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: luc.brun@univ-reims.fr,
krw@prip.tuwien.ac.at

URL: http://www.prip.tuwien.ac.at/

PRIP-TR-82 July 8, 2003

Labeled Pyramids with Combinatorial Maps

Luc Brun and Walter Kropatsch[1]

Abstract

Combinatorial Pyramids are defined as a stack of successively reduced combinatorial maps. The Pyramid construction plan defined in TR-63 [7] allows to describe a pyramid by two functions *level* and *state* defined respectively on the set of darts of the initial combinatorial map and the set of levels of the pyramid. These two functions encode respectively the maximum level on which a dart survives and the type of each reduction operation. Based on these functions any combinatorial map of the pyramid may be built from the base by a one pass algorithm scanning all the darts of the initial combinatorial map [7]. In this technical report we show that algorithms with a same sequential and parallel complexity may be designed in order to build all the reduced combinatorial maps of the Pyramid.

[1]This Work was supported by the Austrian Science Foundation under P14445-MAT.

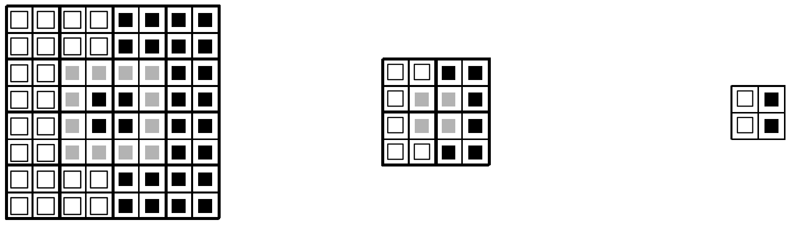
Contents

1 Introduction

Regular image pyramids have been introduced 1981/82 [12] as a stack of images with decreasing resolutions. Since then, regular pyramids have been widely used in image segmentation [12] and image analysis [24]. Using the neighborhood relationships defined on each image the *Reduction window* relates each pixel of the pyramid with a set of pixels defined in the level below. The pixels belonging to one reduction window are the *children* of the pixel which defines it. The value of each father is computed from the one of its children using a *Reduction function*. A regular pyramid is thus defined by the ratio $N \times N/q$ where $N \times N$ is the size of the reduction window, and q the ratio between the size of two consecutive images in the pyramid.

The father-child relationship defined by the reduction window maybe extended by transitivity down to the base level image. **The set of children of one pixel in the base level** is named its *receptive field* (RF) and defines the embedding of this pixel on the original image. Using the father-child relationship global properties of a receptive field $RF(v)$ with a diameter m may be computed in $\mathcal{O}(\log(m))$ parallel processing steps thanks to local calculus. However, receptive fields defined within the regular pyramid framework are not necessarily connected [1]. Furthermore, the adjacency of two pixels v and w defined at level k may not be easily interpreted on the base level image. This last drawback is illustrated in Fig. 1 where the initial 8×8 image is reduced by a $2 \times 2/4$ regular pyramid using the mean gray level as reduction function. Each black pixel in the central region belongs to a 2×2 reduction window with 3 gray pixels. These pixels are thus mapped onto a gray father and the black region which was disconnected at level 0 becomes connected at level 1. Finally, the boundary between the receptive fields $RF(v)$ and $RF(w)$ associated to this adjacency at level k may be disconnected and even incomplete (see [1] for more details).

Irregular pyramids, first introduced by Meer, Montanvert and Jolion [22] are defined as a stack of successively reduced simple graphs (i.e. graphs without double edges nor self-loops). The base level graph may be built from a sampling grid using one pixel adjacency such as the 4−neighborhood. Each graph of the hierarchy is built from the graph below by selecting a set of vertices named surviving vertices and mapping each non surviving vertex to a surviving one [22]. This mapping induces a father-child relationship between a surviving vertex and the set of non surviving vertices mapped to it. The reduction window of one surviving vertex is then defined as its set of children.

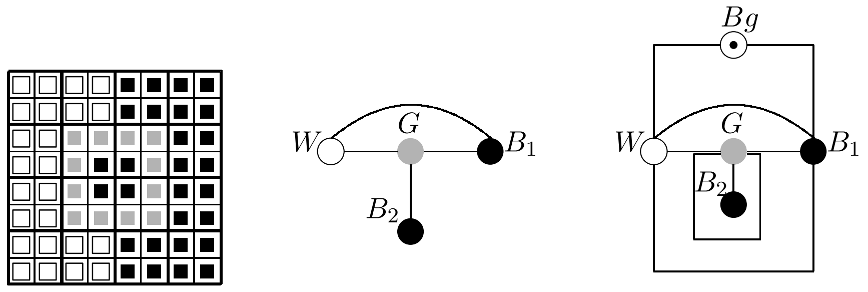


(a) Initial Image (b) level 1 (c) level 2

Figure 1: A $2 \times 2/4$ regular pyramid. The central black region is removed from level 0 to 1 due to the fixed decimation ratio and the reduce size of reduction windows.

The receptive field of one surviving vertex is defined by the transitive closure of the father-child relationship. Using this reduction scheme, the receptive field of each vertex in the hierarchy is a **connected set of vertices in the base level graph**. However, using simple graphs, the adjacency between two vertices is encoded by only one edge while the receptive fields of two vertices may share several boundary segments. An edge in the hierarchy may thus encode a non-connected set of boundaries between the associated receptive fields. Moreover, the lack of self-loops in simple graphs does not allow to differentiate an adjacency relationship between two receptive fields from an inclusion relationship. These two drawbacks are illustrated in Fig. 2(b) which represents the top of a simple graph pyramid encoding the connected components of Fig. 2(a). The two boundaries between the white region (W) and the black one on the right of the image (B_1) are encoded by only one edge. Moreover, the adjacencies between the gray region (G) and the two black ones (B_1 and B_2) are encoded in the same way by a simple edge. Therefore, these two different types of adjacency cannot be distinguished at the top of a simple graph pyramid.

These last two drawbacks may be overcome by using the Dual graph pyramids introduced by Kropatsch and Willersinn [26]. Using Kropatsch's reduction scheme, the reduction operation is encoded by edge contractions [26]. This operation contracts one edge and its two end points into a single vertex. It corresponds to the edge collapse operation used by Hoppe et al. [15] to simplify triangular meshes. The contraction of a graph reduces the number



(a) Initial image

(b) Simple graph pyramid

(c) Dual graph or combinatorial pyramids

Figure 2: Encoding of the connected components of a 8×8 image (a) by a simple graph pyramid (b) and the Dual Graph or Combinatorial Pyramids (c). The vertex B_g in (c) encodes the background of the image.

of vertices while maintaining the connections to other vertices. As a consequence some redundant edges such as self-loops or double edges may occur, some encode relevant topological relations (e.g. an island in a lake) others can be removed without any harm to the involved topology. These redundant edges may be characterized in the dual of the contracted graph. The removal of such edges is called a dual decimation step. Since the reduction scheme requires both features of the initial graph and its dual, such pyramids are called *Dual graph pyramids*. Within such hierarchies, each receptive field is a **connected set of vertices in the base level**. Moreover, each edge between two vertices encodes an unique connected boundary between the associated receptive fields. Finally, the use of self-loops within the hierarchy allows to differentiate adjacency relationships between receptive fields from inclusion relations. These two properties are illustrated in Fig. 2(c) which represents the top of a dual graph pyramid encoding the connected components of Fig. 2(a). The inclusion of the central black region within the gray one is encoded by the self-loop which surrounds the vertex B_2 associated to this region. Moreover, the two boundaries between the white region (W) and the black one on the right of the image (B_1) are encoded by two edges, each edge being associated to one boundary.

Experiences with connected component analysis [20], universal segmentation [18] and with topological analysis of line drawings [17, 19] show the great potential of Dual graph pyramids. However, since decimation and dual decimation require respectively features of the initial and dual graphs, both graphs must be encoded and maintained [16]. Therefore, any contraction operation in the initial graph must be followed by a removal operation in its dual. In the same way, during the dual decimation stage, any removal in the initial graph must be followed by a contraction operation in its dual. Moreover, the orientation of edges around one vertex is not explicitly encoded by the data structure [6].

A combinatorial map [14, 25, 13, 10] may be seen as a planar graph encoding explicitly the orientation of edges around a given vertex. Figure 3a) demonstrates the derivation of a combinatorial map from a plane graph. First edges are split into two half edges called *darts*, each dart having its origin at the vertex it is 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. Each orbit of σ is associated to one vertex and encodes the set of darts encountered when turning counterclockwise around this vertex. The symbols $\alpha^*(d)$ and $\sigma^*(d)$ stand, respectively, for the α and σ orbits of the dart d. For example, the upper-right vertex in Figure 3a) is associated to the $\sigma-$orbit $\sigma^*(1) = (1, 7)$. In the same way, the vertex located in the first row and the second column of Figure 3a) is associated to the $\sigma-$orbit $\sigma^*(2) = (2, -1, 8)$. More generally, if d is a dart and π a permutation we will denote the π-orbit of d by $\pi^*(d)$.

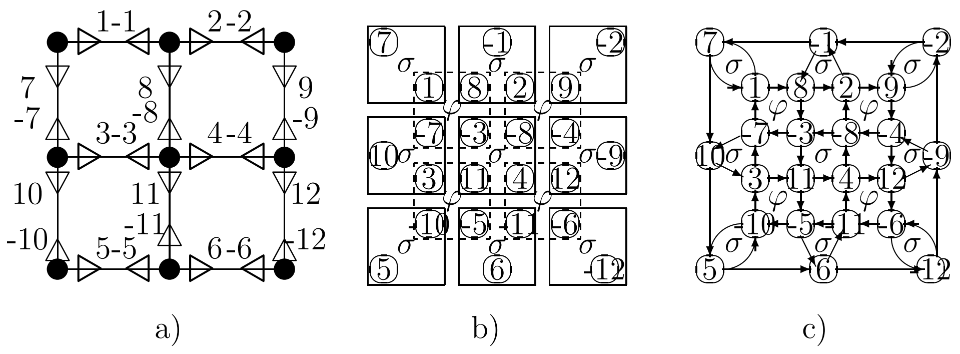


Figure 3: A 3×3 grid encoded by a combinatorial map

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

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

Note that, if the darts are encoded by positive and negative integers, the involution α may be implicitly encoded by sign (Figure 3a)). This convention is often used for practical implementations [3] where the combinatorial map is simply implemented by an array of integers encoding the permutation σ.

Given a combinatorial map $G = (\boldsymbol{\mathcal{D}}, \sigma, \alpha)$, its dual is defined by $\overline{G} = (\boldsymbol{\mathcal{D}}, \varphi, \alpha)$ with $\varphi = \sigma \circ \alpha$. The orbits of the permutation φ encode the set of darts encountered when turning around a face (see e.g. the φ-orbit $(1, 8, -3, -7)$ in Figure 3a)). Note that, using a counter-clockwise orientation for permutation σ, each dart of a φ-orbit has its associated face on its right.

Figures 3b) and 3c) illustrate an alternative representation of the combinatorial map encoding. Within such a representation, each dart is represented by one vertex and one edge connects a dart d_1 to d_2 iff either $d_2 = \sigma(d_1)$ or $d_2 = \varphi(d_1)$. Using this representation, the σ and φ orbits of the combinatorial map are represented by the faces of the oriented graph. The α successor of one dart may be retrieved on Figure 3c) by reading its σ-successor and then taking the predecessor of the resulting dart by the permutation φ. Indeed, since $\varphi = \sigma \circ \alpha$, we have $\alpha = \varphi^{-1} \circ \sigma$.

Using combinatorial maps, contraction and removal operations may be defined in order to preserve the orientation of darts around each vertex [6]. Given these two basic operations decimation and dual decimation parameters may be defined in order to produce a stack of successively reduced combinatorial maps [8]. The expected advantages of combinatorial maps within the irregular pyramid framework are:

1. Combinatorial maps encode multiple boundaries between regions. Moreover, edges encoding surrounding relationships may also be characterized.

2. Combinatorial maps explicitly encode the orientation of darts around one vertex. This information is not encoded by region adjacency graphs nor explicitly available in dual graph data structures.

3. Given a combinatorial map G defined by one set of darts $\boldsymbol{\mathcal{D}}$ and the permutations σ and α, its dual $\overline{G}$ is defined on the same set of darts by the permutations $\varphi = \sigma \circ \alpha$ and α. The simplicity and the efficiency of

this transformation allows us to avoid an explicit encoding of the dual graph [6]. Therefore, only one data structure has to be encoded and maintained along the pyramid [8].

4. The combinatorial map formalism may be extended to higher dimensions [21]. The definition of a partition of the 3D discrete grid using combinatorial maps is an active research field [2].

Basic definitions and properties of Combinatorial maps used in this document are defined in TR-54 [10]. This last technical report also defines the notion of decimation parameters for combinatorial maps. Using decimation parameters, each non surviving vertex has to be adjacent to a surviving one. This constraint allows to map each vertex of the initial combinatorial map on one processor and to perform the decimation process on a parallel machine for real time image analysis. However this decimation process provides a slow decimation rate. The contraction kernels, defined in TR-57 [11] extend the notion of decimation parameter. A removal kernel encodes a set of removal operations performed simultaneously and extends the notion of dual decimation parameter. Such kernels provide a better control of the decimation rate. If the pyramid is defined by one set of contraction kernels or one set of removal kernels, the reduced combinatorial maps may be defined by walks defined in the initial combinatorial map and named *connecting walks* [11]. Each connecting walk defined at level i of the pyramid, is defined by one dart d which survives up to level i. We have shown [11], that if the pyramid is defined by a sequence of contraction kernels, the φ successor of the last dart of the connecting walk $CW_i(d)$ defined at level i by the dart d is equal to the φ successor of d in the reduced combinatorial map. A similar result has been shown for pyramids defined by a sequence of removal kernels. Therefore, connecting walks allow us to determine all the reduced combinatorial maps of the pyramids. However, this result holds only if all the kernels have the same type e.g. either all kernels are contraction kernels or all kernels are removal kernels. If the pyramid is defined by a sequence of contraction and removal kernels connecting walks may be used to define a more general object named *connecting dart sequences*. The notion of connecting dart sequence have been introduced in TR-63 [7]. The essential property shown in TR-63, is that the last dart of the connecting dart sequence defined at level i by a dart d in $\mathcal{SD}_i$ allows to retrieve either the φ or σ successor of d according to the type of the kernel K_i. This property allows to retrieve all the contracted combinatorial maps defining the pyramid. However, using algorithms proposed in TR-63 only the last dart of each connecting dart sequence is used to compute the reduced combinatorial maps. In this technical report we show that all the darts of the connecting dart sequences may be used to retrieve the reduced combinatorial maps. The resulting algorithms have the same complexity as the algorithms defined in TR-63. But while algorithms defined in TR-63 only compute the reduced combinatorial map defined at a given level i, the proposed algorithms compute all the contracted combinatorial maps up to level i.

The rest of this paper is organized as follows: In section 2 we show that we can suppose without loss of generality that each kernel has a different type than its predecessor. This technical property allows us to suppose that each kernel have a different type than its predecessor during the construction of the pyramid. The following demonstrations are greatly simplified by this assumption. In section 3 we provide theoretical results which allows to determine all the successors of a given dart d surviving up to level i by one scanning of its connecting dart sequence at level i. These results are generalized in section 4 where we determine all the successors of the darts belonging to one connecting dart sequence in one scan of this sequence. These results are then used in section 5 where we provide several algorithms which allows to retrieve all the reduced combinatorial maps defined up to level i by one scanning of the connecting dart sequences defined at this level.

2 From successive to alternating kernels

Connecting dart sequences are defined in TR-63 [7] (Definition 11) using a recursive construction scheme. All the connecting dart sequences at level i are defined from the connecting dart sequences at level $i-1$. However, the exact definition of one connecting dart sequence at level i is relative to the respective types of K_i and K_{i-1}. More precisely, given a 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:

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

[1]The symbol $CDS^*_i(d)$ denotes the connecting dart sequence $CDS_i(d)$ without its first dart d (see TR-63 [7] for more details)

Where $(d_1, \dots, d_p)$ is equal to the connecting walk $CW_i(d)$ of d defined at level i.

The construction of connecting dart sequences is illustrated in Figure 5. This Figure shows the connecting dart sequences defined at each level on the 3×3 grid illustrated in Figure 3. The sequence of kernels K_1, K_2, K_3, K_4 and the resulting reduced combinatorial maps are illustrated in Figure 4. The sequence of kernel is composed of two contraction kernels $K_1 = \alpha^*(1, 2, 7, 10, 12, 6, 4)$ and $K_3 = \alpha^*(3)$ and two removal kernels $K_2 = \alpha^*(9, 8)$ and $K_4 = \alpha^*(5)$

Using the above definition of connecting dart sequences, the exact formulation of the connecting dart sequences at level i from the connecting dart sequences defined at a level $k < i-1$, depends on the different type of kernels from k to i.

In this section, we show that this construction scheme may be simplified by supposing, without loss of generality, that each kernel has a different type than its predecessor. The basic idea of this new construction scheme is to insert between each successive kernels with a same type, an empty kernel having by convention a different type than the two surrounding ones. Note that, the pyramid defined by this new sequence of kernels may be considered as equivalent to the first one only if:

1. The sequence of reduced combinatorial map generated by the two sequence of kernels are the same.
2. The construction of the connecting dart sequences is not altered by the insertion of the empty kernels.

Definition 1 Alternating sequence of kernels

Given an initial combinatorial map G_0, a sequence of successive kernels $K_1, \dots, K_n$ is alternated if $n = 1$ or if any two successive kernels do not have the same type.

In oder to show that two sequences of kernels define the same pyramid we first have to define precisely the notion of Equivalent sequence of kernels:

Definition 2 Equivalent sequence of kernels

Given an initial combinatorial map and two sequences of successive kernels $S = K_1, \dots, K_n$, and $S' = K'_1, \dots, K'_p$. The sequence S' is said to be finer than S iff the sequence of reduced combinatorial maps $G_1, \dots, G_n$ defined

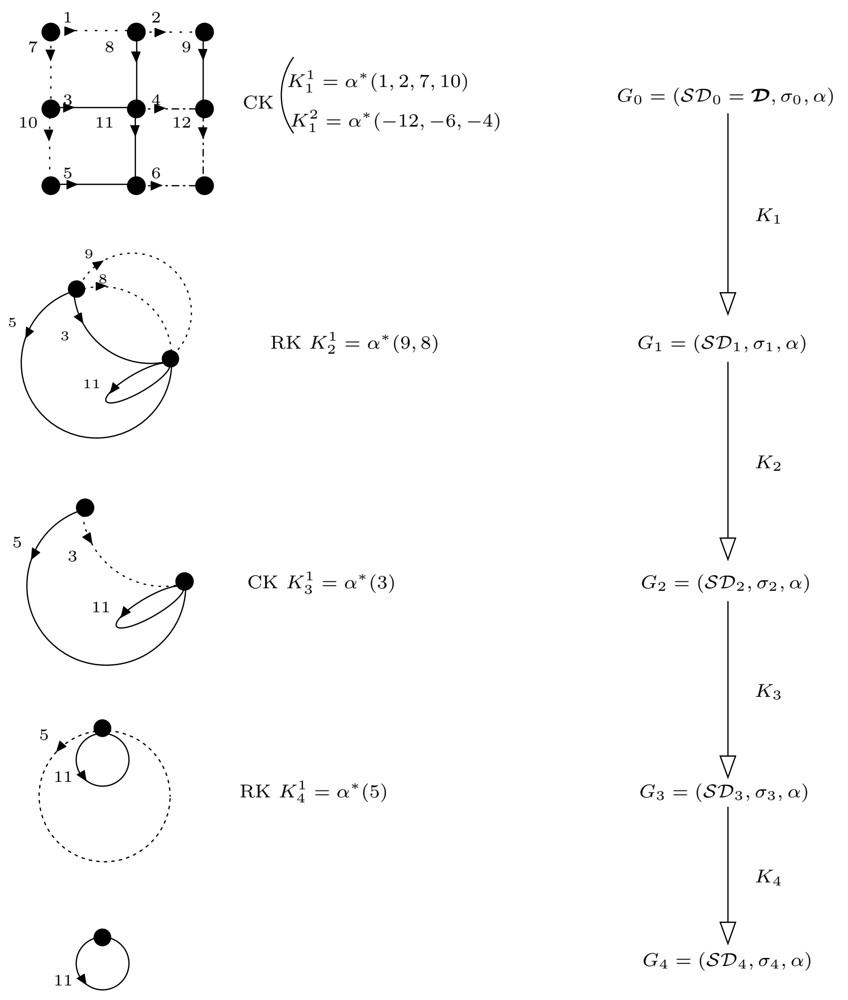
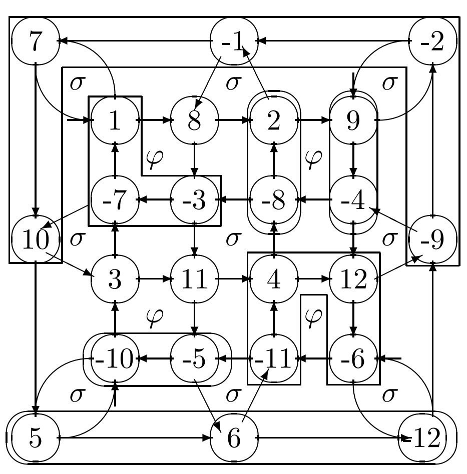
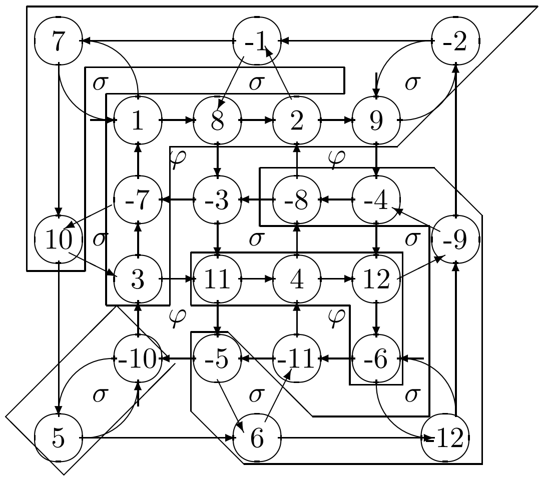

