 5 *Using the same notations and hypothesis as proposition 21 if I_d is non empty, then it can be denoted by $I_d = \{l, \ldots, m\}$, where m denotes the upper bound of I_d. If $m < n$, the successor of d in $CDS_{m+1}(d')$ is equal to $\varphi(d)$ if d is contracted and $\sigma(d)$ if d is removed where $CDS_{m+1}(d')$ denotes the connecting dart sequence which contains d at level $m+1$.*

Proof:

- If K_m and K_{m+1} have the same type, let us denote by d' the dart whose connecting walk at level $m+1$ includes d^m:

$$d^m \in CW_{m+1}(d') = (d_1, \ldots, d_p)$$

Let us suppose that $d^m = d_i$ with i in $\{1, \dots, p\}$. Since d is the last dart of $CDS_m(d^m)$, but not the last dart of $CDS_{m+1}(d')$ (by definition of m) the index i cannot be equal to p. The successor of d in $CDS_{m+1}(d')$ is then equal to d_{i+1}. Using proposition 17:

– If d is contracted

$$d_{i+1} = \varphi(d) = \begin{cases} \varphi_m(d^m) & \text{If } K_{m+1} \text{ is a contraction kernel} \\ \sigma_m(d^m) & \text{If } K_{m+1} \text{ is a removal kernel} \end{cases}$$

– If d is removed

$$d_{i+1} = \sigma(d) = \begin{cases} \varphi_m(d^m) & \text{If } K_{m+1} \text{ is a contraction kernel} \\ \sigma_m(d^m) & \text{If } K_{m+1} \text{ is a removal kernel} \end{cases}$$

- If K_m and K_{m+1} have not the same type, let us denote by d' the dart whose connecting walk at level $m+1$ include $\alpha(d^m)$:

$$\alpha(d^m) \in CW_{m+1}(d') = (d_1, \dots, d_p)$$

We have then:

$$CDS_{m+1}(d') = d_1 \cdot CDS^*_m(\alpha(d_1)) \cdots d_p \cdot CDS^*_m(\alpha(d_p))$$

Like previously we cannot have $\alpha(d^m) = d_p$ since in this case d is also the last dart of $CDS_{m+1}(d')$ which is in contradiction with the definition of m. Let us suppose that $\alpha(d^m) = d_i$ with $i \in \{1, \dots, p-1\}$. Then d is the last dart of $CDS^*_m(\alpha(d_i))$ and its successor in $CDS_{m+1}(d')$ is d_{i+1}. We obtain thus the same conclusion as previously by using proposition 17.

□

Remark 5 *Note that, using the same notations as corollary 5, since d is contracted or removed before level m, it can't be the first dart of $CDS_{m+1}(d')$. Moreover, it is not the last dart of this connecting walk by definition of m. Therefore, using proposition 20, the successor of d in $CDS_{m+1}(d')$ remains the same in all connecting dart sequences which are defined at a level greater or equal to $m+1$ and which contain d.*

Proposition 22 *Given a combinatorial map $G_0 = (\boldsymbol{\mathcal{D}}, \sigma, \alpha)$ and a sequence of contraction or removal kernels $K_1, K_2 \ldots, K_n$. Given a dart d contracted or removed at level l, if I_d is empty, and if $CDS_l(d')$ denotes the connecting dart sequence containing d at level l, the successor of d in $CDS_l(d')$ is equal to $\varphi(d)$ if d is contracted and $\sigma(d)$ if d is removed.*

Proof:

Let us consider d' in $\mathcal{SD}_l$ such that:

$$d \in CW_l(d') = (d_1, \ldots, d_p)$$

- If K_l have K_{l-1} have he same type:

$$CDS_l(d') = CDS_{l-1}(d_1) \cdots CDS_{l-1}(d_p)$$

  - If $[CDS_{l-1}(d)| > 1$, the dart following d in $CDS_l(d')$ is given by proposition 16 and is equal to:

$$\begin{cases} \varphi(d) & \text{If } K_l \text{ and } K_{l-1} \text{ are contraction kernels} \\ \sigma(d) & \text{If } K_l \text{ and } K_{l-1} \text{ are removal kernels} \end{cases}$$

    Note that d is contracted at level l, therefore, the dart following d in $CDS_l(d')$ is equal to $\varphi(d)$ if d is contracted (K_l is a contraction kernel) and $\sigma(d)$ if d is removed (K_l is a removal kernel)

  - If $|CDS_{l-1}(d)| = 1$ the dart following d in $CDS_l(d')$ is given by proposition 15 and is equal to:

$$\begin{cases} \varphi_{l-1}(d) &=& \varphi(d) & \text{If } K_l \text{ and } K_{l-1} \text{ are contraction kernels} \\ \sigma_{l-1}(d) &=& \sigma(d) & \text{If } K_l \text{ and } K_{l-1} \text{ are removal kernels} \end{cases}$$

- If K_l and K_{l-1} have not the same type:

$$CDS_l(d') = d_1 \cdot CDS^*_{l-1}(\alpha(d_1)) \cdots d_p \cdot CDS^*_{l-1}(\alpha(d_p))$$

  - If $|CDS_{l-1}(\alpha(d))| > 1$ then the dart following d in $CDS_l(d')$ is the second dart of $CDS_{l-1}(\alpha(d))$ and is equal to(see proposition 16):

$$\begin{cases} \sigma(\alpha(d)) &=& \varphi(d) & \text{If } K_{l-1} \text{ is a removal kernel} \\ \varphi(\alpha(d)) &=& \sigma(d) & \text{If } K_{l-1} \text{ is a contraction kernel} \end{cases}$$

    Since K_l and K_{l-1} have not the same type, the dart following d in $CDS_l(d')$ is equal to $\varphi(d)$ if d is contracted (K_l is a contraction kernel) and $\sigma(d)$ if d is removed (K_l is a removal kernel).

- If $|CDS_{l-1}(\alpha(d))| = 1$ then the successor of d in $CDS_l(d')$ is the successor of d in $CW_l(d')$ and is equal to (see proposition 15):
  - ∗ If K_l is a contraction kernel and K_{l-1} a removal one:
  $$\varphi_{l-1}(d) = \sigma_{l-1}(\alpha(d)) = \sigma(\alpha(d)) = \varphi(d)$$
  - ∗ If K_l is a removal kernel and K_{l-1} a contraction kernel:
  $$\sigma_{l-1}(d) = \varphi_{l-1}(\alpha(d)) = \varphi(\alpha(d)) = \sigma(d)$$

  Note that d cannot be the last dart of $CW_l(d')$ since I_d is empty by hypothesis.

□

Corollary 6 *Using the same notations as proposition 22, since d is contracted at level l, it can't be the first dart of $CDS_l(d')$. Moreover, since I_d is empty, d is not the last dart of $CDS_l(d')$. Therefore, using proposition 20, the successor of d in all connecting dart sequences defined at a level greater or equal to l and containing d is equal to $\varphi(d)$ if d is contracted and $\sigma(d)$ if d is removed.*

Theorem 1 *Given a combinatorial map $G_0 = (\boldsymbol{\mathcal{D}}, \sigma, \alpha)$, a sequence of contraction kernels or removal kernels $K_1, K_2 \ldots, K_n$, the relation between the successive darts of a connecting dart sequence $CDS_i(d) = (d_1, \ldots, d_p)$, with $i \in \{1, \ldots, n\}$ and $d \in \mathcal{SD}_i$ is as follow:*

$$\begin{array}{rcl} & d_2 & = & \left\{ \begin{array}{ll} \varphi(d_1) & \textit{If } K_i \textit{ is a contraction kernel} \\ \sigma(d_1) & \textit{If } K_i \textit{ is a removal kernel} \end{array} \right. \\ & \textit{and} & & \\ \forall j \in \{3, \ldots, p\} & d_j & = & \left\{ \begin{array}{ll} \varphi(d_{j-1}) & \textit{if } d_{j-1} \textit{ is contracted} \\ \sigma(d_{j-1}) & \textit{if } d_{j-1} \textit{ is removed} \end{array} \right. \end{array}$$