CK $K_1^1 = \alpha^*(1,2,7,10)$
$K_1^2 = \alpha^*(-12,-6,-4)$
RK $K_2^1 = \alpha^*(9,8)$
CK $K_3^1 = \alpha^*(3)$
RK $K_4^1 = \alpha^*(5)$
$G_0 = (\mathcal{SD}_0 = \boldsymbol{\mathcal{D}}, \sigma_0, \alpha)$
K_1
$G_1 = (\mathcal{SD}_1, \sigma_1, \alpha)$
K_2
$G_2 = (\mathcal{SD}_2, \sigma_2, \alpha)$
K_3
$G_3 = (\mathcal{SD}_3, \sigma_3, \alpha)$
K_4
$G_4 = (\mathcal{SD}_4, \sigma_4, \alpha)$

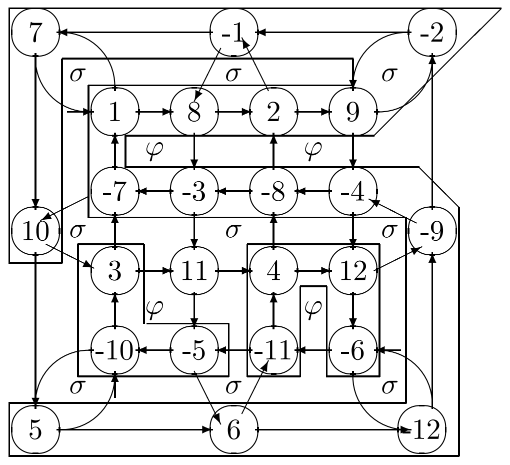
Figure 4: *Successive reductions of an initial* 3×3 *grid. Kernels with even indexes denote contraction kernels (CK) while odd indexes denote removal kernels(RK).*



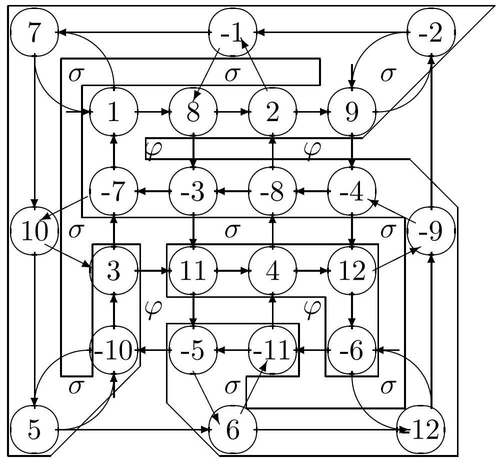
a) level 1



b) level 2



c)level 3



d)level 4

Figure 5: Recursive definition of connection dart sequences

by S is included in the sequence of reduced combinatorial maps $G'_1,\dots,G'_p$ defined by S'.

The sequences S and S' are said equivalent if S is finer than S' and S' is finer than S. In this case $\{G_1,\dots,G_n\} = \{G'_1,\dots,G'_p\}$.

Note that, this last relation between sequences of kernels is defined from an inclusion relation between the sets of reduced combinatorial maps. Therefore, given an initial combinatorial map G_0 and a sequence of kernels S, such that S produces the sequence of reduced combinatorial maps $G_1 \dots G_i \dots G_n$. A sequence of kernels S' producing the sequence of reduced combinatorial maps $(G_1,\dots,G_i,G_i,\dots,G_n)$ is said to be finer than S.

The following propositions establish the expected result: The contraction of a combinatorial map by an empty kernel produces an identical combinatorial map.

Proposition 1 *Given an empty contraction kernel K defined on a combinatorial map $G = (\boldsymbol{\mathcal{D}},\sigma,\alpha)$, we have $G/K = G$.*

Proof:

If K is empty, $\mathcal{SD} = \boldsymbol{\mathcal{D}}$, and the restriction operator $p_{\boldsymbol{\mathcal{D}},\mathcal{SD}}$ (see TR-54 [10] Lemma 1) is equal to the identity of $\boldsymbol{\mathcal{D}}$. Then, if $G/K = (\boldsymbol{\mathcal{D}},\sigma',\alpha)$, we have by definition of the contraction operation (see TR-54 definition 28):

$$\forall d \in \boldsymbol{\mathcal{D}} \quad \sigma'(d) = \varphi' \circ \alpha(d) = \varphi \circ p_{\boldsymbol{\mathcal{D}},\boldsymbol{\mathcal{D}}} \circ \alpha(d) = \varphi(\alpha(d)) = \sigma(d)$$

The two permutations σ' and σ are thus equal and $G/K = G$. □

A same result may be obtained for removal kernels.

Theorem 1 *Any sequence of kernels $K_1,\dots,K_n$ defined on an initial combinatorial map G_0 is equivalent to an alternating sequence of kernels.*

Proof:

From any sequence $S = K_1,\dots,K_n$ let us build a new sequence $S' = K'_1,\dots,K'_p$ such that an empty kernel is inserted between any two successive kernels of S with a same type. The sequence S' is alternate by construction. Moreover, using proposition 1 the two sequences are equivalent. □

The above theorem shows that the same sequence of reduced combinatorial maps may be defined by a sequence of kernels and an equivalent alternating sequence. However, it remains to show that the definition of connecting dart sequences remains consistent for empty kernels. Indeed, the set of reduced combinatorial maps defined by non-empty kernels remaining the same for both sequence of kernels, the set of connecting dart sequences must satisfy the same property.

Proposition 2 *Given a generalized pyramid construction plan defined by an alternating sequence of n kernels. If one kernel K_i with $i \in \{2, \dots, n-1\}$ is empty then the connecting dart sequences at levels $i-1$ and $i+1$ are linked by the following relationship:*

$$CDS_{i+1}(d) = CDS_{i-1}(d)CDS_{i-1}(d_1)\dots CDS_{i-1}(d_p)$$

Where $CW_{i+1}(d) = dd_1, \dots d_p$ is the connecting walk of d defined at level $i+1$.

Proof:

Since K_i and K_{i+1} have not the same type, we have for each dart d in $\mathcal{SD}_{i+1}$:

$$CDS_{i+1}(d) = dCDS_i^*(\alpha(d))d_1CDS_i^*(\alpha(d_1))\dots d_pCDS_i^*(\alpha(d_p))$$

Where $CW_{i+1}(d) = dd_1, \dots d_p$ is the connecting walk of d defined at level $i+1$.

Since $K_i = \emptyset$ by hypothesis, we have $CW_i(d') = d'$ for each dart $d' \in \mathcal{SD}_i$ by definition of connecting walks. Therefore (Figure 6):

$$\forall d' \in \mathcal{SD}_i \; CDS_i(d) = d.CDS_{i-1}^*(\alpha(d))$$

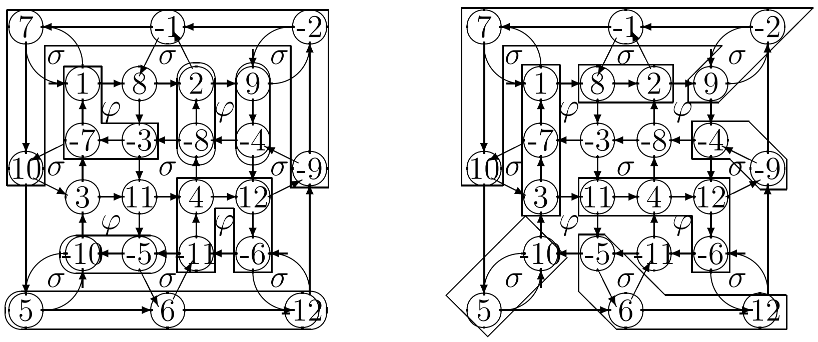
Or, and this is equivalent:

$$\forall d' \in \mathcal{SD}_i \; CDS_i^*(\alpha(d)) = CDS_{i-1}^*(d)$$

Therefore, we obtain for each dart d in $\mathcal{SD}_{i+1}$:

$$\begin{array}{rcl} CDS_{i+1}(d) & = & dCDS_i^*(\alpha(d))d_1CDS_i^*(\alpha(d_1))\dots d_pCDS_i^*(\alpha(d_p)) \\ & = & dCDS_{i-1}^*(d).d_1CDS_{i-1}^*(d_1)\dots d_pCDS_{i-1}^*(d_p) \\ & = & CDS_{i-1}(d)CDS_{i-1}(d_1)\dots CDS_{i-1}(d_p) \end{array}$$

Where $CW_{i+1}(d) = dd_1, \dots d_p$ is the connecting walk of d defined at level $i+1$. $\square$



a) level 1 b) empty kernel

Figure 6: Connecting dart sequences defined by an empty kernel

Note that, since each kernel has by hypothesis a different type than its predecessor, two kernels indexed by $i+1$ and $i-1$ have the same type. Proposition 2 shows that, when one kernel K_i is empty, the relationship which links the connecting dart sequences at level $i+1$ and $i-1$ is the same as the one which relates the connecting dart sequences defined by two successive kernels with the same type (see TR-63 definition 11). This proposition is illustrated by Figure 6, which shows (Figure 6b)) the set of connecting dart sequences defined by an empty kernel.

Therefore, given any sequence of successive kernels, the same sequence of reduced combinatorial maps may be generated by an equivalent alternating sequence of kernels. Moreover, the sequence of connecting dart sequences associated to non-empty kernels is the same in both sequences of kernels. Therefore, we can consider, without loss of generality, that all sequences of kernels are alternating. The adjective alternating will thus be omitted in the following sections. Under this general assumption the definition of a connecting dart sequence is simplified as follows:

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

and

$$\left.\begin{array}{l}\forall i \in \{1,\ldots,n\} \\ \forall d \in \mathcal{SD}_i\end{array}\right\}$$
$$CDS_i(d) = dCDS^*_{i-1}(\alpha(d))d_1CDS^*_{i-1}(\alpha(d_1))\ldots d_pCDS^*_{i-1}(\alpha(d_p)) \quad (1)$$

Where $CW_i(d) = dd_1,\ldots d_p$ is the connecting walk of d defined at level i.

3 Successors of the first dart of a connecting dart sequence

Let us consider the edge $\alpha^*(5)$ in Figures 4 and 5. The kernels K_1 and K_4 encoding contraction operations, the connecting walk associated to a dart d belonging to $\mathcal{SD}_1$ or $\mathcal{SD}_4$ is defined as $CW_i(d) = d.\varphi_{i-1}(d)\ldots,\varphi^{n-1}_{i-1}(d)$ with $i \in \{1,4\}$ and $n = Min\{k \in \mathbb{N}^* \mid \varphi^k_{i-1}(d) \in \mathcal{SD}_i\}$. In the same way, the kernel K_2 being a removal kernel, the connecting walk of a dart $d \in \mathcal{SD}_2$ is defined by $CW_2(d) = d.\sigma_1(d)\ldots\sigma_1^{n-1}(d)$ with $n = Min\{k \in \mathbb{N}^* \mid \sigma_1^k(d) \in \mathcal{SD}_2\}$. The edge $\alpha^*(5)$ being removed at level 4, the darts of this edge define connecting dart sequences up to level 3. We have thus according to equation 1:

$$\begin{aligned} CDS_1(5) &= 5\varphi_0(5)\ldots & (2)\\ CDS_2(-5) &= -5CDS^*_1(5)\sigma_1(-5)\ldots.\text{ and} & (3)\\ CDS_3(5) &= 5CDS^*_2(-5).\varphi_2(5)\ldots. & (4)\end{aligned}$$

If we insert equations 2 and 3 into equation 4 we obtain:

$$CDS_3(5) = 5\varphi_0(5)\ldots.\varphi_1(5)\ldots..\varphi_2(5)\ldots \quad . \quad (5)$$

where $\varphi_1(5)$ has been substituted to $\sigma_1(-5)$ in equation 3.

One traversal of $CDS_3(5)$ may thus allow us to retrieve the φ successors of 5 from level 0 to level 2. Note that, the φ or σ successor of the last dart of $CDS_3(5)$ must be equal to $\varphi_3(5)$ (TR-63, Proposition 17).

Using TR-63 Theorem 1 , we are able to traverse any connecting dart sequence. However, the determination of the φ_i, $i \in \{0,1,2\}$ successors of the dart 5 may be achieved only if we are able to determine when we encounter one of the φ successors of 5 within its connecting dart sequence at level 3. We use to this end the Proposition 13 (TR-63) which shows that the first dart of a connecting dart sequence defined at level i is the only one to survive up to level i. Therefore, the dart $\sigma_1(-5)$ is removed at level 2 by definition of connecting walks and has thus a level equal to 2 (TR-63, Proposition 23). Conversely, all the darts within $CDS_1(5)$ have been contracted at level 1 and have thus a level equal to 1. The dart $\sigma_1(-5)$ is thus the first dart of level 2 that we encounter when we traverse $CDS_2(-5)$. In the same way, $\varphi_2(5)$ is the first dart with level 3 that we encounter when traversing $CDS_3(5)$. The traversal of $CDS_3(5)$ begins thus with $\varphi_0(5)$ (TR-63, Proposition 12) and after the traversal of a series of darts with level 1, the first encountered dart with a level equal to 2 is equal to $\sigma_1(-5) = \varphi_1(5)$. Then after, $\varphi_1(5)$, the first encountered dart with a level greater than 3 is $\varphi_2(5)$.

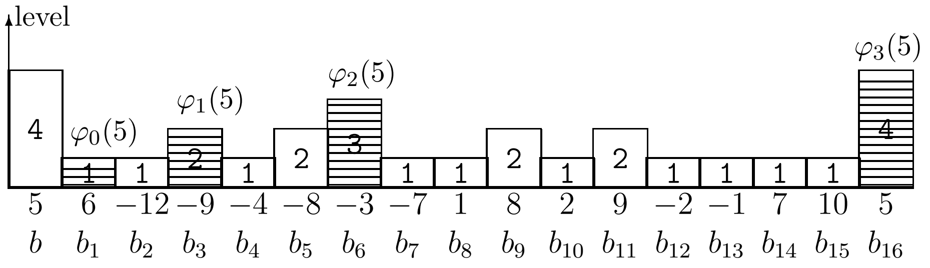


Figure 7: The sequence of darts $CDS_3(5)\varphi_3(5)$. The level of each dart is indicated in its associated bar.

Figure 7 shows the sequence $CDS_3(5)\varphi_3(5)$. The φ successors of 5 are represented by dashed bars. We can note that each φ successor of level j is the first dart of the sequence $CDS^*(5)\varphi_3(5)$ with a level greater than j.

A formal study of these results requires a precise description of the relationships between the connecting dart sequences of one dart at different levels. To this end we have to define the notions of sub word, prefix and suffix within the connecting dart sequences framework:

Definition 3 Sub-word, Prefix, Suffix and Empty sequence

Given two sequences of darts S_1 and S_2, S_1 is said to be a sub-word of S_2 ($S_1 \trianglelefteq S_2$) iff there exist two sequences S_3 and S_4 such that:

$$S_2 = S_3.S_1.S_4$$

The empty dart sequence is denoted by ε. If $S_3.S_4$ is non empty, S_1 is said to be a strict sub-word of S_2. This relation is denoted: $S_1 \lhd S_2$.

- *If S_3 is empty, S_1 is said to be a prefix of S_2 ($S_1 \preceq_{pre} S_2$). If S_4 is non empty, S_1 is a strict prefix of S_2. This relation is denoted: $S_1 \prec_{pre} S_2$.*
- *If S_4 is empty, S_1 is said to be a suffix of S_2 ($S_1 \preceq_{suff} S_2$). If S_3 is non empty, S_1 is said to be a strict suffix of S_2 ($S_1 \prec_{suff} S_2$).*

Remark 1 *In the following the notions of sub-word, prefix and suffix (Definition 3) will only be used for connecting dart sequences. However, since these notions do not require any particular properties of connecting dart sequences we defined them on general sequences of dart.*

Given a connecting dart sequence $CDS_i(d)$ defined at level i, the determination of the σ or φ successors of d at levels $j < i$ is based on a determination of the sequences $CDS_j(d)$ or $CDS_j(\alpha(d))$ included in $CDS_i(d)$. For example, the determination of $\varphi_2(5)$ from $CDS_3(5)$ requires to determine the relationship between $CDS_3(5)$ and $CDS_2(-5)$ (equation 4). The following proposition states which connecting dart sequence ($CDS_j(d)$ or $CDS_j(\alpha(d))$) is included in $CDS_i(d)$ according to the parity of $i - j$.

Proposition 3 *Given a generalized pyramid construction plan defined by n kernels, for each connecting dart sequence $CDS_i(d)$ defined by d at level $i < level(d)$, for each $j \in \{0, \dots, i\}$:*

- *If $(i - j)$ mod $2 = 0$, $CDS_j(d) \preceq_{pre} CDS_i(d)$*
- *If $(i - j)$ mod $2 = 1$, $d.CDS^*_j(\alpha(d)) \preceq_{pre} CDS_i(d)$*

Proof:

Let us make a recurrence on j, the proposition is trivial if $j = i$. Let us suppose it true for a given j. Since d is not contracted before level j, we can consider the two following connecting walks:

$$\begin{cases} CW_j(d) & = & d.d_1, \dots, d_p \\ CW_j(\alpha(d)) & = & \alpha(d).b_1, \dots, b_q \end{cases}$$

- If $(i-j)\ mod\ 2\ =\ 0$, then we have by our recurrence hypothesis: $CDS_j(d) \preceq_{pre} CDS_i(d)$ and:

  $$CDS_j(d) = dCDS^*_{j-1}(\alpha(d)).d_1CDS^*_{j-1}(\alpha(d_1))\dots d_pCDS^*_{j-1}(\alpha(d_p))$$

  Therefore:

  $$dCDS^*_{j-1}(\alpha(d)) \preceq_{pre} CDS_j(d) \preceq_{pre} CDS_i(d)$$

  with $(i-j-1)\ mod\ 2\ =\ 1$.

- If $(i-j)\ mod\ 2\ =\ 1$, then we have by our recurrence hypothesis: $dCDS^*_j(\alpha(d)) \preceq_{pre} CDS_i(d)$ and:

  $$CDS_j(\alpha(d)) = \alpha(d)CDS^*_{j-1}(d).b_1CDS^*_{j-1}(\alpha(b_1))\dots b_qCDS^*_{j-1}(\alpha(b_p))$$

  Thus:

  $$CDS_{j-1}(d) \preceq_{pre} dCDS^*_j(\alpha(d)) \preceq_{pre} CDS_i(d),$$

  with $(i-j-1)\ mod\ 2 = 0$.

$\square$

The above proposition may be illustrated using once again the connecting dart sequence $CDS_3(5)$. Indeed, we have :

$$\begin{array}{lcl} CDS_3(5) & = & 5.6.-12.-9.-4.-8.-3.-7.1.8.2.9.-2.-1.7.10 \\ 5.CDS^*_2(-5) & = & 5.6.-12.-9.-4.-8 \\ CDS_1(5) & = & 5.6.-12 \end{array}$$

Corollary 1 *Given a pyramid construction plan defined by n kernels, and a dart d in $\mathcal{SD}_i$ such that $CDS^*_i(d) = \varepsilon$, then:*

$$\forall j \in \{1,\dots,i\} \begin{cases} CDS^*_j(d) = \varepsilon & \text{If } (i-j)\ mod\ 2 = 0 \\ CDS^*_j(\alpha(d)) = \varepsilon & \text{If } (i-j)\ mod\ 2 = 1 \end{cases}$$

Proof:

Let us consider a given $j \in \{1,\dots,i\}$.

If $(i-j)\ mod\ 2$=0, we have by proposition 3: $CDS_j(d) \preceq_{pre} CDS_i(d)$. Therefore, if $CDS^*_i(d) = \varepsilon$, $CDS_i(d) = d$ and $CDS_j(d)$ must be equal to d. Thus $CDS^*_j(d) = \varepsilon$.

If $(i-j)\ mod\ 2$=1, we have by proposition 3: $dCDS^*_j(\alpha(d)) \preceq_{pre} CDS_i(d)$. Then, if $CDS_i(d) = d$, $CDS^*_j(\alpha(d))$ must be empty. $\square$

Using the example displayed in Figure 5, we have $CDS_3(11) = \varepsilon$ (Figure 5(c)). Therefore, using the above corollary we have $CDS_2(-11) = \varepsilon$ (Figure 5(b)) and $CDS_1(11) = \varepsilon$ (Figure 5(a)).

The examples of equations 2 to 5 show that the first dart with a level greater than j in $CDS_i(d)$ with $j < i$ is a promising tool for finding the φ or σ successor of d at level j. In the following we will often have to search for such darts. However, we first have to define on which conditions such darts exist. The following definition set the maximum level which may be encountered in a given non empty connecting dart sequence.

Definition 4 MaxLevel function

Given a pyramid construction plan defined by n kernels, the MaxLevel function L_i maps each dart $d \in \mathcal{SD}_i$ with $CDS^*_i(d) \neq \varepsilon$ to the greatest level contained in $CDS^*_i(d)$.

$$L_i(d) = \max_{d' \in CDS^*_i(d)} \{level(d')\}$$

Remark 2 *Note that, if $L_i(d)$ is defined for a given $i \leq n$ and $d \in \mathcal{SD}_{i+1}$, then $L_{i+1}(\alpha(d))$ is also defined. Indeed, using corollary 1, if $CDS^*_i(d)$ is non empty, $CDS^*_{i+1}(\alpha(d))$ is also non empty.*

The maximum level encountered in $CDS_3(5)$ being equal to 3 we have $L_3(5) = 3$. In the same way, we have $CDS_2(5) = 5.-10$ with $level(-10) = 1$ (Figure 5) thus $L_2(5) = 1$. The maxlevel function L_i seems thus to be bounded by i. The following proposition confirms this result:

Proposition 4 *Given a pyramid construction plan defined by n kernels, the MaxLevel function L_i satisfies the following properties:*

1. *For each i in $\{1, \ldots, n\}$, L_i is bounded by i.*

$$\forall i \in \{1, \ldots, n\},\ \forall d \in \mathcal{SD}_i \mid CDS^*_i(d) \neq \varepsilon \quad L_i(d) \leq i$$

2. *For each i in $\{1, \ldots, n-1\}$, and for each dart d in $\mathcal{SD}_{i+1}$, such that $CDS^*_i(d) \neq \varepsilon$, $L_i(d)$ is less than $L_{i+1}(\alpha(d))$:*

$$\forall i \in \{1, \ldots, n-1\}, \quad \forall d \in \mathcal{SD}_{i+1} \mid CDS^*_i(d) \neq \varepsilon \quad L_i(d) \leq L_{i+1}(\alpha(d))$$

Proof:

The first property, is a direct consequence of TR-63 [7] proposition 13. Indeed, we have:

$$\forall i \in \{1, \ldots, n\} \quad \forall d \in \mathcal{SD}_i \quad CDS^*_i(d) \subset \bigcup_{j=0}^{i} K_j$$

Therefore, using TR-63 proposition 23:

$$\forall d' \in CDS^*_i(d), \quad \mathit{level}(d') \leq i \Rightarrow L_i(d) \leq i$$

Given a level i in $\{1, \ldots, n-1\}$, and a dart d in $\mathcal{SD}_{i+1}$, we have by proposition 3:

$$CDS^*_i(d) \preceq_{pre} CDS^*_{i+1}(\alpha(d)) \Rightarrow L_i(d) \leq L_{i+1}(\alpha(d))$$

□

Given a pyramid construction plan defined on an initial combinatorial map $G_0 = (\boldsymbol{\mathcal{D}}, \sigma, \alpha)$ by a sequence of kernels $K_1, \ldots, K_n$, the set of connecting dart sequences forms a partition of $\boldsymbol{\mathcal{D}}$ at each level $i \in \{1, \ldots, n\}$(see TR-63 proposition 18). Since the set of surviving darts decreases from levels to levels, the mean size of connecting dart sequences defined at each level increases. However, the growing of each connecting dart sequence is not uniform and some of them may remain unchanged for several levels. In the following we show that such connecting dart sequences may be characterized by $L_i(d) < i$. We also study the properties of these connecting dart sequences.

Proposition 5 *Given a pyramid construction plan defined by n kernels, for each connecting dart sequence $CDS_i(d) \neq d$ defined by d at level $i < \mathit{level}(d)$, if $L_i(d) < i$ the following property holds:*

$$\forall j \in \{L_i(d)+1, \ldots, i\} \left\{ \begin{array}{ll} CDS^*_j(d) = CDS^*_{j-1}(\alpha(d)) & \textit{If } (i-j) \textit{ mod } 2{=}0 \\ CDS^*_j(\alpha(d)) = CDS^*_{j-1}(d) & \textit{If } (i-j) \textit{ mod } 2{=}1 \end{array} \right.$$

Proof:

Let us consider a given j in $\{L_i(d)+1, \ldots, i\}$.

- If $(i-j)$ *mod* 2=0, $CDS_j(d) \preceq_{pre} CDS_i(d)$ (Proposition 3). Moreover:

$$CDS_j(d) = dCDS^*_{j-1}(\alpha(d))d_1 CDS^*_{j-1}(\alpha(d_1)) \ldots d_p CDS^*_{j-1}(\alpha(d_p))$$

With $CW_j(d) = dd_1, \dots d_p$.

Since $level(d_1) = j \in \{L_i(d)+1, \dots, i\}$, the maximum level inside $CDS^*_j(d) \preceq_{pre} CDS_i(d)$ do not exceed $L_i(d)$ only if $p = 0$. Therefore $CDS_j(d) = dCDS^*_{j-1}(\alpha(d))$ and:

$$CDS^*_j(d) = CDS^*_{j-1}(\alpha(d))$$

- If $(i-j)$ mod 2=1, we have: $dCDS^*_j(\alpha(d)) \preceq_{pre} CDS_i(d)$ (Proposition 3). Moreover:

$$CDS_j(\alpha(d)) = \alpha(d)CDS^*_{j-1}(d)b_1CDS^*_{j-1}(\alpha(b_1))\dots b_qCDS^*_{j-1}(\alpha(b_q))$$

  With $CW_j(\alpha(d)) = db_1, \dots b_q$.

  As previously, the two properties $CDS^*_j(\alpha(d)) \preceq_{pre} CDS_i(d)$ and $j > L_i(d)$ may be simultaneously true only if $q = 0$, we thus obtain:

$$CDS^*_j(\alpha(d)) = CDS^*_{j-1}(d)$$

□

Given one dart $d \in \mathcal{SD}_i$ such that $L_i(d) < i$, Proposition 5 shows that for $j \in \{L_i(d), \dots, i\}$ the growth of $CDS_j(d)$ and $CDS_j(\alpha(d))$ must be alternatively blocked according to the parity of $i-j$ in order to produce the connecting dart sequence $CDS_i(d)$ whose max level does not exceed $L_i(d)$. In other words, these connecting dart sequences have reached a "plateau". The first property of Corollary 2 uses this result to show that for $j \in \{L_i(d), \dots, i\}$, the connecting dart sequences $CDS^*_j(d)$ with (i-j) even and $CDS^*_j(\alpha(d))$ with (i-j) odd must be equal to $CDS^*_i(d)$. This result is thus another formulation of the fact that the growth of the connecting dart sequences between levels $L_i(d)$ and i must be blocked until level i. The second property of Corollary 2 uses this last formulation in conjunction with TR-63, proposition 17 to show that either the φ of σ successor of the dart d should remain the same from level $L_i(d)$ to level i.

Corollary 2 *Given a pyramid construction plan defined by n kernels, for each connecting dart sequence $CDS_i(d) \neq (d)$ defined by d at level $i < level(d)$ the following properties hold:*

1. *Plateau of connecting dart sequences.*

$$\forall j \in \{L_i(d), \dots, i\} \left\{ \begin{array}{lcll} CDS^*_j(d) & = & CDS^*_i(d) & \textit{If } (i-j) \textit{ mod } 2{=}0 \\ CDS^*_j(\alpha(d)) & = & CDS^*_i(d) & \textit{If } (i-j) \textit{ mod } 2{=}1 \end{array} \right.$$

2. *Shrinkage blocking for $L_i(d) < i$.*

   The φ or σ successors remain unchanged at each level j in $\{L_i(d), \ldots, i\}$ according to the type of K_i.

$$\forall j \in \{L_i(d), \ldots, i\} \quad \left\{ \begin{array}{lcll} \varphi_j(d) & = & \varphi_i(d) & \textit{If } K_i \textit{ is a contraction kernel} \\ \sigma_j(d) & = & \sigma_i(d) & \textit{If } K_i \textit{ is a removal kernel} \end{array} \right.$$

Proof:

Note that if $L_i(d) = i$ the above properties are trivial. Let us thus suppose that $L_i(d) < i$. The first property may be easily deduced by iterating the equalities defined in proposition 5. Let us now consider $j \in \{L_i(d), \ldots, i\}$ such that $(i-j)$ *mod* 2= 0, then K_i and K_j have the same type, using TR-63 proposition 17,

- If K_i (and K_j) is a contraction kernel:

$$\varphi_i(d) = \varphi_j(d) = \left\{ \begin{array}{ll} \varphi(d') & \text{If } d' \text{ is contracted} \\ \sigma(d') & \text{If } d' \text{ is removed} \end{array} \right.$$

  where d' denotes the last dart of $CDS_i(d) = CDS_j(d)$.

- In the same way, if K_i is a removal kernel, we obtain we the same notations:

$$\sigma_i(d) = \sigma_j(d) = \left\{ \begin{array}{ll} \varphi(d') & \text{If } d' \text{ is contracted} \\ \sigma(d') & \text{If } d' \text{ is removed} \end{array} \right.$$

If $(i-j)$ *mod* 2=1, K_i and K_j have different types, thus using TR-63-proposition 17:

- If K_i is a contraction kernel, K_j is a removal kernel and:

$$\varphi_i(d) = \sigma_j(\alpha(d)) = \varphi_j(d) = \left\{ \begin{array}{ll} \varphi(d') & \text{If } d' \text{ is contracted} \\ \sigma(d') & \text{If } d' \text{ is removed} \end{array} \right.$$

  where d' is the last dart of $CDS_i(d) = d.CDS_j^*(\alpha(d))$.

- If K_i is a removal kernel, we obtain with the same notations:

$$\sigma_i(d) = \varphi_j(\alpha(d)) = \sigma_j(d) = \left\{ \begin{array}{ll} \varphi(d') & \text{If } d' \text{ is contracted} \\ \sigma(d') & \text{If } d' \text{ is removed} \end{array} \right.$$

□

The second point of Corollary 2 provides an intuitive explanation of the "plateau' phenomenon states in the first property of the same corollary. Indeed, in such a case ($L_i(d) < i$), $CDS_j(d)$ and $CDS_j(\alpha(d))$ are alternatively blocked by either $\varphi_i(d)$ or $\sigma_i(d)$. These darts are reduced after level i and can't thus belong to a connecting walk before this level. In other words, the connecting dart sequences "want" to pass by one of these dart but they can do so only after $level(d)$ and are thus blocked until this level.

Remark 3 *If the first property of Corollary 2 holds, $L_i(d)$ is equal to $L_j(d)$ if $(i-j)$ mod $2 = 0$ and $L_j(\alpha(d))$ if $(i-j)$ mod $2 = 1$. In both cases we obtain $L_i(d) \leq j < i$. The implication of Proposition 5 is thus an equivalence.*

Moreover, note that, the second property of Corollary 2, is equivalent to:

$$\forall k \in \{L_i(d)+1, \ldots, i+1\} \quad \begin{cases} \varphi_{k-1}(d) = \varphi_i(d) & \text{If } K_i \text{ is a contraction kernel} \\ \sigma_{k-1}(d) = \sigma_i(d) & \text{If } K_i \text{ is a removal kernel} \end{cases}$$

with $k = j+1$.

This last notation will be used in the following.

The MaxLevel function $L_i(d)$ allows us to encode the maximum level which may be encountered in the connecting dart sequence $CDS_i^*(d)$. Therefore, if $L_i(d)$ is defined we can ensure, for any $j \leq L_i(d)$ that at least one dart with a level greater or equal to j belongs to $CDS_i^*(d)$. The use of this function already allows us to retrieve the successors of a given dart for all levels inside $\{L_i(d), \ldots, i\}$ (see corollary 2). In order to retrieve the successors at the other levels (i.e. in $\{0, \ldots, L_i(d)-1\}$), we need to define more precisely the first dart with a level greater than $j \leq L_i(d)$ encountered when traversing $CDS_i^*(d)$:

Definition 5 IndexLevel function

Given a pyramid construction plan defined by n kernels, a level i in $\{1, \ldots, n\}$, and one dart d in $\mathcal{SD}_i$, the connecting dart sequence of which is still empty at level i, the index-level function $l_{i,d}$ maps each level $j \in \{1, \ldots, L_i(d)\}$ to the index of the first dart in $CDS_i(d)$ with a level greater or equal to j:

$$l_{i,d} \left(\begin{array}{rcl} \{1, \ldots, L_i(d)\} & \rightarrow & \mathbb{N}^* \\ j & \mapsto & \min\{k \in \{1, \ldots, p\} \quad | \quad level(d_k) \geq j\} \end{array} \right.$$

with $CDS_i(d) = d.d_1 \ldots d_p$, $p > 0$.

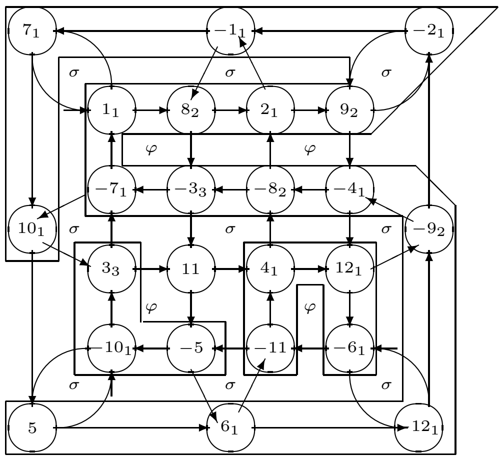


Figure 8: Connecting dart sequences at level 3. The index of each non surviving dart shows its level.

Figure 8 shows the third level of the pyramid displayed in Figure 5. The level of each non surviving dart is presented on this figure as an index (see also Figure 7). The greatest level encountered in $CDS_3^*(-11) = 4.12. - 6$ is 1. We have thus $L_3(-11) = 1$. In the same way, the dart -3 with a level 3 belongs to $CDS_3(5)$. Since $L_3(5) \leq 3$ (Proposition 4) we have $L_3(5) = 3$. The first darts with a level greater or equal to $1, 2$ and 3 encountered when traversing $CDS_3(5)$ are respectively 6(level 1),-9(level 2) and -3(level 3). The indices of these darts in $CDS_3(5)$ being respectively $1, 3$ and 6 (Figure 7 and 9) we have:

$$l_{3,5}(1) = 1,\ l_{3,5}(2) = 3,\ l_{3,5}(3) = 6$$

Note that using Figure 7, we observe (see also Figure 4) that:

$$\begin{array}{rcccccr} \varphi_0(5) & = & b_{l_{3,5}(1)} & = & b_1 & = & 6 \\ \varphi_1(5) & = & b_{l_{3,5}(2)} & = & b_3 & = & -9 \\ \varphi_2(5) & = & b_{l3,5(3)} & = & b_6 & = & -3. \end{array}$$

The remaining of this section provides an exhaustive proof of this observation.

Using equation 1 the connecting dart sequence at level $i+1$ of $\alpha(d) \in \mathcal{SD}_i$

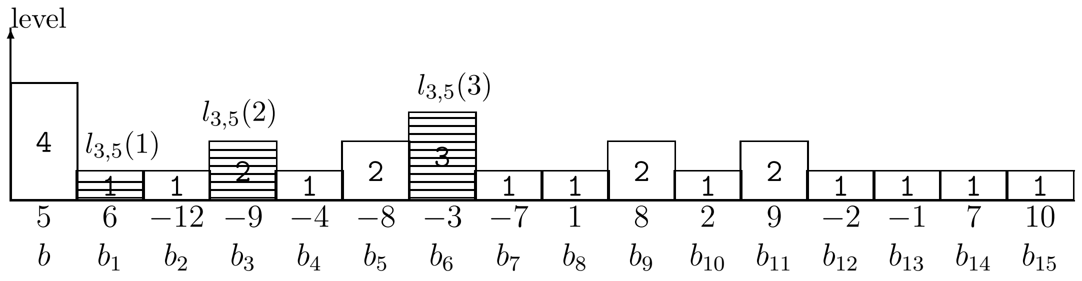


Figure 9: The index levels of the dart 5 at level 3 superimposed to $CDS_3(5)$. The level of each dart is indicated in its associated bar.

is defined by:

$$CDS_{i+1}(\alpha(d)) = \alpha(d)CDS^*_i(d)\dots \tag{6}$$

Let d' be the first dart with a level greater than j encountered when traversing $CDS^*_i(d)$. Using equation 6 this dart is also the first dart with a level greater than j encountered when traversing $CDS^*_i(\alpha(d))$. The following proposition confirms this result:

Proposition 6 *Given a pyramid construction plan defined by n kernels, the index-level function satisfies the following property:*

$$\left.\begin{array}{l}\forall i \in \{1,\dots,n-1\} \\ \forall d \in \mathcal{SD}_{i+1} \mid CDS^*_i(d) \neq \varepsilon\end{array}\right\} \forall j \in \{1,\dots,L_i(d)\} \quad l_{i,d}(j) = l_{i+1,\alpha(d)}(j)$$

Proof:

First note, that the condition $CDS^*_i(d) \neq \varepsilon$ insures that $L_i(d)$ and $L_{i+1}(\alpha(d))$ are well defined. Using proposition 4, $L_i(d) \leq L_{i+1}(\alpha(d))$. Therefore, $l_{i+1,\alpha(d)}(j)$ is well defined for all $j \in \{1,\dots,L_i(d)\}$. Moreover, using proposition 3, $CDS^*_i(d)$ is a prefix of $CDS^*_{i+1}(\alpha(d))$. Therefore, $d_{l_{i,d}(j)}$ is the first dart with a level greater or equal to j encountered when traversing $CDS^*_{i+1}(\alpha(d))$ and its index is the same in $CDS_i(d)$ and $CDS_{i+1}(\alpha(d))$. We have thus $l_{i,d}(j) = l_{i+1,\alpha(d)}(j)$. $\square$

This last proposition is illustrated in Figure 10 which represents the connecting dart sequences of -3 and $\alpha(-3)$ respectively at levels 1 and 2. The greatest level encountered in $CDS^*_1(-3)$ being equal to 1 we have $L_1(-3) = 1$. Proposition 6 can thus be applied only on the index $l_{1,-3}(1)$. Since $CDS^*_1(-3)$ is a prefix of $CDS_2(3)$ we have as stated by Proposition 6: $l_{2,3}(1) = l_{1,-3}(1)$.

We have shown in Proposition 15 (TR-63) that if a connecting dart sequence satisfies $CDS_i(d) = d$ we have either $\varphi_i(d) = \varphi(d)$ or $\sigma_i(d) = \sigma(d)$

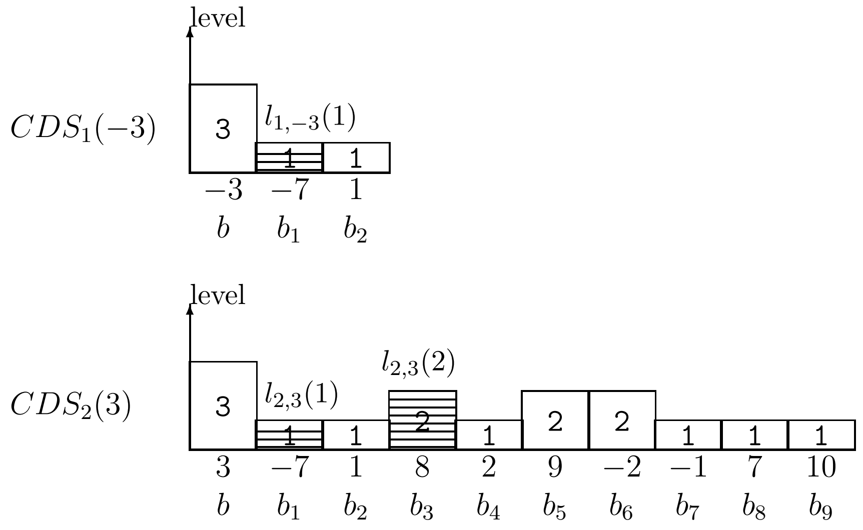


Figure 10: The connecting dart sequences $CDS_1(-3)$ and $CDS_2(3)$. As mentioned in Proposition 6 we have $l_{1,-3}(1) = l_{2,3}(1)$

according to the type of K_i. On the other hand, we have shown in Corollary 1 of this report that if $CDS_i(d) = d$, we have for all $j \in \{1, \dots, i\}$ either $CDS_j(d) = d$ or $CDS_j(\alpha(d)) = \alpha(d)$. The following theorem combines these two results:

Theorem 2 *Given a pyramid construction plan defined by n kernels, for each connecting dart sequence $CDS_i(d) = (d)$ defined by d at level $i < level(d)$:*

- *If K_i is a contraction kernel:*

$$\forall j \in \{0, \dots, i\} \quad \varphi_j(d) = \varphi(d)$$

- *If K_i is a removal kernel:*

$$\forall j \in \{0, \dots, i\} \quad \sigma_j(d) = \sigma(d)$$

Proof:

Let us suppose that K_i is a contraction kernel. Given a dart $d \in \mathcal{SD}_i$ such that $CDS_i(d) = d$ and a level j in $\{0, \dots, i\}$:

- If $(i - j)$ *mod* 2=0, K_j is a contraction kernel and we have by Corollary 1: $CDS_j(d) = d$. Therefore, using TR-63 proposition 15:

$$\varphi_j(d) = \varphi(d)$$

- If $(i-j)$ *mod* 2=1, K_j is a removal kernel and $CDS_j(\alpha(d)) = \alpha(d)$ (see Corollary 1). Therefore, using TR-63 proposition 15:

$$\sigma_j(\alpha(d)) = \varphi_j(d) = \sigma(\alpha(d)) = \varphi(d)$$

In the same way, if K_i is a removal kernel:

- If $(i-j)$ *mod* 2=0, K_j is a removal kernel and $CDS_j(d) = d$. Therefore, using TR-63 proposition 15:

$$\sigma_j(d) = \sigma(d)$$

- If $(i-j)$ *mod* 2=1, K_j is a contraction kernel and $CDS_j(\alpha(d)) = \alpha(d)$. Therefore:

$$\varphi_j(\alpha(d)) = \sigma_j(d) = \varphi(\alpha(d)) = \sigma(d)$$

□

Using the example displayed in Figure 4, we have $CDS_3(11) = 11$ (Figure 5) with K_3 being a contraction kernel. Using Theorem 2 we obtain (Figure 4):

$$\varphi_1(11) = \varphi_2(11) = \varphi_3(11) = \varphi(11) = -5$$

Note that, if K_i is a contraction kernel, $CDS_i(d) = d$ only if $\varphi(d)$ has a level greater than i. For example, $CDS_3(11)$ is reduced to 11 due to the dart -5 whose level is equal to 3. Therefore, this theorem establishes one quite natural result: If one dart d and its φ-successor $\varphi(d)$ are not reduced until level i, the φ successor of d remains unchanged until this level.