Proof:

Given a connecting dart sequence $CDS_i(d) = (d_1, \ldots, d_p)$ defined at level i, the successor of d_1 is given by proposition 16. Let us consider a dart d_j with $j \in \{2, \ldots, p-1\}$. Since d_j is not the first dart of $CDS_i(d)$, it must be contracted at a level less than or equal to i (see proposition 13). Moreover, since d_j has a successor in $CDS_i(d)$ it cannot belong to the set I_{d_j}. Therefore, one of the two following statements must hold:

1. The set I_{d_j} is empty. In this case, the successor of d_j is given by proposition 22 (see also corollary 6) and is equal to $\varphi(d_j)$ if d_j is contracted and to $\sigma(d_j)$ if d_j is removed.
2. I_{d_j} is not empty and i is strictly greater than the maximal level contained in I_{d_j}. In this last case, we can apply corollary 5 (see also remark 5) and the successor of d_j, d_{i+1} is equal to $\varphi(d_j)$ if d_j is contracted and $\sigma(d_j)$ is d_j is removed.

□

3.2 Coding Contractions and Removals

Since two kinds of operations are allowed and necessary in the pyramid we have to add some information in the pyramid construction plan in order to encode in which way a dart disappears at a given level:

Definition 12 Generalized Pyramid Construction Plan
Given an initial combinatorial map G_0 and a sequence of successive contraction or removal kernels $K_1, \ldots, K_n$, the generalized pyramid construction plan $\mathcal{GLP}$ associated to this sequence is defined by the initial combinatorial map G_0, a function level defined on $\boldsymbol{\mathcal{D}}$ by:

$$\forall d \in \boldsymbol{\mathcal{D}} \quad level(d) = Max\{i \in \{1, \ldots, n+1\} \quad | \quad d \in \mathcal{SD}_{i-1}\}$$

and a (binary) function state from $\{1, \ldots, n\}$ to $\{Contracted, Removed\}$, which maps each level i into:

- *Contracted, if K_i is a contraction kernel,*
- *Removed, if K_i is a removal kernel.*

Proposition 23 *Given a combinatorial map $G_0 = (\boldsymbol{\mathcal{D}}, \sigma, \alpha)$, and a generalized pyramid construction plan $\mathcal{GLP} = (G_0, level, state)$ defined by n kernels, each kernel K_i is equal to the set of darts mapped to i by the function level.*

$$\forall i \in \{1, \ldots, n\} \quad K_i = \{d \in \boldsymbol{\mathcal{D}} \mid level(d) = i\}$$

Proof:
This demonstration is similar to the one of proposition 5. □

Proposition 24 *Given a combinatorial map $G_0 = (\boldsymbol{\mathcal{D}}, \sigma, \alpha)$, and a generalized pyramid construction plan $\mathcal{GLP} = (G_0, level, state)$ defined by n kernels, the set of surviving darts of the i^{th} contracted map is equal to the set of darts having a level strictly greater than i:*

$$\mathcal{SD}_i = \{d \in \boldsymbol{\mathcal{D}} \quad | \quad level(d) > i\}$$

Proof:

The surviving darts at level i are defined by (see proposition 10):

$$\mathcal{SD}_i = \boldsymbol{\mathcal{D}} - \bigcup_{j=1}^{i} K_j$$

Since (see proposition 23) $K_i = \{d \in \boldsymbol{\mathcal{D}} \quad | \quad level(d) = i\}$, a surviving dart of the i^{th} contraction kernel must have a level strictly greater than i:

$$\mathcal{SD}_i = \{d \in \boldsymbol{\mathcal{D}} \quad | \quad level(d) > i\}$$

□

Note that a dart d with $level(d) = n+1$ must survive: $\mathcal{SD}_n = \boldsymbol{\mathcal{D}} - \cup_{i=1}^{n} K_i$. Hence function state is not defined for the top level $n+1$. Furthermore recall that a dart d is removed from $\mathcal{SD}_i$ if K_i is a removal kernel. This is expressed now by: $state(level(d)) = Removed$.