In the more general case ($CDS_i(d) \neq (d)$), the following theorem allows us to retrieve the successors of a dart $d \in \mathcal{SD}_i$ for all levels which are not covered by corollary 2:

Theorem 3 *Given a pyramid construction plan defined by n kernels, for each connecting dart sequence $CDS_i(d) \neq (d)$ defined by d at level $i < level(d)$:*

$$\forall j \in \{1, \ldots, L_i(d)\} \begin{cases} \varphi_{j-1}(d) = d_{l_{i,d}(j)} & \textit{If } K_i \textit{ is a contraction kernel} \\ \sigma_{j-1}(d) = d_{l_{i,d}(j)} & \textit{If } K_i \textit{ is a removal kernel} \end{cases}$$

Proof:

Let us show this proposition by a recurrence on i. If $i = 1$ and $CDS_1(d) \neq (d)$, $L_1(d) = 1$ and $CDS_1(d)$ may be written as:

$$CDS_1(d) = d.d_1, \ldots, d_p \; with \; p > 0$$

By definition of connecting dart sequence, we must have $level(d_1) = 1$, therefore $l_{1,d}(1) = 1$, and (TR-63-Proposition 16):

- $d_1 = \varphi_0(d)$ If K_1 is a contraction kernel.
- $d_1 = \sigma_0(d)$ If K_1 is a removal kernel.

The property is thus true at rank 1. Let us suppose it is true for all darts defined at level $i - 1 \in \{0, \ldots, n-1\}$. Given a dart d in $\mathcal{SD}_i$ such that $CDS^*_i(d) \neq \varepsilon$:

$$\begin{array}{rcl} CDS_i(d) & = & d.d_1.\ldots.d_p \\ CDS_i(d) & = & d.CDS^*_{i-1}(\alpha(d))b_1CDS^*_{i-1}(\alpha(b_1))\ldots b_qCDS^*_{i-1}(\alpha(b_q)) \end{array}$$

with $CW_i(d) = d.b_1 \ldots, b_q$.

If $CDS^*_{i-1}(\alpha(d)) = \varepsilon$, since $CDS^*_i(d) \neq \varepsilon$ we must have $q > 0$ and $b_1 = d_1$. Since b_1 is contracted at level i we have $level(b_1) = i$ and thus $L_i(d) = i$. Moreover, by definition of the function $l_{i,d}$:

$$\forall j \in \{1, \ldots, i\} \quad l_{i,d}(j) = 1$$

Using the definition of connecting walks b_1 is equal to:

- $\varphi_{i-1}(d)$ if K_i is a contraction kernel and,
- $\sigma_{i-1}(d)$ is K_i is a removal kernel.

Since K_i and K_{i-1} have not the same type, Theorem 2 applied on $CDS_{i-1}(\alpha(d))$ provides the following equations:

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

$$\forall l \in \{0, \ldots, i-1\}\sigma_l(\alpha(d)) = \varphi_l(d) = \sigma_{i-1}(\alpha(d)) = \varphi_{i-1}(d) = b_1$$

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

$$\forall l \in \{0, \ldots, i-1\}\varphi_l(\alpha(d)) = \sigma_l(d) = \varphi_{i-1}(\alpha(d)) = \sigma_{i-1}(d) = b_1$$

Using $j = l + 1$ and the fact that $l_{i,d}(j)$ is equal to 1 for all j, we obtain the expected result:

$$\forall j \in \{1, \ldots, L_i(d)\} \left\{ \begin{array}{ll} \varphi_{j-1}(d) = d_{l_{i,d}(j)} & \text{If } K_i \text{ is a contraction kernel} \\ \sigma_{j-1}(d) = d_{l_{i,d}(j)} & \text{If } K_i \text{ is a removal kernel} \end{array} \right.$$

If $CDS^*_{i-1}(\alpha(d)) \neq \varepsilon$, using propositions 4 and 6 $L_i(d) \geq L_{i-1}(\alpha(d))$, and:

$$\forall j \in \{1, \ldots, L_{i-1}(\alpha(d))\} \quad l_{i-1,\alpha(d)}(j) = l_{i,d}(j)$$

Using our recurrence hypothesis on $CDS_{i-1}(\alpha(d))$:

$$\forall j \in \{1, \ldots, L_{i-1}(\alpha(d))\} \left\{ \begin{array}{ll} \text{If} & K_{i-1} \text{ is a removal kernel:} \\ & \sigma_{j-1}(\alpha(d)) = d_{l_{i-1,\alpha(d)}(j)} = d_{l_{i,d}(j)} \\ \text{If} & K_{i-1} \text{ is a contraction kernel:} \\ & \varphi_{j-1}(\alpha(d)) = d_{l_{i-1,\alpha(d)}(j)} = d_{l_{i,d}(j)} \end{array} \right.$$

Since K_i and K_{i-1} have not the same type, we obtain:

$$\forall j \in \{1, \ldots, L_{i-1}(\alpha(d))\} \left\{ \begin{array}{ll} \text{If} & K_i \text{ is a contraction kernel} \\ & \varphi_{j-1}(d) = d_{l_{i,d}(j)} \\ \text{If} & K_i \text{ is a removal kernel} \\ & \sigma_{j-1}(d) = d_{l_{i,d}(j)} \end{array} \right. \tag{7}$$

If $q = 0$, we have $CDS_i(d) = d.CDS_{i-1}(\alpha(d))$ and thus $L_i(d) = L_{i-1}(\alpha(d))$. In this case, nothing remains to be demonstrated at level i for the dart d.

If $q \neq 0$, let us consider k such that $b_1 = d_k$. We have by definition of a connecting walk $level(d_k) = i$ thus $L_i(d) = i$. Moreover, using proposition 4, $L_{i-1}(\alpha(d)) \leq i - 1$, therefore $l_{i,d}(i) = k$.

- If $L_{i-1}(\alpha(d)) = i - 1$, we have only to show that equation 7 remains valid for $j = L_i(d) = i$. Using the definition of a connecting walk:

  $$\left\{ \begin{array}{lcll} \varphi_{i-1}(d) & = & d_k & \text{If } K_i \text{ is a contraction kernel} \\ \sigma_{i-1}(d) & = & d_k & \text{If } K_i \text{ is a removal kernel} \end{array} \right.$$

  with $k = l_{i,d}(i)$. Thus, using equation 7:

  $$\forall j \in \{1, \ldots, L_i(d)\} \left\{ \begin{array}{lcll} \varphi_{j-1}(d) & = & d_{l_{i,d}(j)} & \text{If } K_i \text{ is a contraction kernel} \\ \sigma_{j-1}(d) & = & d_{l_{i,d}(j)} & \text{If } K_i \text{ is a removal kernel} \end{array} \right.$$

- If $L_{i-1}(\alpha(d)) < i-1$. Then, by definition of function L_i, all the darts of $CDS_{i-1}(\alpha(d))$ have a level less than $L_{i-1}(\alpha(d))$. Moreover, since:

  $$CDS_i^*(d) = CDS_{i-1}^*(\alpha(d))b_1CDS_{i-1}^*(\alpha(b_1))\dots b_qCDS_{i-1}^*(\alpha(b_q))$$

  $d_k = b_1$ is the first dart of $CDS_i^*(d)$ with a level strictly greater than $L_{i-1}(\alpha(d))$. We have thus:

  $$\forall j \in \{L_{i-1}(\alpha(d))+1,\dots,i\} \quad l_{i,d}(j) = k \tag{8}$$

  Since $L_{i-1}(\alpha(d)) < i-1$, using Corollary 2 and remark 3 with $CDS_{i-1}(\alpha(d))$ we obtain:

  $$\forall j \in \{L_{i-1}(\alpha(d))+1,\dots,i\} \left\{ \begin{array}{ll} \text{If } K_{i-1} & \text{is a removal kernel} \\ & \sigma_{j-1}(\alpha(d)) = \sigma_{i-1}(\alpha(d)) = d_k \\ \text{If } K_{i-1} & \text{is a contraction kernel} \\ & \varphi_{j-1}(\alpha(d)) = \varphi_{i-1}(\alpha(d)) = d_k \end{array} \right.$$

  Since K_i and K_{i-1} have not the same type, using equation 8, the above equation may be written as:

  $$\forall j \in \{L_{i-1}(\alpha(d))+1,\dots,i\} \left\{ \begin{array}{ll} \text{If } K_i & \text{is a contraction kernel} \\ & \varphi_{j-1}(d) = d_k = d_{l_{i,d}(j)} \\ \text{If } K_i & \text{is a removal kernel} \\ & \sigma_{j-1}(d) = d_k = d_{l_{i,d}(j)} \end{array} \right.$$

  Since $L_i(d) = i$, the above equation, combined with equation 7 provides the expected result:

  $$\forall j \in \{1,\dots,L_i(d)\} \left\{ \begin{array}{ll} \text{If} & K_i \text{ is a contraction kernel} \\ & \varphi_{j-1}(d) = d_{l_{i,d}(j)} \\ \text{If} & K_i \text{ is a dual contraction kernel} \\ & \sigma_{j-1}(d) = d_{l_{i,d}(j)} \end{array} \right.$$

□

Note that Theorem 3 is consistent with TR-63 Proposition 16 which states that the second dart of a connecting dart sequence $CDS_i(d)$ is equal to $\varphi_0(d)$ if K_i is a contraction kernel and $\sigma_0(d)$ if K_i is a removal kernel. Indeed, since each dart has a level greater or equal to 1, we have $l_{i,d}(1) = 1$ for each dart d and level i such that $CDS_i(d) \neq (d)$. In this case using Theorem 3, we

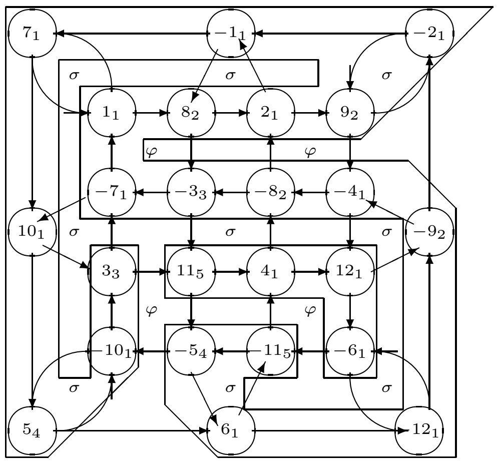


Figure 11: Successors of the surviving darts

obtain $\varphi_0(d) = d_1$ if K_i is a contraction kernel and $\sigma_0(d) = d_1$ if K_i is a removal kernel with $CDS_i(d) = d_1 \dots, d_p$.

Figure 11 presents the two connecting dart sequences defined at level 4 by the sequence of kernels described in Section 2(Figure 5). Each dart of this figure is indexed by its level. An unfolded representation of these sequence is provided by Figure 12.

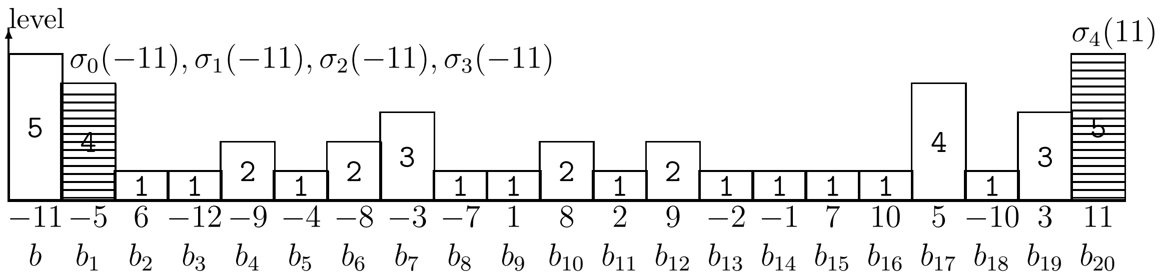
On this example, the dart -5 with a level 4 follows immediately -11 in $CDS_4(-11)$ (see also Figure 12(a)). We have thus $L_4(-11) = 4$ and $l_{4,-11}(1) = l_{4,-11}(2) = l_{4,-11}(3) = l_{4,-11}(4) = 1$. Since K_4 is a removal kernel, we have according to Theorem 3 (see also Figure 4):

$$\sigma_0(-11) = \sigma_1(-11) = \sigma_2(-11) = \sigma_3(-11) = -5$$

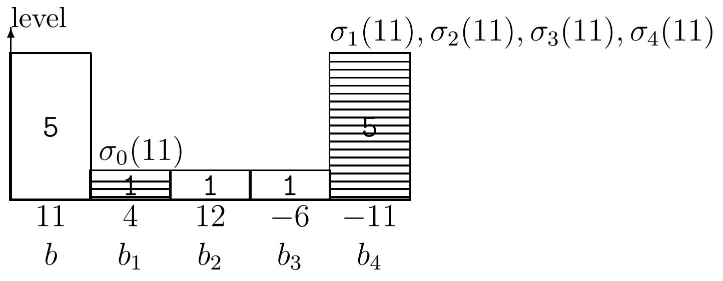
The successor of -11 at level 4 is provided by TR-63 Proposition 17. Since the last dart of $CDS_4(-11)$ is 3 which is contracted before level 4 we have:

$$\sigma_4(-11) = \varphi(3) = 11$$

In the same way, all darts of $CDS_4^*(11)$ have a level equal to 1. We have thus $L_4(11) = 1$ and $\sigma_0(11)$ is equal to the first dart of $CDS_4(11)$: 4. Moreover, using TR-63 Proposition 17, we have $\sigma_4(11) = \varphi(-6) = -11$. Then, using


level
$\sigma_0(-11), \sigma_1(-11), \sigma_2(-11), \sigma_3(-11)$
$\sigma_4(11)$
5 4 1 1 2 1 2 3 1 1 2 1 2 1 1 1 1 4 1 3 5
−11 −5 6 −12 −9 −4 −8 −3 −7 1 8 2 9 −2 −1 7 10 5 −10 3 11
b b_1 b_2 b_3 b_4 b_5 b_6 b_7 b_8 b_9 b_{10} b_{11} b_{12} b_{13} b_{14} b_{15} b_{16} b_{17} b_{18} b_{19} b_{20}

(a) $CDS_4(-11)\sigma_4(-11)$


level
$\sigma_1(11), \sigma_2(11), \sigma_3(11), \sigma_4(11)$
$\sigma_0(11)$
5 1 1 1 5
11 4 12 −6 −11
b b_1 b_2 b_3 b_4

(b) $CDS_4(11)\sigma_4(11)$

Figure 12: The sequence of darts $CDS_4(-11)\sigma_4(-11)$ and $CDS_4(11)\sigma_4(11)$. The level of each dart is indicated in its associated bar.

Corollary 2:

$$\sigma_1(11) = \sigma_2(11) = \sigma_3(11) = \sigma_4(11) = -11$$

4 Successors relations within a connecting dart sequence

Given a generalized pyramid construction plan defined by n kernels and a level $i \in \{1, \ldots, n\}$, the conjoint use of theorems 3 and corollary 2 allows us to compute the σ successors of all the darts in $\mathcal{SD}_i$ for all levels in $\{1, \ldots, i\}$. However, for each level $j < i$ the σ successors of the darts contained in the sets $\mathcal{SD}_j - \mathcal{SD}_i$ remain to be computed.

For example, the connecting dart sequences of -11 and 11 at level 4 allow us to compute the σ successors of 11 and -11 from level 0 to level 4 (Figures 11 and 12). However, if we wish to compute the combinatorial map defined at level 2, we have to compute the σ successors at level 2 of the darts $\mathcal{SD}_2 = \alpha^*(5, 3, 11)$ (Figure 4). The σ successors of $5, -5, 3$ and -3 remain thus to be computed. Let us study the construction of the pyramid defined by Figure 4 from level 2 to level 4. We have (TR-63 Tables 4 and 5):

$$\begin{aligned} CW_3(5) &= 5.-3 && (9)\\ CDS_3(5) &= 5.CDS_2^*(-5).-3.CDS_2^*(3) && (10) \end{aligned}$$

$$\begin{aligned} CW_4(-11) &= -11.-5.5 && (11)\\ CDS_4(-11) &= -11.CDS_3^*(11).-5.CDS_3^*(5).5.CDS_3^*(-5) && (12) \end{aligned}$$

We obtain by combining equations 10 and 12:

$$CDS_4(-11) = -11.CDS_3^*(11).\boxed{-5.CDS_2^*(-5).-3.CDS_2^*(3)}.5.CDS_3^*(-5)$$

The successors of -5 and 3 from level 0 to 2 may thus be determined (by Theorems 3 and corollary 2) from $CDS_4(-11)$ if we are able to retrieve $CDS_2^*(-5)$ and $CDS_2^*(3)$ from $CDS_4(-11)$. One first step toward this result consists to determine the dart at level i whose connecting dart sequence contains a given non surviving dart d. Such a dart is denoted d^i. We have for example, $5^4 = -3^4 = -11$ (equation 12) and $-3^3 = 5$ (equation 10). Note that, using TR-63-propositions 18 and 19, the set of connecting dart sequences forms a partition of $\boldsymbol{\mathcal{D}}$ at each level i in $\{1, \ldots, n\}$. Therefore, any dart d in $\boldsymbol{\mathcal{D}}$ may be associated to an unique dart $d^i \in \mathcal{SD}_i$ such that $d \in CDS_i(d^i)$. In the following we will determine the σ or φ successors of d within $CDS_i(d^i)$.

Proposition 7 *Given a pyramid construction plan defined by n kernels and an initial combinatorial map $G_0 = (\boldsymbol{\mathcal{D}}, \sigma_0, \alpha)$. Given a dart $d \in \boldsymbol{\mathcal{D}}$, the sequence of darts d^i such that $d \in CDS_i(d^i)$ satisfies:*

- *d belongs to its own connecting dart sequence until it is contracted or removed:*
$$\forall i \in \{0, \dots, l-1\} \quad d^i = d$$
with $l = level(d)$.

  Note that if d is not contracted nor removed until level n, we have $l = n+1$. Therefore, in this case the above equation becomes:
$$\forall i \in \{0, \dots, n\} \quad d^i = d$$

- *If d belongs to one K_l, the sequence $d^l, \dots, d^n$ satisfies:*
$$d \in CW_l(d^l) \text{ and } d.CDS^*_{l-1}(\alpha(d)) \trianglelefteq CDS^*_l(d^l)$$
Moreover,
$$\forall i \in \{l+1, \dots, n\} \left\{ \begin{array}{l} \alpha(d^{i-1}) \in CW_i(d^i) \text{ and} \\ d \in CDS^*_{i-1}(d^{i-1}) \trianglelefteq CDS^*_i(d^i) \end{array} \right.$$
where $CW_i(d^i)$ denotes the connecting walk of d^i defined at level i.

Proof:

The first property is a direct consequence of the definition of connecting dart sequences. Let us now suppose that d is contracted or removed at level l and let us consider $d^l \in \mathcal{SD}_l$ such that $d \in CW_l(d^l)$. The dart d^l is fully determined by this relation since each dart in $\mathcal{SD}_{l-1}$ belongs to exactly one connecting walk defined at level l. Moreover, by definition of connecting dart sequences :

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

with $CW_l(d^l) = d_1, \dots, d, \dots, d_p$.

We have thus $d \in CDS_l(d^l)$. Moreover, since d is contracted at level l it can't be the first dart of $CDS_l(d^l)$, therefore $d \neq d_1$ and:

$$d.CDS^*_{l-1}(\alpha(d)) \trianglelefteq CDS^*_l(d^l)$$

Let us now consider d^{l+1} such that $\alpha(d^l) \in CW_{l+1}(d^{l+1})$. Like previously, d^{l+1} is fully determined by this relation. Using the definition of connecting dart sequences:

$$CDS_{l+1}(d^{l+1}) = b_1 CDS_l^*(\alpha(b_1)), \cdots \alpha(d^l) CDS_l^*(d^l) \cdots, b_p CDS_l^*(\alpha(b_p))$$

with $CW_{l+1}(d^{l+1}) = b_1, \ldots, \alpha(d^l), \ldots, b_p$. Therefore:

$$d \in CDS_l^*(d^l) \trianglelefteq CDS_{l+1}^*(d^{l+1}) \Rightarrow d \in CDS_{l+1}^*(d^{l+1})$$

The property is thus true at level l. Let us suppose it true at a given level i and let us consider d^{i+1} such that $\alpha(d^i) \in CW_{i+1}(d^{i+1})$. We have by definition of connecting dart sequences:

$$\alpha(d^i) CDS_i^*(d^i) \trianglelefteq CDS_{i+1}(d^{i+1}) \Rightarrow CDS_i^*(d^i) \trianglelefteq CDS_{i+1}^*(d^{i+1})$$

Since by our recurrence hypothesis, $d \in CDS_i^*(d^i)$, d belongs to $CDS_{i+1}^*(d^{i+1})$. □

Remark 4 *With the same hypothesis as proposition 7, if $d \in K_l$, using the transitivity of the sub-word relation:*

$$\forall i \in \{l, \ldots, n\} \quad d.CDS_{l-1}^*(\alpha(d)) \trianglelefteq CDS_i^*(d^i)$$

Note that this last equation implies that:

$$\forall i \in \{l, \ldots, n\} \quad CDS_i^*(d^i) \neq \emptyset$$

$CDS_1(9)$:	9	-4														
$CDS_2(3)$:	3	-7	1	8	2	9	-2	-1	7	10						
$CDS_3(5)$:	5	6	-12	-9	-4	-8	-3	-7	1	8	2	9	-2	-1	7	10

Table 1: Connecting dart sequences containing the dart 9 from level 1 to level 3. The sequence $9.CDS_1^*(-9)$ is surrounded.

Table 1 shows the different connecting dart sequences containing the dart 9 from level 1 to 3 (Figure 5). The dart 9 is removed at the level $l = 2$ and $9CDS_1^*(-9) = 9. - 2. - 1.7.10$ (surrounded sequence in Table 1). We have from Table 1: $9^2 = 3$ and $9^3 = 5$. As stated by Proposition 7 and Remark 4, the sequence $9.CDS_1(-9)$ is a sub word of the sequences $CDS_2(3)$ and $CDS_3(5)$. We can note that on this example that $9.CDS_1(-9)$ is a suffix of $CDS_2(3)$, $CDS_3(5)$. This last configuration being particularly interesting in the following we define it precisely:

Definition 6 Set J_d

Given a pyramid construction plan defined by n kernels and an initial combinatorial map $G_0 = (\boldsymbol{\mathcal{D}}, \sigma_0, \alpha)$. Given a dart $d \in \boldsymbol{\mathcal{D}}$ with level l, the set J_d is the subset of levels included in $\{l, \dots, n\}$ such that $dCDS^*_{l-1}(\alpha(d))$ is a suffix of $CDS^*_i(d^i)$:

$$J_d = \{i \in \{l, \dots, n\} \quad | \quad dCDS^*_{l-1}(\alpha(d)) \preceq_{suff} CDS^*_i(d^i)\}$$

Corollary 3 *Using the same hypothesis and notations as Definition 6, the set J_d is equal to $I_{d'} \cap \{l, \dots, n\}$ (see TR-63 proposition 21) where d' is the last dart of $dCDS^*_{l-1}(\alpha(d))$. Moreover, the set J_d is either empty or is a contiguous interval of $\{l, \dots, n\}$ including l.*

Proof:

If $i \in J_d$, we have $i \geq l$ and d' is the last dart of $CDS_i(d^i)$. Consequently $i \in I_{d'}$ and $J_d \subset I_{d'} \cap \{l, \dots, n\}$.

Conversely, $dCDS^*_{l-1}(\alpha(d))$ is a sub-word of $CDS^*_i(d^i)$ for each i in $\{l, \dots, n\}$:

$$\forall i \in \{l, \dots, n\}, \exists S^1_i, S^2_i \mid CDS^*_i(d^i) = S^1_i dCDS^*_{l-1}(\alpha(d)) S^2_i$$

Therefore, d' belongs to $CDS_i(d^i)$ for each $i \in \{l, \dots, n\}$.

If $i \in I_{d'} \cap \{l, \dots, n\}$, since $d' \in dCDS^*_{l-1}(\alpha(d)) \trianglelefteq CDS^*_i(d^i)$ and each dart appears at most once in each connecting dart sequence defined at level i (TR-63-Proposition 19), d' does not belong to S^2_i. The dart d' being by hypothesis the last one of $CDS_i(d^i)$, the sequence S^2_i must thus be empty. The sequence $dCDS^*_{l-1}(\alpha(d))$ is thus a strict suffix of $CDS^*_i(d^i)$ and we have:

$$I_{d'} \cap \{l, \dots, n\} \subset J_d \implies I_{d'} \cap \{l, \dots, n\} = J_d$$

If $I_{d'} \cap \{l, \dots, n\}$ is empty J_d is empty. Otherwise, using TR-63-Proposition 21, $I_{d'} = \{level(d'), \dots, m\}$ where m is the upper bound of $I_{d'}$. Note that, since $d' \in CDS_{l-1}(\alpha(d))$ we have $level(d') < l$. Therefore, the intersection between $I_{d'}$ and $\{l, \dots, n\}$ is non-empty only if m is greater than l. We have then:

$$J_d = I_{d'} \cap \{l, \dots, n\} = \{level(d'), \dots, m\} \cap \{l, \dots, n\} = \{l, \dots, m\}$$

The set J_d is thus a non empty interval of $\{l, \dots, n\}$ including l. $\square$

Using Table 1, the dart 9 is removed at level 2 and $9CDS_1^*(-9)$ (surrounded sequence in Table 1) is a suffix of $CDS_2(3)$ and $CDS_3(5)$. Note that $9^4 = -11$ (Figure 5) and $9CDS_1^*(-9)$ is not a suffix of $CDS_4(-11)$. Consequently the set J_9 is equal to $\{2,3\}$. Since $I_{10} = \{1,2,3\}$, we obtain by using corollary 3:

$$J_9 = I_{10} \cap \{l,\dots,n\} = \{1,2,3\} \cap \{2,3,4\} = \{2,3\}$$

Note that since $CDS_1^*(-9) \subset K_1$, 9 is the last dart with a level 2 in $CDS_2(3)$ and $CDS_3(5)$.

Conversely, the sequence $8CDS_1(-8) = 8.2$ is not a suffix of $CDS_2(8^2 = 3)$ (Table 1 or Figure 5). We have in this case $J_8 = \emptyset$. In this case, the set $I_2 = \{1\}$ has an empty intersection with $\{l,\dots,n\} = \{2,3,4\}$.

Proposition 8 *Given a Pyramid construction plan defined by n kernels, a dart $d \in K_l$ and a level $i \in I_d$ we have:*

$$\forall j \in \{0,\dots,l-1\} \left\{ \begin{array}{ll} If & d \text{ is contracted} \\ & \varphi_j(d) = \left\{ \begin{array}{ll} \varphi_i(d^i) & \text{If } K_i \text{ is a contraction kernel} \\ \sigma_i(d^i) & \text{If } K_i \text{ is a removal kernel} \end{array}\right. \\ If & d \text{ is removed} \\ & \sigma_j(d) = \left\{ \begin{array}{ll} \varphi_i(d^i) & \text{If } K_i \text{ is a contraction kernel} \\ \sigma_i(d^i) & \text{If } K_i \text{ is a removal kernel} \end{array}\right. \end{array}\right.$$

Proof:

Since (Remark 4):

$$\forall i \in \{l,\dots,n\} \quad d.CDS_{l-1}^*(\alpha(d)) \trianglelefteq CDS_i^*(d^i)$$

The dart d, is the last dart of $CDS_i(d^i)$, only if $CDS_{l-1}^*(\alpha(d)) = \emptyset$. Then, using Theorem 2 on $CDS_{l-1}^*(\alpha(d))$:

$$\forall j \in \{0,\dots,l-1\} \left\{ \begin{array}{ll} \text{If} & K_{l-1} \text{ is a removal kernel} \\ & \sigma_j(\alpha(d)) = \sigma(\alpha(d)) \\ \text{If} & K_{l-1} \text{ is a contraction kernel} \\ & \varphi_j(\alpha(d)) = \varphi(\alpha(d)) \end{array}\right.$$

Since K_l and K_{l-1} have not the same type, and d is contracted or removed at level l, the above equation may be written as:

$$\forall j \in \{0,\dots,l-1\} \left\{ \begin{array}{ll} \text{If} & d \text{ is contracted} \\ & \varphi_j(d) = \varphi(d) \\ \text{If} & d \text{ is removed} \\ & \sigma_j(d) = \sigma(d) \end{array}\right.$$

Then, using TR63-proposition 17 [7]:

$$\forall j \in \{0,\dots,l-1\} \left\{ \begin{array}{ll} If & d \text{ is contracted} \\ & \varphi_j(d) = \varphi(d) = \left\{ \begin{array}{ll} \varphi_i(d^i) & \text{If } K_i \text{ is a contraction kernel} \\ \sigma_i(d^i) & \text{If } K_i \text{ is a removal kernel} \end{array} \right. \\ If & d \text{ is removed} \\ & \sigma_j(d) = \sigma(d) = \left\{ \begin{array}{ll} \varphi_i(d^i) & \text{If } K_i \text{ is a contraction kernel} \\ \sigma_i(d^i) & \text{If } K_i \text{ is a removal kernel} \end{array} \right. \end{array} \right.$$

□

Note that, this last proposition allows us to retrieve all the successors of the last dart of a connecting dart sequence. Using the example displayed in Figure 11, the dart 3, contracted at level 3 is the last dart of $CDS_3(-11)$. Using Proposition 8 we have:

$$\varphi_0(3) = \varphi_1(3) = \varphi_2(3) = \varphi_3(-11) = 11$$

Proposition 9 *Given a dart $d \in K_l$, the series $(d^j)_{j \in J_d}$, satisfy the following properties:*

1. *The set J_d is nonempty iff d is the last dart of $CW_l(d^l)$.*
2. *If J_d is nonempty, then for each j in J_d such that $j+1 \in J_d$, $\alpha(d^j)$ is the last dart of $CW_{j+1}(d^{j+1})$*

Proof:

1. **Equivalence between J_d nonempty and d last dart of $CW_l(d^l)$:**

   Since $d \in CW_l(d^l)$(Proposition 7) it exists two sequences of darts S_1 and S_2 such that:

   $$CDS_l(d^l) = S_1.d.CDS^*_{l-1}(\alpha(d))S_2$$

   If J_d is nonempty, it contains at least l (Corollary 3). Therefore S_2 must be empty by definition of J_d. If d is not the last dart of $CW_l(d^l)$, it exists at least one dart in S_2 which is refused by hypothesis.

   Conversely, if d is the last dart of $CW_l(d^l)$, S_2 is empty by definition of connecting dart sequences. Therefore, $l \in J_d$ which is not empty.

2. J_d nonempty implies that if $\{j, j+1\} \subset J_d$, $\alpha(d^j)$ is the last dart of $CW_{j+1}(d^{j+1})$

   Given a level j in J_d such that $j+1 \in J_d$, we have:

   $$CDS_{j+1}(d^{j+1}) = d_1 CDS_j^*(\alpha(d_1)) \cdots \alpha(d^j) CDS_j^*(d^j) \cdots d_p CDS_j^*(\alpha(d_p))$$

   with $\alpha(d^j) \in CW_{j+1}(d^{j+1}) = d_1, \ldots, \alpha(d^j) \ldots, d_p$(Proposition 7).
   Since $j \in J_d$, the sequence $dCDS_{l-1}^*(\alpha(d))$ is a suffix of $CDS_j^*(d^j)$. Therefore, $j+1$ belongs to J_d only if $CDS_j^*(d^j)$ is a suffix of $CDS_{j+1}(d^{j+1})$. This last relation holds only if $\alpha(d^j)$ is the last dart of $CW_{j+1}(d^{j+1})$.

□

Considering the dart 9 in Figure 5, we have $9^2 = 3$, $9^3 = 5$ and:

$$\left\{ \begin{array}{lcl} CW_2(3) & = & 3.8.9 \\ CW_3(5) & = & 5.-3 \end{array} \right.$$

The dart 9 is thus the last dart of $CW_2(9^2)$ while $\alpha(9^2) = -3$ is the last dart of $CW_3(9^3)$.

Note that we have according to the definition of connecting walks: $\sigma_2(3) = \sigma_1(9) = 5$ and $\varphi_3(5) = \varphi_2(-3) = \sigma_2(3) = \sigma_1(9) = 5$. The following corollary generalizes this observation:

Corollary 4 *With the same hypothesis as Proposition 9, given a dart $d \in K_l$ with $l < n$:*

$$\forall i \in J_d \left\{ \begin{array}{ll} If & d \text{ is contracted} \\ & \varphi_{l-1}(d) = \left\{ \begin{array}{ll} \varphi_i(d^i) & \text{If } K_i \text{ is a contraction kernel} \\ \sigma_i(d^i) & \text{If } K_i \text{ is a removal kernel} \end{array} \right. \\ If & d \text{ is removed} \\ & \sigma_{l-1}(d) = \left\{ \begin{array}{ll} \varphi_i(d^i) & \text{If } K_i \text{ is a contraction kernel} \\ \sigma_i(d^i) & \text{If } K_i \text{ is a removal kernel} \end{array} \right. \end{array} \right.$$

Proof:

Since d is the last dart of $CW_l(d^l)$ (Proposition 9), we have by definition of a connecting walk (TR-54, Definition 21):

- If K_l is a contraction kernel $\varphi_l(d^l) = \varphi_{l-1}(d)$

- If K_l is a removal kernel $\sigma_l(d^l) = \sigma_{l-1}(d)$

Since d is either contracted or removed according to K_l, the proposition is true at rank l. Let us suppose it true for a given i in J_d such that $i+1 \in J_d$. Since $\alpha(d^i)$ is the last dart of $CW_{i+1}(d^{i+1})$:

- If K_{i+1} is a contraction kernel, K_i is a removal kernel and:
$$\varphi_{i+1}(d^{i+1}) = \varphi_i(\alpha(d^i)) = \sigma_i(d^i)$$
- If K_{i+1} is a removal kernel, K_i is a contraction kernel and:
$$\sigma_{i+1}(d^{i+1}) = \sigma_i(\alpha(d^i)) = \varphi_i(d^i)$$

Let us now decompose the demonstration according to the type of K_l.

- If K_l is a contraction kernel, d is contracted and:
  - If K_{i+1} is a contraction kernel, K_i is a removal kernel. Moreover according to our recursive hypothesis:
  $$\varphi_{i+1}(d^{i+1}) = \sigma_i(d^i) = \varphi_{l-1}(d)$$
  - If K_{i+1} is a removal kernel:
  $$\sigma_{i+1}(d^{i+1}) = \varphi_i(d^i) = \varphi_{l-1}(d)$$
- If K_l is a removal kernel, d is removed at level l and:
  - If K_{i+1} is a contraction kernel:
  $$\sigma_{i+1}(d^{i+1}) = \varphi_i(d^i) = \sigma_{l-1}(d)$$
  - If K_{i+1} is a contraction kernel:
  $$\varphi_{i+1}(d^{i+1}) = \sigma_i(d^i) = \sigma_{l-1}(d)$$

□

Using the example displayed in Figure 5, the dart 9 is removed at level 2 and $J_9 = \{2,3\}$ (see page 36). Moreover, we have $9^2 = 3$ and $9^3 = 5$. Using TR-63, Proposition 17, $\sigma_1(9) = \varphi_1(-9)$ is equal to the φ_0 successor of the last dart of $CDS_1(-9)$ which is equal to 5(Figure 5a)). Since K_2 is a removal kernel and K_3 a contraction one, we obtain by Corollary 4 (see also Figure 4):

$$\sigma_1(9) = \sigma_2(3) = \varphi_3(5) = 5$$

Note that $\sigma_2(3)$ and $\varphi_3(5)$ may also be obtained from $CDS_2(3)$ and $CDS_3(5)$ by using TR-63, Proposition 17. The major interest of Corollary 4 is that it allows us to retrieve the σ_{l-1} or φ_{l-1} successor of one dart d such that J_d is non empty by using the connecting dart sequences defined at the current level. In the above example, the value of $\sigma_1(9)$ may be retrieve by using the connecting dart sequences defined at level 2 or 3.

The following corollary extends this result to the successors of a dart d from level $L_{l-1}(\alpha(d))$ to level $l = level(d)$:

Corollary 5 *Given a dart d in K_l, with $l < n$, if $CDS^*_{l-1}(\alpha(d)) \neq \varepsilon$:*

$$\left.\begin{array}{lcl} \forall & i & \in & J_d \\ \forall & j & \in & \{L_{l-1}(\alpha(d)), \dots, l-1\} \end{array}\right\}$$
$$\left\{\begin{array}{l} If\ d\ is\ contracted \\ \varphi_j(d) = \left\{\begin{array}{ll} \varphi_i(d^i) & If\ K_i\ is\ a\ contraction\ kernel \\ \sigma_i(d^i) & If\ K_i\ is\ a\ removal\ kernel \end{array}\right. \\ If\ d\ is\ removed \\ \sigma_j(d) = \left\{\begin{array}{ll} \varphi_i(d^i) & If\ K_i\ is\ a\ contraction\ kernel \\ \sigma_i(d^i) & If\ K_i\ is\ a\ removal\ kernel \end{array}\right. \end{array}\right. \quad (13)$$

Proof:

If J_d is empty the demonstration is trivial. Otherwise, using Corollary 2 on $CDS_{l-1}(\alpha(d))$:

$$\forall j \in \{L_{l-1}(\alpha(d)), \dots, l-1\} \left\{\begin{array}{ll} \text{If } K_{l-1} & \text{is a removal kernel:} \\ & \sigma_j(\alpha(d)) = \sigma_{l-1}(\alpha(d)) \\ \text{If } K_{l-1} & \text{is a contraction kernel:} \\ & \varphi_j(\alpha(d)) = \varphi_{l-1}(\alpha(d)) \end{array}\right.$$

Since K_{l-1} and K_l have not the same type and $d \in K_l$:

$$\forall j \in \{L_{l-1}(\alpha(d)), \dots, l-1\} \left\{\begin{array}{ll} \text{If } d & \text{is contracted} \\ & \varphi_j(d) = \varphi_{l-1}(d) \\ \text{If } d & \text{is removed} \\ & \sigma_j(d) = \sigma_{l-1}(d) \end{array}\right.$$

Using the above equation and Corollary 4, for each (i, j) in $J_d \times \{L_{l-1}(\alpha(d)), \dots, l-1\}$:

$$\begin{array}{ll} If & d \text{ is contracted} \\ & \varphi_j(d) = \varphi_{l-1}(d) = \left\{ \begin{array}{ll} \varphi_i(d^i) & \text{If } K_i \text{ is a contraction kernel} \\ \sigma_i(d^i) & \text{If } K_i \text{ is a removal kernel} \end{array} \right. \\ If & d \text{ is removed} \\ & \sigma_j(d) = \sigma_{l-1}(d) = \left\{ \begin{array}{ll} \varphi_i(d^i) & \text{If } K_i \text{ is a contraction kernel} \\ \sigma_i(d^i) & \text{If } K_i \text{ is a removal kernel} \end{array} \right. \end{array}$$

□

Corollary 5 may thus be considered as an extension of Corollary 4. Indeed, under an additional hypothesis ($CDS^*_{l-1}(\alpha(d)) \neq \varepsilon$), this corollary allows to retrieve additional φ or σ successors of one dart d such that $dCDS^*_{l-1}(\alpha(d))$ is a suffix of some $CDS_i(d^i)$ with $i \geq l$.

In the following we will study more carefully the sequence of darts encountered in a connecting dart sequence after a given dart. However, we have first to define in which case such a sequence exists.

Definition 7 Set for successors

Given a pyramid construction plan defined by n kernels and a dart $d \in \boldsymbol{\mathcal{D}}$, the set $\mathcal{DS}_d$ is the set of levels in $\{1, \dots, n\}$ such that d is neither the last nor the only dart of $CDS_i(d^i)$:

$$\mathcal{DS}_d = \{i \in \{1, \dots, n\} \mid \exists S_1, S_2,\ CDS_i(d^i) = S_1 d S_2 \textit{ with } S_2 \neq \varepsilon\}$$

Remark 5 *Note that given a dart d contracted or removed at level $l < n$, we have:*

$$\mathcal{DS}_d = \{i \in \{1, \dots, l-1\} \mid CDS^*_i(d) \neq \varepsilon\} \cup (\{l, \dots, n\} - I_d)$$

Since I_d is a contiguous interval of $\{l, \dots, n\}$ including l:

$$\mathcal{DS}_d \cap \{l, \dots, n\} = \{m, \dots, n\}$$

where $m-1$ is equal to $l-1$ if I_d is empty and the upper bound of I_d otherwise.

Proposition 10 *Given a pyramid construction plan defined by n kernels, a dart $d \in \boldsymbol{\mathcal{D}}$ and a level $i < l \in \mathcal{DS}_d$. The sets $\mathcal{DS}_d$ and $\mathcal{DS}_{\alpha(d)}$ include the following sets:*

$$\begin{array}{ll} \{j \in \{i, \dots, l-1\} \mid (j-i) \textit{ mod } 2 = 0\} & \subset \mathcal{DS}_d \\ \{j \in \{i, \dots, l-1\} \mid (j-i) \textit{ mod } 2 = 1\} & \subset \mathcal{DS}_{\alpha(d)} \end{array}$$

with $l = level(d)$

Proof:

Since $i < l$, $i \in \mathcal{DS}_d$ iff $CDS_i(d) \neq (d)$(Remark 5). Using Proposition 3:

$$\forall j \in \{i, \dots, l-1\} \left\{ \begin{array}{ll} CDS_i(d) \preceq_{pre} CDS_j(d) & \text{if } (j-i) \ mod\ 2 = 0 \\ CDS_i^*(d) \preceq_{pre} CDS_j^*(\alpha(d)) & \text{if } (j-i) \ mod\ 2 = 1 \end{array} \right.$$

Therefore, if $(j-i)\ mod\ 2 = 0$, $CDS_i^*(d)$ is not empty and is a prefix of $CDS_j^*(d)$ which is thus non-empty. Therefore $j \in \mathcal{DS}_d$. If $(j-i)\ mod\ 2 = 1$, since $CDS_i^*(d)$ is non empty $CDS_j^*(\alpha(d))$ is non empty and $j \in \mathcal{DS}_{\alpha(d)}$. □

Remark 6 *Proposition 10 states only necessary conditions. Therefore, using the same hypothesis and notations, we may have $j \in \mathcal{DS}_d$ with $i < level(d)$ while $(j-i)\ mod\ 2 = 1$.*

Our analysis of the sequence of darts which follows a given dart d in $CDS_i(d^i)$ is mainly based on the level of each dart following d in $CDS_i(d^i)$. The study of this sequence of darts requires to define the range of levels encountered after d in $CDS_i(d^i)$ and the index of the first dart with a level greater than a given value encountered after d in $CDS_i(d^i)$.