We will show in the following, that the generalized pyramid construction plan and the function *survive* (see Algorithm 5) based on it allow us to retrieve the different contraction kernels and contracted combinatorial map defined by a sequence of contractions and/or removals.

Remark 6 *Note that given a generalized pyramid construction plan and a level $i \leq n$, three cases may occur for each dart d in $\boldsymbol{\mathcal{D}}$:*

1. *$level(d) > i$: the dart d remains at level i. This dart may disappear at an upper level or remain until level n.*
2. *$level(d) = i$ and $state(i) = Contracted$: The dart d is contracted at level i.*
3. *$level(d) = i$ and $state(i) = Removed$: The dart d is removed at level i.*

Let us now consider the Algorithm 5. Proposition 23 and 24 show that the generalized pyramid construction plan allows us to retrieve the different kernels and surviving darts of our pyramid. We will show in the following that Algorithm 5 together with a generalized pyramid construction plan allows us also to retrieve the contracted or removed combinatorial maps defined at the different levels of the pyramid.

```
dart survive(int i, dart d)
{
    if ( level(d) > i )
        return d;

    if(state(level(d)) == Contracted)
        return survive(i,φ(d))

    return survive(i,σ(d))
}
```

Algorithm 5: *The general survive algorithm*

Definition 13 survive Stack

Given a combinatorial map $G_0 = (\mathcal{D}, \sigma, \alpha)$ *and the generalized pyramid construction plan,* $\mathcal{GLP} = (G_0, level, state)$*. The ordered set* $Stack(i, d)$ *is the sequence of darts which will be passed as second argument of the recursive function survive during a call to* $survive(i, d)$*.*

Remark 7 *Using the same notations and hypothesis as definition 13, the last dart of* $Stack(i, d)$ *is equal to* $survive(i, d)$*.*

Proposition 25 *Using the same notations and hypothesis as definitions 13 and 11. For each* i *in* $\{1, \ldots, n\}$ *and for each* d *in* $\mathcal{SD}_i$*,* $CDS_i(d)$ *may be deduced from the stack of function survive thanks to the following relations:*

- *If* $state(i) = Contracted$

$$d \cdot Stack(i, \varphi(d)) = CDS_i(d) \cdot \varphi_i(d)$$

- *If* $state(i) = Removed$

$$d \cdot Stack(i, \sigma(d)) = CDS_i(d) \cdot \sigma_i(d)$$

Proof:

According to proposition 12, the first dart of $CDS_i(d) = (d_1, \ldots, d_p)$ is d, moreover, using proposition 16 the second dart of $CDS_i(d)$ is equal to $\varphi(d)$ if K_i is a contraction kernel and to $\sigma(d)$ if K_i is a removal kernel.

Therefore, the first two darts of $d \cdot Stack(i, \varphi(d))$ (resp. $d \cdot Stack(i, \sigma(d))$) and $CDS_i(d) \cdot \varphi_i(d)$ (resp. $CDS_i(d) \cdot \sigma_i(d)$) are equal if K_i is a contraction kernel (resp. a removal kernel). Let us now suppose that $state(i) = Contracted$ (the demonstration may be adapted easily if $state(i) = Removed$) and let us consider the series $(d'_1, \ldots, d'_{p+1})$ such that:

$$d \cdot Stack(i, \varphi(d)) = (d'_1, \ldots, d'_{p+1}) \text{ with } d'_1 = d$$