Definition 8 Index, Max-level and successor functions

Given a pyramid construction plan defined by n kernels:

- *The index of a dart d at level i: $Ind_i(d)$ is equal to its index in the connecting dart sequence $CDS_i(d^i)$ which contains it at level i.*
- *For each dart $d \in \boldsymbol{\mathcal{D}}$, for each $i \in \mathcal{DS}_d$, $\mathcal{L}_i(d)$ is defined as the minimum between $level(d)$ and the greater level of the darts which follow d in $CDS_i^*(d^i)$:*

$$\mathcal{L}_i(d) = \min(level(d), \max_{k \in \{Ind_i(d)+1,\dots,p\}} \{level(d_k)\}) \qquad (14)$$

with $CDS_i(d^i) = d_1, \dots, d_p$.

- *The $j-successor$ of a dart d at level $i \in \mathcal{DS}_d$: $succ_{i,d}(j)$ with $j \in \{1, \dots, \mathcal{L}_i(d)\}$ is the index of the first dart which follows d in $CDS_i(d^i)$ with a level greater than j.*

$$\left. \begin{array}{ll} \forall d \in & \boldsymbol{\mathcal{D}} \\ \forall i \in & \mathcal{DS}_d \\ \forall j \in & \{1, \dots, \mathcal{L}_i(d)\} \end{array} \right\}$$

$$succ_{i,d}(j) = \min\{k \in \{Ind_i(d)+1, \dots, p\} \mid level(d_k) \geq j\} \qquad (15)$$

with $CDS_i(d^i) = d_1, \dots, d_p$.

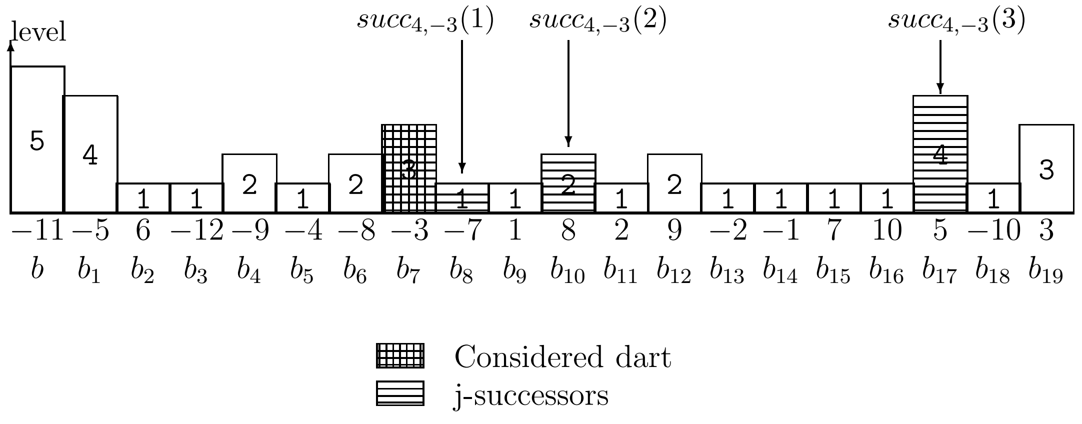


Figure 13: The connecting dart sequence $CDS_4(-11)$ with the j-successors of the dart -3 at levels $1, 2$ and 3. The level of each dart is indicated in its associated bar.

Using example displayed in Figure 13, the dart -3 with level 3 belongs to $CDS_4(-11)$. The index of -3 in $CDS_4(-11)$ is equal to 7. The maximal level encountered after -3 is the one of the dart 5 and is equal to 4. We have thus: $\mathcal{L}_4(-3) = min(3, 4) = 3$. The first darts encountered after -3 with a level greater than $1, 2$ and 3 are respectively $-7, 8$ and 5. The indexes of these darts are respectively equal to: $8, 10$ and 17. We finally obtain (Figure 13):

$$Ind_4(-3) = 7,\ \mathcal{L}_4(-3) = 3 \text{ and}$$
$$succ_{4,-3}(1) = 8,\ succ_{4,-3}(2) = 10,\ succ_{4,-3}(3) = 17$$

Note that the function $succ_{i,d}$ is not bijective. For example using the dart 5 in Figure 13 we obtain:

$$Ind_4(5) = 17,\ \mathcal{L}_4(5) = 3 \text{ and}$$
$$succ_{4,5}(1) = 18,\ succ_{4,5}(2) = 19,\ succ_{4,5}(3) = 19$$

The function $\mathcal{L}_i$ may be considered as an extension of the MaxLevel function L_i (Definition 4). Indeed, both functions provide the maximum level which may be encountered in a connecting dart sequence after a given dart (Proposition 11). In the same way, the function $succ_{i,d}$ may be considered as an extension of the function $l_{i,d}$ (Definition 5). The following proposition formalize the relations between functions L_i and $\mathcal{L}_i$ and extends some properties of the function L_i(Proposition 4) to $\mathcal{L}_i$.

Proposition 11 *Given a pyramid construction plan defined by n kernels, and a dart d in $\boldsymbol{\mathcal{D}}$ with a level l:*

1. *The functions $L_i(d)$ and $\mathcal{L}_i(d)$ are equal for each i in $\{1,\dots,l-1\}\cap\mathcal{DS}_d$.*

$$\forall i\in\{1,\dots,l-1\}\cap\mathcal{DS}_d \quad \mathcal{L}_i(d)=L_i(d)$$

2. *If $l<n$:*

$$\begin{array}{lll}\forall i\in\{1,\dots,l-1\}\cap\mathcal{DS}_{\alpha(d)} & \mathcal{L}_i(\alpha(d)) & \le \mathcal{L}_{i+1}(d)\\ \forall i\in\{l,\dots,n-1\}\cap\mathcal{DS}_d & \mathcal{L}_i(d) & \le \mathcal{L}_{i+1}(d)\end{array}$$

3. *If $l<n$ for each level i in $\{l,\dots,n-1\}\cap\mathcal{DS}_d$, each dart indexed by $succ_{i,d}(k)$ in $CDS_i(d^i)$ with $k\in\{1,\dots,\mathcal{L}_i(d)\}$, is indexed by $succ_{j,d}(k)$ in $CDS_j(d^j)$ for each j greater than i.*

$$\left.\begin{array}{l}\forall i\in\{l,\dots,n-1\}\cap\mathcal{DS}_d\\ \forall j\in\{i,\dots,n\}\\ \forall k\in\{1,\dots,\mathcal{L}_i(d)\}\end{array}\right\} d_{succ_{i,d}(k)}=b_{succ_{j,d}(k)}$$

$$\textit{with:}\ \left\{\begin{array}{lll}CDS_i(d^i) & = & d_1,\dots,d_p\\ CDS_j(d^{i+1}) & = & b_1,\dots,b_q\end{array}\right.$$

Proof:

1. If $i\le l-1$, we have $d^i=d$ and $Ind_i(d)=0$, with $CDS_i(d)=d=d_0,d_1,\dots,d_p$. Therefore, if $p>0$:

$$\max_{k\in\{Ind_i(d)+1,\dots,p\}}\{level(d_k)\}=\max_{k\in\{1,\dots,p\}}\{level(d_k)\}=L_i(d)$$

   Moreover, in this case $L_i(d)\le i<l$ (see proposition 4), thus $\mathcal{L}_i(d)=L_i(d)$.

2. First note that according to Proposition 10, if $i<l-1$ belongs to $\mathcal{DS}_{\alpha(d)}$ then $i+1\in\mathcal{DS}_d$. Therefore, if $\mathcal{L}_i(\alpha(d))$ with $i<l-1$ is defined $\mathcal{L}_{i+1}(d)$ is also defined.

   In the same way, since $\{l,\dots,n-1\}\cap\mathcal{DS}_d$ may be written as $\{m,\dots,n-1\}$ (Remark 5), if $\mathcal{L}_i(d)$ is defined, $i\in\{m,\dots,n-1\}$. Therefore, $i+1\in\{m+1,\dots,n\}\subset\mathcal{DS}_d$ and $\mathcal{L}_{i+1}(d)$ is also defined.

Using Proposition 4:

$$\forall i \in \{1, \ldots, l-2\} \cap \mathcal{DS}_{\alpha(d)} \quad L_i(\alpha(d)) \leq L_{i+1}(d)$$

Moreover, if $l-1 \in \mathcal{DS}_{\alpha(d)}$, we must have $CDS^*_{l-1}(\alpha(d)) \neq \varepsilon$. Using Proposition 7:

$$d \in dCDS^*_{l-1}(\alpha(d)) \trianglelefteq CDS^*_l(d^l) \Rightarrow L_{l-1}(\alpha(d)) \leq \mathcal{L}_{\ l}(d)$$

Since $\mathcal{L}_{\ i}(d) = L_i(d)$ for each i in $\{1, \ldots, l-1\}$, we obtain:

$$\forall i \in \{1, \ldots, l-1\} \cap \mathcal{DS}_{\alpha(d)} \quad \mathcal{L}_{\ i}(\alpha(d)) \leq \mathcal{L}_{\ i+1}(d)$$

Let us now suppose that $l < n$, using Proposition 7:

$$\forall i \in \{m, \ldots, n-1\} \quad d \in CDS^*_i(d^i) \trianglelefteq CDS^*_{i+1}(d^{i+1})$$

where m is the lower bound of $\{l, \ldots, n-1\} \cap \mathcal{DS}_d$.

Therefore, given the sequences of dart S_1, S_2, S'_1, and S'_2 such that:

$$\begin{array}{lcl} CDS^*_{i+1}(d^{i+1}) & = & S_1 CDS^*_i(d^i) S_2 \\ CDS^*_i(d^i) & = & S'_1 d S'_2 \end{array}$$

$$\mathcal{L}_{\ i}(d) = \min(l, \max_{d' \in S'_2}\{level(d')\}) \leq \min(l, \max_{d' \in S'_2 S_2}\{level(d')\}) = \mathcal{L}_{\ i+1}(d)$$

Note that, since $i \in \mathcal{DS}_d$, $S'_2 \neq \emptyset$. We obtain thus:

$$\begin{array}{ll} \forall i \in \{1, \ldots, l-1\} & \mathcal{L}_{\ i}(\alpha(d)) \leq \mathcal{L}_{\ i+1}(d) \\ \forall i \in \{m, \ldots, n-1\} & \mathcal{L}_{\ i}(d) \leq \mathcal{L}_{\ i+1}(d) \end{array}$$

3. Using Proposition 7 and the transitivity of the sub word relation we have for each $i \in \mathcal{DS}_d \cap \{l, \ldots, n\}$ and for each $j \geq i$:

$$CDS^*_i(d^i) \trianglelefteq CDS^*_j(d^j)$$

Since $\mathcal{L}_{\ i}(d)$ is defined, at level i, the function $succ_{i,d}$ is also defined (Remark 6). Given a level j greater than i and an index k in $\{1, \ldots, \mathcal{L}_{\ i}(d)\}$ we can consider the sequences of darts S_1, S_2, S'_1 and S'_2 such that:

$$\begin{array}{lcl} CDS^*_j(d^j) & = & S_1 CDS^*_i(d^i) S_2 \\ CDS^*_i(d^i) & = & S'_1 d.d' \ldots d_{succ_{i,d}(k)} S'_2 \end{array}$$

We obtain:

$$\begin{array}{rcl} CDS_j^*(d^j) & = & S_1 S_1' d.d' \dots d_{succ_{i,d}(k)} S_2' S_2 \\ & = & b_1, \dots, b_q \end{array}$$

With $level(d_{succ_{i,d}(k)}) \geq k$ and:

$$\forall b \in \{d', \dots, d_{succ_{i,d}(k)-1}\} \quad level(b) < k$$

Therefore, $d_{succ_{i,d}(k)}$ is the first dart with a level greater than k encountered when traversing $CDS_j^*(d^j)$ starting from d. Thus $b_{succ_{j,d}(k)} = d_{succ_{i,d}(k)}$ and:

$$\forall j \in \{i, \dots, n\} \; \forall k \in \{1, \dots, \mathcal{L}_i(d)\} \quad d_{succ_{i,d}(k)} = b_{succ_{j,d}(k)}$$

$\square$

The third point of the above proposition only shows that given a level k and a dart d, if $succ_{i,d}(k)$ exists for some level i, $d_{succ_{i,d}(k)}$ will remain the same in all connecting dart sequences defined at level $j \geq i$ although its index may vary.

Proposition 11 establishes some relations between functions L_i and $\mathcal{L}_i$. In the same way, Proposition 12 formalizes some relations between the functions $l_{i,d}$(Definition 5) and $succ_{i,d}$.

Proposition 12 *Given a pyramid construction plan defined by n kernels, and a dart d in $\mathcal{D}$ the function $succ_{i,d}$ satisfies the following properties:*

$$\left.\begin{array}{l} \forall i \in \{1, \dots, l-1\} \cap \mathcal{DS}_d \\ \forall j \in \{1, \dots, L_i(d)\} \end{array}\right\} \quad succ_{i,d}(j) = l_{i,d}(j)$$

If $CDS_{l-1}^*(\alpha(d)) \neq \varepsilon$:

$$\left.\begin{array}{l} \forall i \in \{l, \dots, n\}, \\ \forall j \in \{1, \dots, L_{l-1}(\alpha(d))\} \end{array}\right\} \quad d_{succ_{i,d}(j)} = d_{l_{l-1,\alpha(d)}(j)}$$

with $l = level(d)$. Note that if d does not belong to any kernel, $l = n+1$ and thus the set $\{l, \dots, n\}$ is empty.

Proof:

If $i < l$, since $d \in \mathcal{DS}_d$, $CDS_i(d) \neq \varepsilon$ and the functions $l_{i,d}$ and $succ_{i,d}$ are both defined. Moreover, since d is not yet contracted nor removed at level i, $Ind_i(d) = 1$ and $\mathcal{L}\ _i(d) = L_i(d)$ (Proposition 11). Thus the research of the first dart with a level greater than $j \in \{1, \ldots, L_i(d)\}$ is performed on $CDS^*_i(d)$. This last calculus corresponds to the computation of $l_{i,d}(j)$:

$$\forall i \in \{1, \ldots, l-1\},\ \forall j \in \{1, \ldots, L_i(d)\} \quad succ_{i,d}(j) = l_{i,d}(j)$$

Let us now suppose that d is contracted or removed at level l. Since the sequence $CDS^*_{l-1}(\alpha(d))$ is not empty, $L_{l-1}(\alpha(d))$ is defined. We can thus consider an index j in $\{1, \ldots, L_{l-1}(\alpha(d))\}$. By definition of the function $l_{l-1,\alpha(d)}$, $d_{l_{l-1,\alpha(d)}(j)} \neq \alpha(d)$. Therefore, using Remark 4:

$$\forall i \in \{l, \ldots, n\} \quad d \ldots d_{l_{l-1,\alpha(d)}(j)} \trianglelefteq d.CDS^*_{l-1}(\alpha(d)) \trianglelefteq CDS^*_i(d^i)$$

Therefore, for each i in $\{l, \ldots, n\}$, $d_{l_{l-1,\alpha(d)}(j)}$ is the first dart with a level greater than j encountered when traversing $CDS_i(d^i)$ starting from d. Therefore:

$$\left.\begin{array}{l}\forall i \in \{l, \ldots, n\},\\ \forall j \in \{1, \ldots, L_{l-1}(\alpha(d))\}\end{array}\right\} \quad d_{succ_{i,d}(j)} = d_{l_{l-1,\alpha(d)}(j)}$$

□

Given one dart d such that J_d is non empty, and one level $i \in J_d$, $dCDS^*_{l-1}(\alpha(d))$ is a suffix of $CDS_i(d^i)$(Definition 6). This last property should have an influence on the value of $\mathcal{L}\ _i(d)$. The following proposition formalizes the relations between J_d and $\mathcal{L}\ _i(d)$.

Proposition 13 *Given a pyramid construction plan defined by n kernels, a dart d in K_l and a level $i \in \mathcal{DS}_d \cap \{l, \ldots, n\}$. If $CDS^*_{l-1}(\alpha(d)) \neq \emptyset$ the following propositions are equivalent:*

1. $i \in J_d$
2. $\mathcal{L}\ _i(d) = L_{l-1}(\alpha(d))$
3. $\mathcal{L}\ _i(d) < l$

Proof:

(1)⇒(2): If $i \in J_d$, it exists a sequence S of darts such that: $CDS_i(d^i) = S.d.CDS^*_{l-1}(\alpha(d))$. Thus:

$$\mathcal{L}\ _i(d) = Max_{d' \in CDS^*_{l-1}(\alpha(d))}\{level(d')\} = L_{l-1}(\alpha(d))$$

(2)⇒(3): Since $L_{l-1}(\alpha(d)) \leq l-1$ (Proposition 4) the implication is trivial.

(3)⇒(1): If J_d is empty, d is not the last dart of $CW_l(d^l)$ (Proposition 9). Thus it exists one dart d' such that $CW_l(d^l) = \dots d.d' \dots$ with $level(d') = l$. Moreover, by definition of connecting dart sequences:

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

Therefore $\mathcal{L}_{\ l}(d) = l \leq \mathcal{L}_{\ i}(d)$(Proposition 11). This last result being refused by hypothesis we have $J_d \neq \emptyset$. Let us denote by m the upper bound of J_d. The set J_d being an interval we have $J_d = \{l, \dots, m\}$. If m is strictly lower than i it exists one dart d'' such that:

$$CDS_{m+1}(d^{m+1}) = \cdots , dCDS^*_{l-1}(\alpha(d))d'' \cdots$$

with $level(d'') = m$.

Then: $\mathcal{L}_{\ m+1}(d) = l \leq \mathcal{L}_{\ i}(d)$ which is again refused by hypothesis. Therefore we must have $i \leq m$ and thus $i \in J_d$.

□

Remark 7 *Note that using proposition 13, $L_{l-1}(\alpha(d))$ may be replaced by $\mathcal{L}_{\ i}(d)$ in Corollary 5.*

The two following theorems use the results already obtained to retrieve the successors of a dart d from the darts which follow it in a connecting dart sequence. In order to not overload the demonstration this result has been split into two theorems according to the value of $CDS^*_{l-1}(\alpha(d))$ with $l = level(d)$.

Theorem 4 *Given a pyramid construction plan defined by n kernels, and a dart d with a level $l < n$ such that $CDS^*_{l-1}(\alpha(d)) = \varepsilon$. For each $i \in \{l, \dots, n\} \cap \mathcal{DS}_d$, the φ or σ successors of d defined at level $j < \mathcal{L}_{\ i}(d)$, may be retrieved by the following property:*

$$\left.\begin{array}{l} \forall i \in \{l, \dots, n\} \cap \mathcal{DS}_d, \\ \forall j \in \{1, \dots, \mathcal{L}_{\ i}(d)\} \end{array}\right\} \left\{\begin{array}{lcll} \varphi_{j-1}(d) & = & d_{succ_{i,d}(j)} & \textit{If } d \textit{ is contracted} \\ \sigma_{j-1}(d) & = & d_{succ_{i,d}(j)} & \textit{If } d \textit{ is removed} \end{array}\right.$$

Proof:

Since $CDS^*_{l-1}(\alpha(d)) = \varepsilon$, Theorem 2 applied on $CDS^*(\alpha(d))$ provides the following result:

$$\forall k \in \{0,\dots,l-1\} \left\{ \begin{array}{ll} \text{If} & K_{l-1} \text{ is a removal kernel} \\ & \sigma_k(\alpha(d)) = \sigma(\alpha(d)) \\ \text{If} & K_{l-1} \text{ is a contraction kernel} \\ & \varphi_k(\alpha(d)) = \varphi(\alpha(d)) \end{array} \right.$$

Since K_{l-1} and K_l have not the same type and $d \in K_l$ we obtain, with $j = k+1$:

$$\forall j \in \{1,\dots,l\} \left\{ \begin{array}{ll} \text{If} & d \text{ is contracted} \\ & \varphi_{j-1}(d) = \varphi(d) \\ \text{If} & d \text{ is removed} \\ & \sigma_{j-1}(d) = \sigma(d) \end{array} \right.$$

Note that, this last equation is equivalent to:

$$\forall j \in \{1,\dots,l\} \left\{ \begin{array}{ll} \text{If} & d \text{ is contracted} \\ & \varphi_{j-1}(d) = \varphi_{l-1}(d) = \varphi(d) \\ \text{If} & d \text{ is removed} \\ & \sigma_{j-1}(d) = \sigma_{l-1}(d) = \sigma(d) \end{array} \right. \tag{16}$$

Since $CDS^*_{l-1}(\alpha(d)) = \varepsilon$, $dCDS^*_{l-1}(\alpha(d)) = d$ and $J_d = I_d$. Therefore (Remark 5):

$$\{l,\dots,n\} \cap \mathcal{DS}_d = \{l,\dots,n\} - J_d$$

Let us now decompose the demonstration according to the value of J_d.