Let us suppose that both series $d \cdot Stack(i, \varphi(d))$ and $CDS_i(d) \cdot \varphi_i(d)$ are equal until a given rank $j \in \{2, \ldots, p-1\}$. Then, since $d_j = d'_j$ is not the first dart of $CDS_i(d)$ it must have been contracted before level i. We have thus $level(d_j) \leq i$. Moreover if d_j is contracted, its successor in $Stack(i, \varphi(d))$ is equal to $\varphi(d)$ while if d_j is removed its successor is equal to $\sigma(d_j)$ (see Algorithm 5). Using Theorem 1, the successor of d_j in $CDS_i(d)$ is equal to $\varphi(d_j)$ if d_j is contracted and $\sigma(d_j)$ if d_j is removed, therefore $d_{j+1} = d'_{j+1}$. We have thus:

$$\forall j \in \{1, \ldots, p\} \quad d'_j = d_j$$

Since d_p is not the first dart of $CDS_i(d)$, its level must be strictly less than i. Therefore $survive(i, d_p)$ is equal to:

$$\begin{array}{ll} survive(i, \varphi(d_p)) & \text{if } d_p \text{ is contracted} \\ survive(i, \sigma(d_p)) & \text{if } d_p \text{ is removed} \end{array}$$

Thus, the successor d'_{p+1} of d_p in $Stack(i, \varphi(d))$ is equal to $\varphi(d_p)$ if d_p is contracted and $\sigma(d_p)$ if d_p is removed. Using proposition 17, d'_{p+1} is equal to $\varphi_i(d)$. Therefore, $d'_{p+1} \in \mathcal{SD}_i$ and, using proposition 24, its level is strictly greater than i. Therefore, d'_{p+1} is the last dart of $Stack(i, \varphi(d))$ and both series $d \cdot Stack(i, \varphi(d))$ and $CDS_i(d) \cdot \varphi_i(d)$ are equal. □

Definition 14 Application Σ

Given a combinatorial map $G_0 = (\boldsymbol{\mathcal{D}}, \sigma, \alpha)$ and the generalized pyramid construction plan, $\mathcal{GLP} = (G_0, level, state)$. The application Σ from $\{1, \ldots, n\} \times \boldsymbol{\mathcal{D}}$ to $\boldsymbol{\mathcal{D}}$ is defined by:

$$\Sigma \left(\begin{array}{lcl} \{1, \ldots, n\} \times \boldsymbol{\mathcal{D}} & \rightarrow & \boldsymbol{\mathcal{D}} \\ (i, d) & \mapsto & survive(i, \sigma(d)) \end{array} \right.$$

Proposition 26 *Given an initial combinatorial map G_0 and the generalized pyramid construction plan, $\mathcal{GLP} = (G_0, level, state)$ the permutation σ_i of the i^{th} contracted map is equal to $\Sigma \circ p_i$:*

$$\forall i \in \{1, \ldots, n\} \quad \forall d \in \mathcal{SD}_i \quad \Sigma \circ p_i(d) = \sigma_i(d)$$

Proof:

Let us consider a level i in $\{1, \ldots, n\}$ and a dart d in $\mathcal{SD}_i$. The application $\Sigma \circ p_i$ maps d into $survive(i, \sigma(d))$. Using proposition 25 and remark 7:

- If $state(i) = Contracted$, $survive(i, \sigma(d))$ is the last dart of $Stack(i, \sigma(d) = \varphi(\alpha(d)))$. Therefore, using proposition 25:

$$survive(i, \sigma(d)) = \varphi_i(\alpha(d)) = \sigma_i(d)$$

- If $state(i) = Removed$, $survive(i, \sigma(d))$ is the last dart of $Stack(i, \sigma(d))$ and is thus equal to $\sigma_i(d)$.