If J_d is empty, $\{l,\dots,n\} \cap \mathcal{DS}_d = \{l,\dots,n\}$.

In this case d is not the last dart of $CW_l(d^l)$ (Proposition 9). Therefore, if k denotes the index of d in $CW_l(d^l)$, using the definition of connecting walks, b_{k+1} satisfies:

$$b_{k+1} = \left\{ \begin{array}{ll} \varphi_{l-1}(d) & \text{If d is contracted} \\ \sigma_{l-1}(d) & \text{If d is removed} \end{array} \right.$$

Moreover, $dCDS^*_{l-1}(\alpha(d))b_{k+1} = db_{k+1}$ is a sub word of $CDS_l(d^l)$. Therefore, since $level(b_{k+1}) = l$, we have:

$$\left\{ \begin{array}{ll} \forall i \in \{l,\dots,n\} & \mathcal{L}_{\ i}(d) = l \\ \forall j \in \{1,\dots,l\} & d_{succ_{l,d}(j)} = b_{k+1} \end{array} \right.$$

Using the above equations and equation 16:

$$\forall j \in \{1,\dots,l\} \left\{ \begin{array}{ll} \text{If} & d \text{ is contracted} \\ & \varphi_{j-1}(d) = \varphi_{l-1}(d) = b_{k+1} = d_{succ_{l,d}(j)} \\ \text{If} & d \text{ is removed} \\ & \sigma_{j-1}(d) = \sigma_{l-1}(d) = b_{k+1} = d_{succ_{l,d}(j)} \end{array} \right.$$

Therefore, using Proposition 11:

$$\begin{array}{l} \forall i \in \{l,\dots,n\} \\ \forall j \in \{1,\dots,l = \mathcal{L}_i(d)\} \end{array} \left\{ \begin{array}{ll} \text{If} & d \text{ is contracted} \\ & \varphi_{j-1}(d) = d_{succ_{l,d}(j)} = d_{succ_{i,d}(j)} \\ \text{If} & d \text{ is removed} \\ & \sigma_{j-1}(d) = d_{succ_{l,d}(j)} = d_{succ_{i,d}(j)} \end{array} \right.$$

Which corresponds to the expected result.

If J_d is non empty, for each j in $J_d = I_d$, d is the last dart of $CDS_i(d^i)$.

In this case, $\mathcal{L}_i(d)$ is undefined for each i in J_d. Let us denote by m the upper bound of $I_d = J_d$. If $m = n$ we have nothing to demonstrate since $\mathcal{L}_i(d)$ is undefined for all levels in $\{l,\dots,n\}$. In this case the successors of d are provided by Proposition 8.

If $m < n$, let us denote by k the index of $\alpha(d^m)$ in $CW_{m+1}(d^{m+1})$. We have:

$$CDS_{m+1}(d^{m+1}) = b_1 CDS^*_m(\alpha(b_1)) \cdots \alpha(d_m) \dots d b_{k+1} \cdots b_p CDS^*_{m+1}(\alpha(b_p))$$

with $CW_{m+1}(d^{m+1}) = b_1,\dots,\alpha(d^m) b_{k+1} \dots b_p$.

We have thus, by definition of connecting walks:

$$\begin{array}{rcl} b_{k+1} & = & \left\{ \begin{array}{ll} \sigma_m(\alpha(d^m)) & \text{If } K_{m+1} \text{ is a removal kernel} \\ \varphi_m(\alpha(d^m)) & \text{If } K_{m+1} \text{ is a contraction kernel} \end{array} \right. \\ \\ & = & \left\{ \begin{array}{ll} \varphi_m(d^m) & \text{If } K_m \text{ is a contraction kernel} \\ \sigma_m(d^m) & \text{If } K_m \text{ is a removal kernel} \end{array} \right. \end{array}$$

Using TR-63 proposition 17:

- If d is contracted:

$$\varphi(d) = \left\{ \begin{array}{ll} \varphi_m(d^m) & \text{If } K_m \text{ is a contraction kernel} \\ \sigma_m(d^m) & \text{If } K_m \text{ is a removal kernel} \end{array} \right.$$

- If d is removed:

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

Therefore:

$$b_{k+1} = \begin{cases} \varphi(d) & \text{If } d \text{ is contracted} \\ \sigma(d) & \text{If } d \text{ is removed} \end{cases}$$

Since $level(b_{k+1}) = m > l$:

$$\begin{array}{lll} \forall i \in \{m+1, \dots, n\} & \mathcal{L}_{\ i}(d) & = & l \\ \forall j \in \{1, \dots, l\} & d_{succ_{l,d}(j)} & = & b_{k+1} \end{array}$$

Using equation 16:

$$\forall j \in \{1, \dots, l\} \begin{cases} \varphi_{j-1}(d) = \varphi(d) = b_{k+1} = d_{succ_{l,d}(j)} & \text{If } d \text{ is contracted} \\ \sigma_{j-1}(d) = \sigma(d) = b_{k+1} = d_{succ_{l,d}(j)} & \text{If } d \text{ is removed} \end{cases}$$

Using Proposition 11:

$$\left.\begin{array}{l} \forall i \in \{m+1, \dots, n\} \\ \forall j \in \{1, \dots, l = \mathcal{L}_{\ i}(d)\} \end{array}\right\} \begin{cases} \text{If} & d \text{ is contracted} \\ & \varphi_{j-1}(d) = d_{succ_{l,d}(j)} = d_{succ_{i,d}(j)} \\ \text{If} & d \text{ is removed} \\ & \sigma_{j-1}(d) = d_{succ_{l,d}(j)} = d_{succ_{i,d}(j)} \end{cases}$$

□

This last theorem may be illustrated by Figure 14 (see also Figure 5). We have indeed, $l = level(-8) = 2$ with $CDS_2^*(\alpha(-8)) = CDS_2^*(8) = \varepsilon$. Moreover, $CDS_1^*(-8) \neq \varepsilon$ and -8 is the last dart of the connecting dart sequence which contains it only at level 2. We have thus:

$$\begin{cases} J_{-8} & = & I_{-8} = \{2\} \text{ and} \\ \mathcal{DS}_{-8} & = & \{1\} \cup \{3, 4\} \end{cases}$$

Using the notations of Theorem 4 we are thus in the case where J_d is not empty. The upper bound m of J_{-8} is in this case equal to 2 with $-8^2 = -5$ and $-8^3 = 5$. We have additionally:

$$CW_3(5) = 5. - 3 = \alpha(d^m).b_{k+1}$$

We have thus, $b_{k+1} = -3$ and since -8 is removed:

$$\sigma_0(-8) = d_{succ_{4,-8}(1)} = \sigma_1(-8) = d_{succ_{4,-8}(2)} = b_{k+1} = -3$$

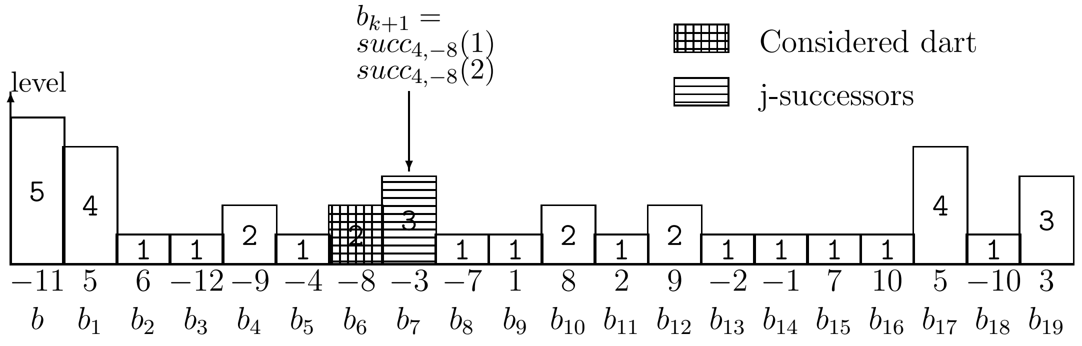


Figure 14: The connecting dart sequence $CDS_4(-11)$ with the j-successors of the dart -8 at levels 1 and 2. The level of each dart is indicated in its associated bar.

Theorem 5 *Given a pyramid construction plan defined by n kernels, and a dart d with level $l < n$ such that $CDS^*_{l-1}(\alpha(d)) \neq \varepsilon$. For each $i \in \{l, \dots, n\}$, the φ or σ successors of d defined at level $j < \mathcal{L}_i(d)$, may be retrieved by the following property:*

$$\left.\begin{array}{l} \forall i \in \{l, \dots, n\}, \\ \forall j \in \{1, \dots, \mathcal{L}_i(d)\} \end{array}\right\} \left\{\begin{array}{lcll} \varphi_{j-1}(d) & = & d_{succ_{i,d}(j)} & \text{If } d \text{ is contracted} \\ \sigma_{j-1}(d) & = & d_{succ_{i,d}(j)} & \text{If } d \text{ is removed} \end{array}\right.$$

Proof:

First note that since $CDS^*_{l-1}(\alpha(d)) \neq \varepsilon$, and $dCDS^*_{l-1}(\alpha(d)) \trianglelefteq CDS_i(d^i)$ for any i in $\{l, \dots, n\}$(Remark 4), the set I_d is empty. We have thus (Remark 5):

$$\{l, \dots, n\} \cap \mathcal{DS}_d = \{l, \dots, n\}$$

Moreover, since $CDS^*_{l-1}(\alpha(d)) \neq \varepsilon$, $L_{l-1}(\alpha(d))$ is defined and Theorem 3 may be applied on $CDS_{l-1}(\alpha(d))$:

$$\forall j \in \{1, \dots, L_{l-1}(\alpha(d))\} \left\{\begin{array}{ll} \sigma_{j-1}(\alpha(d)) = d_{l_{l-1,\alpha(d)}(j)} & \text{If } K_{l-1} \text{ is a removal kernel} \\ \varphi_{j-1}(\alpha(d)) = d_{l_{l-1,\alpha(d)}(j)} & \text{If } K_{l-1} \text{ is a contraction kernel} \end{array}\right.$$

Since d belongs to K_l which has not the same type than K_{l-1}:

$$\forall j \in \{1, \dots, L_{l-1}(\alpha(d))\} \left\{\begin{array}{ll} \varphi_{j-1}(d) = d_{l_{l-1,\alpha(d)}(j)} & \text{If } d \text{ is contracted} \\ \sigma_{j-1}(d) = d_{l_{l-1,\alpha(d)}(j)} & \text{If } d \text{ is removed} \end{array}\right.$$

Using Proposition 12, $d_{l_{l-1,\alpha(d)}(j)} = d_{succ_{i,d}(j)}$ for each (i,j) in $\{l,\dots,n\} \times \{1,\dots,L_{l-1}(\alpha(d))\}$. Therefore:

$$\begin{array}{rl}\forall i \in & \{l,\dots,n\} \\ \forall j \in & \{1,\dots,L_{l-1}(\alpha(d))\}\end{array}\Bigg\}\left\{\begin{array}{ll}\text{If } d & \text{is contracted:} \\ & \varphi_{j-1}(d) = d_{succ_{i,d}(j)} \\ \text{If } d & \text{is removed:} \\ & \sigma_{j-1}(d) = d_{succ_{i,d}(j)}\end{array}\right. \tag{17}$$

Let us now decompose the demonstration according to the value of J_d.

If J_d is empty, d is not the last dart of $CW_l(d^l) = b_1 \dots, b_p$ (Proposition 9).

Let us denote by k the index of d in $CW_l(d^l)$. The connecting dart sequence of d^l at level l is then equal to:

$$CDS_l(d^l) = b_1 CDS^*_{l-1}(\alpha(b_1) \dots dCDS^*_{l-1}(\alpha(d))b_{k+1} \dots b_p CDS^*_{l-1}(\alpha(b_p))$$

The sequence $dCDS^*_{l-1}(\alpha(d))b_{k+1}$ is thus a sub-word of $CDS_l(d^l)$. Moreover, by definition of connecting walks:

$$\begin{array}{rcll}\varphi_{l-1}(d) & = & b_{k+1} & \text{If } K_l \text{ is a contraction kernel} \\ \sigma_{l-1}(d) & = & b_{k+1} & \text{If } K_l \text{ is a removal kernel}\end{array}$$

Since $level(b_{k+1}) = l$, and $L_{l-1}(\alpha(d)) \le l-1$, b_{k+1} is the first dart with a level strictly greater than $L_{l-1}(\alpha(d))$ encountered when traversing $CDS^*_l(d^l)$ from d. Therefore:

$$\forall j \in \{L_{l-1}(\alpha(d))+1,\dots,l\} \quad b_{k+1} = d_{succ_{l,d}(j)}$$

Using Corollary 2 and Remark 3, with $CDS_{l-1}(\alpha(d))$:

$$\forall j \in \{L_{l-1}(\alpha(d))+1,\dots,l\} \quad \left\{\begin{array}{ll}\text{If } K_{l-1} & \text{is a removal kernel:} \\ & \sigma_{j-1}(\alpha(d)) = \sigma_{l-1}(\alpha(d)) \\ \text{If } K_{l-1} & \text{is a contraction kernel:} \\ & \varphi_{j-1}(\alpha(d)) = \varphi_{l-1}(\alpha(d))\end{array}\right.$$

Since K_{l-1} and K_l have not the same type, and $d \in K_l$:

$$\forall j \in \{L_{l-1}(\alpha(d))+1,\dots,l\} \quad \left\{\begin{array}{l}\text{If } d \text{ is contracted:} \\ \varphi_{j-1}(d) = \varphi_{l-1}(d) = b_{k+1} = d_{succ_{l,d}(j)} \\ \text{If } d \text{ is removed:} \\ \sigma_{j-1}(d) = \sigma_{l-1}(d) = b_{k+1} = d_{succ_{l,d}(j)}\end{array}\right.$$

Using Proposition 11:

$$\left.\begin{array}{lcl}\forall i & \in & \{l,\dots,n\} \\ \forall j & \in & \{L_{l-1}(\alpha(d))+1,\dots,l\}\end{array}\right\} \left\{\begin{array}{l}\text{If } d \text{ is contracted:} \\ \varphi_{j-1}(d) = d_{succ_{l,d}(j)} = d_{succ_{i,d}(j)} \\ \text{If } d \text{ is removed:} \\ \sigma_{j-1}(d) = d_{succ_{l,d}(j)} = d_{succ_{i,d}(j)}\end{array}\right. \tag{18}$$

Where $succ_{l,d}(j)$ and $succ_{i,d}(j)$ denote the index of the same dart respectively in $CDS_l(d^l)$ and $CDS_i(d^i)$.

The combination of equations 17 and 18 provides the expected result. Note that in this case, since $level(b_{k+1}) = l$ we have $\mathcal{L}_{\ l}(d) = l$ and thus $\mathcal{L}_{\ i}(d) = l$ for each i in $\{l,\dots,n\}$.

If J_d is non empty, let us denote by m the greater level contained in J_d. For each i in J_d, $dCDS^*_{l-1}(\alpha(d))$ is a suffix of $CDS^*_i(d^i)$. Therefore, $\mathcal{L}_{\ i}(d) = L_{l-1}(\alpha(d))$ for all i in $\{l,\dots,m\}$. If $m = n$, equation 17 provides the expected result.

Otherwise, let us suppose that $m < n$ and let us denote by k the index of $\alpha(d^m)$ in $CW_{m+1}(d^{m+1}) = b_1,\dots,b_q$. Since $dCDS^*_{l-1}(\alpha(d))$ is a suffix of $CDS^*_m(d^m)$, $CDS_{m+1}(d^{m+1})$ is equal to:

$$\begin{array}{lll} b_1CDS^*_m(\alpha(b_1))\cdots & \alpha(d^m)CDS^*_m(d^m)b_{k+1} & \cdots b_qCDS^*_m(\alpha(b_q)) \\ b_1CDS^*_m(\alpha(b_1)))\cdots & \alpha(d^m)SdCDS^*_{l-1}(\alpha(d))b_{k+1} & \cdots b_qCDS^*_m(\alpha(b_q))\end{array}$$

with S such that $CDS^*_m(d^m) = SdCDS^*_{l-1}(\alpha(d))$.

Therefore, $dCDS^*_{l-1}(\alpha(d))b_{k+1}$ is a sub word of $CDS_{m+1}(d^{m+1})$. Moreover, since $level(b_{k+1}) = m \geq l$, $\mathcal{L}_{\ m+1}(d) = l$ and:

$$\forall j \in \{L_{l-1}(\alpha(d))+1,\dots,l\} \quad b_{k+1} = d_{succ_{m+1,d}(j)}$$

Using Corollary 5 with $i = m \in J_d$, we have for each j in $\{L_{l-1}(\alpha(d))+1,\dots,l\}$:

If d is contracted:

$$\varphi_{j-1}(d) = \left\{\begin{array}{ll}\varphi_m(d^m) & \text{If } K_m \text{ is a contraction kernel} \\ \sigma_m(d^m) & \text{If } K_m \text{ is a removal kernel}\end{array}\right.$$

If d is removed:

$$\sigma_{j-1}(d) = \left\{\begin{array}{ll}\varphi_m(d^m) & \text{If } K_m \text{ is a contraction kernel} \\ \sigma_m(d^m) & \text{If } K_m \text{ is a removal kernel}\end{array}\right.$$

Moreover, since b_{k+1} follows $\alpha(d^m)$ in $CW_{m+1}(d^{m+1})$ and K_m, K_{m+1} have not the same type:

$$b_{k+1} = \left\{ \begin{array}{lcll} \sigma_m(\alpha(d^m)) & = & \varphi_m(d^m) & \text{If } K_m \text{ is a contraction kernel} \\ \varphi_m(\alpha(d^m)) & = & \sigma_m(d^m) & \text{If } K_m \text{ is a removal kernel} \end{array} \right.$$

Therefore:

$$\forall j \in \{L_{l-1}(\alpha(d))+1, \ldots, l\} \quad \left\{ \begin{array}{ll} \text{If } d & \text{is contracted:} \\ & \varphi_{j-1}(d) = b_{k+1} = d_{succ_{m+1,d}(j)} \\ \text{If } d & \text{is removed:} \\ & \sigma_{j-1}(d) = b_{k+1} = d_{succ_{m+1,d}(j)} \end{array} \right.$$

Using Proposition 11:

$$\left. \begin{array}{lcl} \forall i & \in & \{m+1, \ldots, n\} \\ \forall j & \in & \{L_{l-1}(\alpha(d))+1, \ldots, l\} \end{array} \right\} \left\{ \begin{array}{l} \text{If } d \text{ is contracted:} \\ \varphi_{j-1}(d) = d_{succ_{m+1,d}(j)} = d_{succ_{i,d}(j)} \\ \text{If } d \text{ is removed:} \\ \sigma_{j-1}(d) = d_{succ_{m+1,d}(j)} = d_{succ_{i,d}(j)} \end{array} \right. \tag{19}$$

Moreover, since $\mathcal{L}_{m+1}(d) = l$, $\mathcal{L}_i(d) = l$ for each i in $\{m+1, \ldots, n\}$. We have thus:

$$\begin{array}{ll} \forall i \in \{l, \ldots, m\} & \mathcal{L}_i(d) = L_{l-1}(\alpha(d)) \\ \forall i \in \{m+1, \ldots, n\} & \mathcal{L}_i(d) = l \end{array}$$

Therefore, for each i in $\{l, \ldots, m\}$ equation 17 provides the expected result. If $i \in \{m+1, \ldots, n\}$, the expected result is provided by the conjoint use of equation 17 and 19

□

Let us consider the dart -5 in Figure 15 removed at level 4 (Figure 4). The first darts with a level greater than $1, 2, 3$ and 4 which follow -5 in $CDS_4(-11)$ are respectively equal to $6, -9, -3$ and 5. The indices of these darts are equal to $3, 5, 8$ and 18. Since the greater level encountered after 5 in $CDS_4(-11)$ is equal to 4 we have $\mathcal{L}_4(-5) = 4$. We thus obtain:

$Ind_4(-5) = 2$, $\mathcal{L}_4(-5) = 4$ and
$succ_{4,-5}(1) = 3$, $succ_{4,-5}(2) = 5$, $succ_{4,-5}(3) = 8$, $succ_{4,-5}(4) = 18$

The sequence $CDS_3^*(5)$ being non empty (Figure 5), Theorem 5 applies and we obtain:

$$\begin{array}{ll} \sigma_0(-5) = d_{succ_{4,-5}(1)} = 6, & \sigma_1(-5) = d_{succ_{4,-5}(2)} = -9, \\ \sigma_2(-5) = d_{succ_{4,-5}(3)} = -3, & \sigma_3(-5) = d_{succ_{4,-5}(4)} = 5 \end{array}$$

More precisely, using the notations of Theorem 5, the greatest level encountered in $CDS_3(5)$ is the one of the dart -3 equal to 3 (Figure 9). We have thus $L_3(\alpha(-5)) = L_3(5) = 3$ and since -5 is removed, equation 17 provides the σ successors of the dart -5 from levels 0 to 2. Moreover:

$$-5^4 = -11 \text{ and } CW_4(-11) = -11. - 5.5$$

The dart -5 is thus not the last dart of $CW_4(-5^4)$ and the set J_{-5} is empty (Proposition 9). We are thus in the first case of theorem 5 with $b_{k+1} = 5$. Since $level(-5) = 4$, the set $\{L_{l-1}(\alpha(d)) + 1, \dots, l\} = \{L_3(5) + 1, level(-5)\}$ is reduced to $\{4\}$ and equation 18 only provides the σ successor of -5 at level 3.