Therefore, in all cases:

$$\Sigma \circ p_i(d) = survive(i, \sigma(d)) = \sigma_i(d)$$

□

Note that a direct encoding of the function Σ similar to the one illustrated in Table 1 may be defined thanks to the function *survive*.

4 Conclusion

The two major contributions of this technical report are:

- The study of pyramids defined by both contraction kernels and removal kernels.
- The definition of a pyramid construction plan and a generalized pyramid construction plan.

The definition of a pyramid defined by both contraction kernels and removal kernels allows us to remove the restrictions induced by a sole kind of operation (see section 2.2). We thus gain further flexibility which allows us to contract any initial combinatorial map into a smaller one eventually reduced to a self-direct-loop. The definition of the function *level*, allows us to store the set of kernels defining a given pyramid. An encoding of the pyramid based on the function *level* is also proposed.

The function *survive* defined in section 3.2 is designed to build a particular level of the pyramid. The construction of the function Σ or of all permutations σ_i with $i \in \{1, \ldots, n\}$ requires to apply this function on each level i and on each dart in $\mathcal{SD}_i$. This last operation may induce some unnecessary calculations and a new function adapted to the direct construction of several levels of the pyramid is under study.

Given a combinatorial pyramid either defined by $(\boldsymbol{\mathcal{D}}, \Sigma, level, \alpha)$, or by $(\boldsymbol{\mathcal{D}}, (\sigma_i)_{i \in \{1,\ldots,n\}}, \alpha)$ the modification of the pyramid in order to improve a previous result or to adapt it to new input data is often required. We plan to study this operation named relinking [9] in the combinatorial map framework. Finally, an implementation of combinatorial maps pyramids should allow to study interesting applications of our model such as: segmentation [4, 1, 3, 5], structural matching [16] or integration of moving objects.

References

[1] J. P. Braquelaire and L. Brun. Image segmentation with topological maps and inter-pixel representation. *Journal of Visual Communication and Image representation*, 9(1), 1998.

[2] R. P. Brent. *Algorithms for Minimization without Derivatives.* Prentice–Hall, Englewood Cliffs, New Jersey, 1973.

[3] L. Brun. *Segmentation d'images couleur à base Topologique.* PhD thesis, Université Bordeaux I, 351 cours de la Libération 33405 Talence, December 1996.

[4] L. Brun and J. P. Domenger. Incremental modifications on segmented image defined by discrete maps. Technical report, RR-112696 LaBRI, may 1996. Submitted.

[5] L. Brun and J. P.and J.P. Braquelaire Domenger. Discrete maps : a framework for region segmentation algorithms. In *Workshop on Graph based representations*, Lyon, April 1997. to be published in Advances in Computing (Springer).

[6] L. Brun and Walter Kropatsch. Equivalent contraction kernels with combinatorial maps. Technical Report 57, Institute of Computer Aided Design, Vienna University of Technology, lstr. 3/1832,A-1040 Vienna AUSTRIA, January 1999. URL: http://www.prip.tuwien.ac.at/.

[7] S. Fourey and R. Malgouyres. Intersection number of paths lying on a digital surface and a new jordan theorem". In Gilles Bertrand, Michel Couprie, and Laurent Perroton, editors, *Proceedings of DGCI'99, Lecture Notes in Computer Science*, volume 1568, pages 104–117, Marne-la-vallée, France, March 1999. Springer, Berlin Heidelberg New York.

[8] A. Jones Gareth and David Singerman. Theory of maps on orientable surfaces. volume 3, pages 273–307. London Mathematical Society, 1978.

[9] Roland Glantz and Walter G. Kropatsch. Relinking of graph pyramids by means of a new representation. In Tomás Svoboda, editor, *Czech Pattern Recognition Workshop 2000*. Czech Pattern Recognition Society, February 2000.

[10] Joseph Jaja. *An Introduction to Parallel Algorithms*. Addison-Wesley, Reading, 1992.

[11] Walter G. Kropatsch. Equivalent Contraction Kernels and The Domain of Dual Irregular Pyramids. Technical Report PRIP-TR-42, Institute f. Automation 183/2, Dept. for Pattern Recognition and Image Processing, TU Wien, Austria, 1995. Also available through http://www.prip.tuwien.ac.at/ftp/pub/publications/trs/tr42.ps.gz.

[12] Walter G. Kropatsch. Equivalent contraction kernels to build dual irregular pyramids. *Advances in Computer Science*, Advances in Computer Vision:pp. 99–107, 1997.

[13] Walter G. Kropatsch. From equivalent weighting functions to equivalent contraction kernels. In *CZECH PATTERN RECOGNITION WORKSHOP'97*, pages 1–13. Czech Pattern Recognition Society, February 1997.

[14] P. Lienhardt. Subdivisions of n-dimensional spaces and n-dimensional generalized maps. In Kurt Mehlhorn, editor, *Proceedings of the 5th Annual Symposium on Computational Geometry (SCG '89)*, pages 228–236, Saarbrücken, FRG, June 1989. ACM Press.

[15] Brun Luc and Walter Kropatsch. Dual contractions of combinatorial maps. Technical Report 54, Institute of Computer Aided Design, Vienna University of Technology, lstr. 3/1832,A-1040 Vienna AUSTRIA, January 1999. URL: http://www.prip.tuwien.ac.at/.

[16] Jean-Gerard Paillonc y, Walter G. Kropatsch, and Jean-Michel Jolion. Object Matching on Irregular Pyramid. In Anil K. Jain, Svetha Venkatesh, and Brian C. Lovell, editors, *14th International Conference on Pattern Recognition*, volume II, pages 1721–1723. IEEE Comp.Soc., 1998.

[17] Azriel Rosenfeld and Akira Nakamura. Local deformations of digital curves. *Pattern Recognition Letters*, 18:613–620, 1997.

A Appendix

A.1 Recursive construction of connecting dart sequences

The section illustrates the recursive construction of the connecting dart sequences. All tables given below show the different connecting walks associated to the pyramid defined in Figure 5. Note that these connecting dart sequences may also be constructed directly by using Theorem 1.

The connecting dart sequences of K_1				
$CDS_1(9)$	$=$	$CW_1(9)$	$=$	$9, -4$
$CDS_1(-9)$	$=$	$CW_1(-9)$	$=$	$-9, -2, -1, 7, 10$
$CDS_1(8)$	$=$	$CW_1(8)$	$=$	8
$CDS_1(-8)$	$=$	$CW_1(-8)$	$=$	$-8, 2$
$CDS_1(3)$	$=$	$CW_1(3)$	$=$	3
$CDS_1(-3)$	$=$	$CW_1(-3)$	$=$	$-3, -7, 1$
$CDS_1(5)$	$=$	$CW_1(5)$	$=$	$5, 6, -12$
$CDS_1(-5)$	$=$	$CW_1(-5)$	$=$	$-5, -10$
$CDS_1(11)$	$=$	$CW_1(11)$	$=$	11
$CDS_1(-11)$	$=$	$CW_1(-11)$	$=$	$-11, 4, 12, -6$

Table 2: The connecting dart sequences of the first contraction kernel K_1 defined in Figure 5

The connecting dart sequences of K_2		
$CW_2(3)$	$=$	$3, 8, 9$
$CDS_2(3)$	$=$	$3.CDS_1(-3)^*.8.CDS_1^*(-8).9.CDS_1^*(-9)$
	$=$	$3, -7, 1, 8, 2, 9, -2, -1, 7, 10$
$CW_2(-3)$	$=$	-3
$CDS_2(-3)$	$=$	$-3.CDS_1^*(3)$
	$=$	-3
$CW_2(5)$	$=$	5
$CDS_2(5)$	$=$	$5.CDS_1^*(-5)$
	$=$	$5, -10$
$CW_2(-5)$	$=$	$-5, -9, -8$
$CDS_2(-5)$	$=$	$-5.CDS_1^*(5). - 9.CDS_1^*(9). - 8.CDS_1^*(8)$
	$=$	$-5, 6, -12, -9, -4, -8$
$CW_2(11)$	$=$	11
$CDS_2(11)$	$=$	$11.CDS_1^*(-11)$
	$=$	$11, 4, 12, -6$
$CW_2(-11)$	$=$	-11
$CDS_2(-11)$	$=$	$-11.CDS_1^*(11)$
	$=$	-11

Table 3: The connecting dart sequences of the removal kernel K_2 defined in Figure 5

The connecting dart sequences of K_3		
$CW_3(5)$	$=$	$5, -3$
$CDS_3(5)$	$=$	$5.CDS_2^*(-5). -3.CDS_2^*(3)$
	$=$	$5, 6, -12, -9, -4, -8, -3, -7, 1, 8, 2, 9, -2, -1, 7, 10$
$CW_3(-5)$	$=$	$-5, 3$
$CDS_3(-5)$	$=$	$-5.CDS_2^*(5).3.CDS_2^*(-3)$
	$=$	$-5, -10, 3$
$CW_3(11)$	$=$	11
$CDS_3(11)$	$=$	$11.CDS_2^*(-11)$
	$=$	11
$CW_3(-11)$	$=$	-11
$CDS_3(-11)$	$=$	$-11.CDS_2^*(11)$
	$=$	$-11, 4, 12, -6$

Table 4: The connecting dart sequences of the contraction kernel K_3 defined in Figure 5

The connecting dart sequences of K_4		
$CW_4(11)$	$=$	11
$CDS_4(11)$	$=$	$11.CDS_3^*(-11)$
	$=$	$11, 4, 12, -6$
$CW_4(-11)$	$=$	$-11, -5, 5$
$CDS_4(-11)$	$=$	$-11.CDS_3^*(11) - 5.CDS_3^*(5).5.CDS_3^*(-5)$
	$=$	$-11, -5, 6, -12, -9, -4, -8, -3, -7, 1, 8, 2, 9, -2, -1, 7, 10, 5, -10, 3$

Table 5: The connecting dart sequences of the removal kernel K_4 defined in Figure 5 (see also Figure 6)

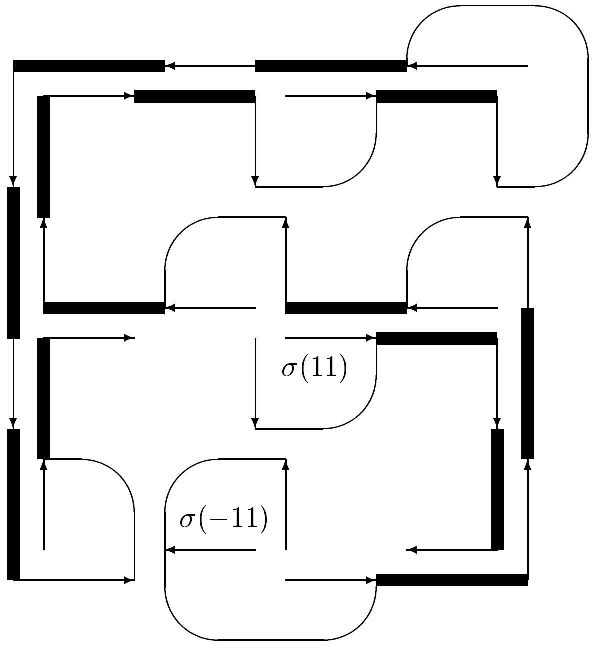


Figure 6: Illustration of the connecting dart sequences given on Table 5.

A.2 Index of Definitions

TR-54 refers to Definitions in [15], TR-57 to Definitions in [6] and TR-63 to Definitions in this technical report.