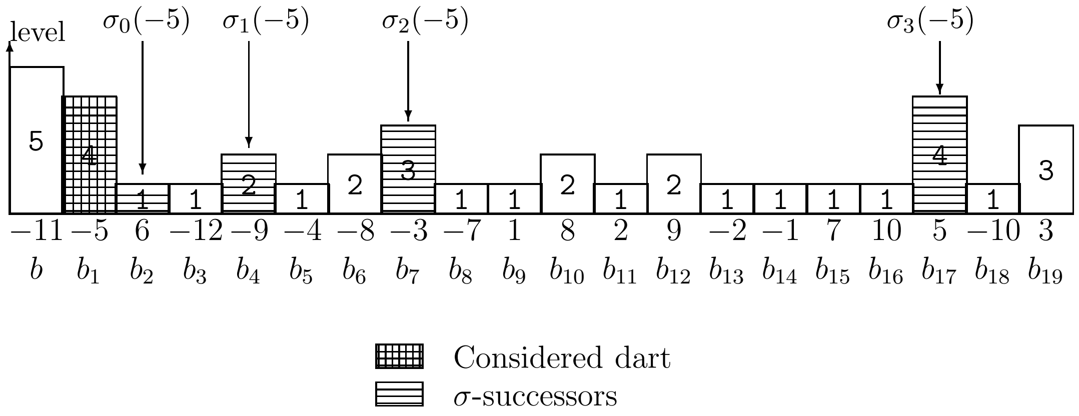


Figure 15: The connecting dart sequence $CDS_4(-11)$ with the σ-successors of the dart -5 from level 1 to level 3. The level of each dart is indicated in its associated bar.

5 Sequential computation of the reduced combinatorial maps

If a dart d is not contracted nor removed at a given level i, Theorems 2 and 3 allow us to retrieve all its φ or σ successors until level i by traversing $CDS_i(d)$. In the same way, if d has been contracted or removed before, level i, Theorems 4 and 5 allow us to retrieve its φ or σ successors until level $level(d)$ by traversing $CDS_i(d^i)$. This traversal may be performed by algorithm *survive* (see TR-63 section 3.2). Indeed, algorithm 2 uses the same instructions as algorithm *survive* to traverse the connecting dart sequence (Algorithm 2, lines 16 to 22). However, using theorems 2 to 5 the successors of a given dart have a greater index than the current dart in the connecting dart sequence which contains it. Therefore, a trivial adaptation of the algorithm *survive* to compute all the successors of the darts within a connecting dart sequence $CDS_i(d)$ would require a call to $survive(i, d')$ for each dart $d' \in CDS_i(d)$. Since the number of steps required by the algorithm *survive* to traverse $CDS_i(d)$ is equal to its length $|CDS_i(d)|$, the complexity of such a trivial algorithm is about $|CDS_i(d)|^2$.

Given a level i we avoid the time overhead induced by the trivial algorithm by storing an array *prec* of size $i + 1$. At each step of Algorithm 2, this array contains the previous darts within the connecting dart sequence which have a level greater or equal to $j \in \{1, \ldots, i+1\}$ (see Table 3 and the sequence above the table). Therefore given a dart d whose level is greater or equal to i, if $d' \in CDS_i(d)$ denotes the current dart traversed by the algorithm $pyramid\_up\_to(i, d, prec)$, all the darts between $prec[j]$ and d' within $CDS_i(d)$ have a level strictly less than j. Therefore for each j in $\{1, \ldots, level(d')\}$, d' is the first dart with a level greater than j encountered when traversing $CDS_i(d)$ form $prec[j]$. In other words:

$$\left.\begin{array}{lll} \forall & d' \in & CDS_i(d) \\ \forall & j \in & \{1, \ldots, level(d')\} \end{array}\right\} d_{succ_{i,prec[j]}(j)} = d' \tag{20}$$

Algorithm 1 is based on the above property. Indeed, Algorithm 1 with parameters $i \in \{1, \ldots, n\}$ and $d \in \mathcal{SD}_i$ initializes the array *prec* to d, and then call Algorithm 2. Since the recursive calls of this last algorithm are the same as the ones of algorithm *survive* (Algorithm 2, lines 16-22 and TR-63 section 3.2), Algorithm 2 traverses the sequence $CDS_i^*(d)\varphi_i(d)$, if K_i is a contraction kernel, and $CDS_i^*(d)\sigma_i(d)$ if K_i is a removal kernel. Moreover, since the set of connecting dart sequences at level i defines a partition of $\boldsymbol{\mathcal{D}}$, Algorithm 1 will consider twice the surviving darts in $\mathcal{SD}_i$ and only once the non surviving darts in $\boldsymbol{\mathcal{D}} - \mathcal{SD}_i$. Its complexity is thus equal to $\mathcal{O}(|\boldsymbol{\mathcal{D}}| + |\mathcal{SD}_i|)$.

At each step of the algorithm we consider a new dart d and use equation 20 and Theorems 2 to 5 to initialize the σ_j or φ_j successors of the dart prec[j] to d (lines 3 to 11). The array prec is then updated up to $level(d)$ (lines 13-14).

```
void pyramid_up_to(int i, dart d)
{
    int prec[1..i+1];

    for(int j=1;j ≤ i+1;j++)
        prec[j]=d;
    if ( state(i)==Contracted)
        pyramid_up_to_rec(i,$\varphi(d)$,prec);
    else
        pyramid_up_to_rec(i,$\sigma(d)$,prec);
}
```

Algorithm 1: *This algorithm initializes the array prec and run pyramid_up_to_rec(See Algorithm 2) which will traverse* $CDS_i^*(d)$

A complete proof of the validity of this algorithm is provided in Section 8.1.

Tables 2 and 3 illustrate the behavior of algorithms 1 and 2. These tables are relative to the pyramid defined on Figure 4 (see also Figure 5). At level 4 the initial 3×3 grid is reduced to a single loop defined by the darts 11 and -11.

The first two columns of Table 2 describe $CDS_4(-11).\sigma_4(-11)$ and the level of each dart of this sequence. Columns 3 to 7 of each line represents the assignments to the function Σ performed by Algorithms 1 and 2. For example, since -11 is the first dart of the connecting dart sequence and K_4 is a removal kernel, the following darts of the sequence which are equal to some $d_{l_{i,d}(j)}$ for $j \in \{1,\dots,5\}$ are equal to $\sigma_{j-1}(-11)$ (Theorem 3 and section 8.1.2). Note that the darts $d_{l_{i,d}(j)}$ may be easily read on Table 2 as the darts belonging to the first line with a level greater than j (Table 3). In the same way, since the dart -5 has been removed at level 4, its σ_{j-1}-successors are equal to the first darts with a level greater than j for each $j \in \{1,\dots,4\}$ (see Theorem 4, 5 and section 8.1.3).

Table 3 presents the different values taken by the array prec[] along the calls to Algorithm 2. The first column of this table displays the value of the dart passed to Algorithm 2, the level of this dart is indicated in the second column. For example, when Algorithm 2 is called with -5 which is the second dart of $CDS_4(-11)$, the array prec has been initialized to -11 by Algorithm 1(for loop). This state of the array prec[] is displayed on the

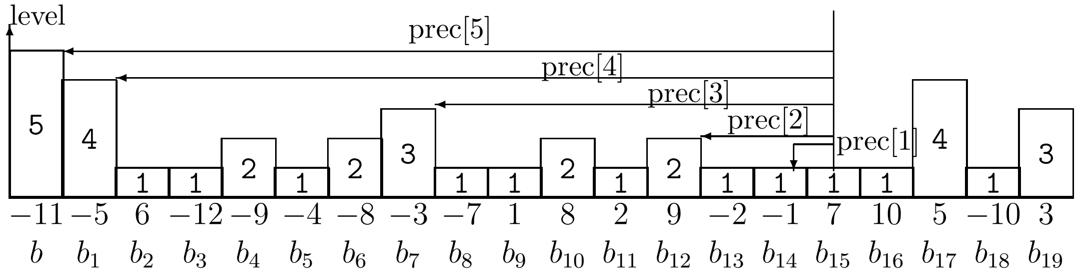
```
void pyramid_up_to_rec(int i, dart d, dart prec[i])
{
    for(int j=1;$j \leq \min(level(d), i+1)$;j++)
    {
        if(state(level(prec[j]))==Contracted or
            (level(prec[j]) $>$ i and
            state(i) == Contracted))
                $\Sigma(j-1, \alpha(prec[j])) = d$; //$\varphi_{j-1}(prec[j]) = d$
            else
                $\Sigma(j-1, prec[j]) = d$; //$\sigma_{j-1}(prec[j]) = d$
    }

    for(int j =1;$j \leq \min(level(d), i+1)$;j++)
        prec[j]=d;

    if(level(d)>i)
        return;

    if(state(level(d))==Contracted)
        return pyramid_up_to_rec(i,$\varphi(d)$,prec)

    return pyramid_up_to_rec(i,$\sigma(d)$,prec)
}
```

Algorithm 2: *This algorithm builds all the combinatorial maps up to level i*

first line of Table 3. Algorithm 2 called with -5 fills the array prec[] by -5 until $level(-5) = 4$(lines 13-14). This new state is displayed on the second line of Table 3. Note that the value of $prec[5]$ remains initialized to -11 until the end of the traversal of $CDS_4(-11)\sigma_4(-11)$. Indeed since $L_4(-11) \leq 4$(Proposition 4), $\sigma_4(-11) = 11$ is the first dart with a level greater or equal to 5 encountered in $CDS_4^*(-11)\sigma_4(-11)$.

	level	G_4	G_3	G_2		G_1		G_0	
d		$\sigma_4(d)$	$\sigma_3(d)$	$\sigma_2(d)$	$\varphi_2(d)$	$\sigma_1(d)$	$\varphi_1(d)$	$\sigma_0(d)$	$\varphi_0(d)$
-11	5	11	-5	-5		-5		-5	
-5	4		5	-3		-9		6	
6	1								-12
-12	1								-9
-9	2					-8		-4	
-4	1								-8
-8	2					-3		-3	
-3	3				5		8		-7
-7	1								1
1	1								8
8	2					9		2	
2	1								9
9	2					5		-2	
-2	1								-1
-1	1								7
7	1								10
10	1								5
5	4		11	3		3		-10	
-10	1								3
3	3				11		11		11
11	5								

Table 2: The connecting dart sequence $CDS_4(-11)$ with the level of each dart and the assignments performed by Algorithm 2



The successive values of the array *prec*						
darts	level	prec[5]	prec[4]	prec[3]	prec[2]	prec[1]
-5	4	-11	-11	-11	-11	-11
6	1	-11	-5	-5	-5	-5
-12	1	-11	-5	-5	-5	6
-9	2	-11	-5	-5	-5	-12
-4	1	-11	-5	-5	-9	-9
-8	2	-11	-5	-5	-9	-4
-3	3	-11	-5	-5	-8	-8
-7	1	-11	-5	-3	-3	-3
1	1	-11	-5	-3	-3	-7
8	2	-11	-5	-3	-3	1
2	1	-11	-5	-3	8	8
9	2	-11	-5	-3	8	2
-2	1	-11	-5	-3	9	9
-1	1	-11	-5	-3	9	-2
7	1	-11	-5	-3	9	-1
10	1	-11	-5	-3	9	7
5	4	-11	-5	-3	9	10
-10	1	-11	5	5	5	5
3	3	-11	5	5	5	-10
11	5	-11	5	3	3	3

Table 3: The different values taken by the array prec during a call to Algorithm 2 on $CDS_4(-11)$. The histogram above the table illustrates the value of the array prec for the dart 7.

6 Parallel Computation of the reduced combinatorial maps

The aim of this section is to study a parallel implementation of Algorithms 1 and 2. One trivial parallel algorithm would consist to allocate one processor for each dart and to design the parallel algorithm in such a way that each processor associated to a dart d traverses the sequence $CDS_i(d^i)$ from the dart d to the first dart of the sequence with a level greater or equal to $level(d)$ or i. However, the complexity of such a parallel algorithm would be determined by the length of the longest connecting dart sequence defined at level i. Therefore, if $i = n$ and only two surviving darts remains at level n the time required by the parallel algorithm should be close to the one of its sequential counter part which has to traverse only two connecting dart sequences.

The main improvements of Algorithms 3 and 4 besides the basic parallel algorithm explained above is to allow each processor to use the results already obtained by the other processors. Indeed, using the example displayed in Figure 12 (see also Figure 11), once the processor associated to -11 has updated $(\sigma_j)_{j\in\{0,1,2,3\}}(-11)$ to -5, the determination of $\sigma_4(-11)$ is equivalent to the determination of the first dart in $CDS_4(-11)\sigma_4(-11)$ with a level greater than 4. If $\sigma_3(-5)$ has been set to 5 by the processor associated to -5, we can insure by Theorems 4 and 5 that all the darts of $CDS_4(-11)\sigma_4(-11)$ between $Ind_4(-5)$ and $Ind_4(5)$ have a lower level than $level(\sigma_3(5)) = 4$. The index of $\sigma_4(-11)$ in $CDS_4(-11)$ is thus greater than the one of $\sigma_3(-5)$ and the processor associated to -11 can continue the traversal of $CDS_4(-11)$ from $\sigma_3(-5) = 5$ without traversing the sequence from -5 to 5. In the same way if the processor associated to -5 has already reached the dart 11 we can set $\sigma_4(-11)$ to 11 without any additional traversal. However, since the dart -5 is removed at level 4, the value of $\sigma_4(-5) = \Sigma(4, -5)$ is not defined. In order to allow each processor to store the value of the darts which have a greater level than the ones previously encountered we define a function $First(.,.)$ from $\{0, \dots, i\} \times \boldsymbol{\mathcal{D}}$ to $\boldsymbol{\mathcal{D}}$. Let us denote by $S_i(d)$ the sequence $CDS_i(d)\sigma_i(d)$ if d is removed and $CDS_i(d)\varphi_i(d)$ if d is contracted and let us consider one dart $d \in \boldsymbol{\mathcal{D}}$ and one level $j \in \{0, \dots, i\}$. The value of $First(j, d)$ is defined by:

- The first dart in $S_i(d^i)$ whose index in $S_i(d^i)$ is strictly greater than the one of d and whose level is greater or equal to j *if d is removed.*

- The first dart in $S_i(\alpha(d)^i)$ whose index in $S_i(\alpha(d)^i)$ is strictly greater than the one of d and whose level is greater or equal to j *if d is contracted.*

Note that, $First(j,d)$ is defined for all $j \in \{0,\dots,i\}$ since the last dart of $S_i(d^i)$ and $S_i(\alpha(d)^i)$ have a level greater or equal to $i+1$.

```
1   void pyramid_up_to_par($\mathcal{GLP}$ G,int i )
2   {
3       for each $d \in \mathcal{D}$ do in parallel
4               $next[d] = \sigma(d)$
5       for all $d \in \mathcal{D}$ do in parallel
6               get_successors(d,i);
7       for all $d \in \mathcal{D}$ do in parallel
8           for j= 0 to level(d) -1
9               $\Sigma(j,d) = First(j,d)$
10
11  }
```

Algorithm 3: *Initialization of common variables (lines 3-4), calls to Algorithm 4 (lines 5-6) and mapping of the results in the array $\Sigma(.,.)$ (lines 7-9)*

The current dart considered by the processor associated to a dart is stored in a shared array *next*[] indexed by $\mathcal{D}$. If $next_k[d]$ denotes the value of the array *next* at iteration k, this array is defined by $next_0[d] = \sigma[d]$ (Algorithm 3 line 4) and the following recursive construction scheme (see Algorithm 4, lines 3, 17 and 24):

- If $next_k[d]$ is removed before level i or if it survives up to level $i+1$ and $state(i) = Removed$:

$$next_{k+1}[d] = next_k[next_k[d]]$$

- If $next_k[d]$ is contracted before level i or if it survives up to level $i+1$ and $state(i) = Contracted$:

$$next_{k+1}[d] = next_k[\alpha(next_k[d])]$$

```
void get_successors(dart $d$,int i)
{
    dart d'=next[$d$]
    int max_level=MIN(level(d'),i+1);

    // Initialisation of the array First
    for(int j=1;j$\leq$ max_level;j++)
        $First(j-1,d)=d'$

    min_level[$d$]=max_level+1;


    while(level(d')$\leq$ i)
    {
        d'' = (state(level(d')) == Contracted?$\alpha(d')$:d')
        if(min_level[d''] $>$ min_level[$d$])
        {
            // d' has determined some interesting darts
            // => copy them into First(.,$d$)

            for(int j=min_level[d];j< min_level[d''];j++)
                $First(j-1,d)=First(j-1,d'')$
            min_level[$d$]=min_level[d''];
        }
        d'=next[$d$]=next[d''];
    }

}
```

Algorithm 4: *Parallel algorithm associated to a dart d(first parameter). This algorithm fills the array $First(.,d)$ whose values up to level(d) are equal to $\Sigma(.,d)$(Algorithm 3, lines 7-9)*

Tables 4 to 9 present the successive values of the array $next$ and $First$ defined by Algorithms 3 and 4. The pyramid used in this example is displayed in Figure 4. Both algorithms compute the whole pyramid and are thus run with $i = 4$. The successive values of the array $next$ are displayed in the second column, each dart being indexed by its level.

For example, $next_0[-11]$ is initialized to $\sigma_0(-11) = -5$ (First row of Figure 4 and Table 4 line 2). Since the dart -5 is removed at level 4, we have $next_1[-11] = next_0[next_0[-11]] = next_0[-5] = 6$ (Table 4 lines 2 and 8 and Table 5 line 2). In the same way, $next_0[-12]$ is initialized to $\sigma_0(-12) = -6$(Table 4 line 1). Since -6 is contracted at level 1, we have (Table 4 lines 1 and 18 and Table 5, line 1):

$$next_1[-12] = next_0[\alpha(next_0[-12])] = next_0[\alpha(-6)] = next_0[6] = -11$$

The complexity of Algorithm 4, is determined by the number of iterations performed in the while loop (line 13-26) and thus by the length of the sequence $next_k[d]$. We have shown (proposition 14, section 8.2) that the sequence $next_k[d]$ is included in a given connecting dart sequence at level i (say $CDS_i(d'))$ and that the distance between $next_k[d]$ and $next_{k+1}[d]$ within $CDS_i(d')$ growth exponentially as a function of k. The complexity of Algorithm 4 is thus equal to $\mathcal{O}(\log_2(|CDS_i(d^{max})|))$ where $|CDS_i(d^{max})|$ is the length of the longest connecting dart sequence defined at level i. In the worst case where level i is composed of only two connecting dart sequences, one being reduced to one dart: $|CDS_i(d^{max})| = |\boldsymbol{\mathcal{D}}| - 1$ and the complexity of the parallel algorithm is equal to $\mathcal{O}(\log_2(|\boldsymbol{\mathcal{D}}|))$. However, if all connecting dart sequences defined at level i have about the same size $|CDS_i(d^{max})| \approx \frac{|\boldsymbol{\mathcal{D}}|}{|\mathcal{SD}_i|}$ the complexity is equal to $\mathcal{O}(\log_2(|\boldsymbol{\mathcal{D}}|) - \log_2(|\mathcal{SD}_i|))$. A complete proof of the validity of algorithms 3 and 4 is provided in Section 8.2.

7 Conclusion

We have shown in this technical report that the whole pyramid may be retrieved from its base by storing for each initial dart the level where it is reduced in the pyramid and the operation applied at this level. We defined one sequential and one parallel algorithm to restore the pyramid from its base. The complexity of these algorithms is respectively linear and logarithmic in the number of darts of the initial combinatorial map.

The fact that a pyramid may be unfold from its base using the function *level* and *state* show that all the information about the pyramid is captured by these two functions. We plan to use them in order to retrieve efficiently the whole receptive field [1] of one vertex or some of its features such as the set of boundary points defining the associated regions. Such an encoding may allow to compute a pyramid by storing only the top level combinatorial maps, all the other features such as an intermediate combinatorial map or a receptive field being computed from the functions *level* and *state*. We called such an encoding of a pyramid an *implicit encoding* by opposition of the classical encoding of all intermediate reduced graphs which is called an *explicit encoding*. We also plan to study other type of hierarchical encoding using default reduction rules.

Finally, an implementation of combinatorial pyramids[2] should allow to study interesting applications of our model such as: segmentation [5, 3, 4, 9], structural matching [23] or integration of moving objects.

[2]see : http://www.univ-reims.fr/leri/membre/luc/PYRAMIDES/
or http://www.prip.tuwien.ac.at/

		First(j,d)				
d	$next[d]$	j=0	j=1	j=2	j=3	j=4
-12	-6_1	-6				
-11	-5_4	-5	-5	-5	-5	
-10	5_4	5	5	5	5	
-9	-4_1	-4				
-8	-3_3	-3	-3	-3		
-7	10_1	10				
-6	-12_1	-12				
-5	6_1	6				
-4	12_1	12				
-3	11_5	11	11	11	11	11
-2	9_2	9	9			
-1	8_2	8	8			
1	7_1	7				
2	-1_1	-1				
3	-7_1	-7				
4	-8_2	-8	-8			
5	-10_1	-10				
6	-11_5	-11	-11	-11	-11	-11
7	1_1	1				
8	2_1	2				
9	-2_1	-2				
10	3_3	3	3	3		
11	4_1	4				
12	-9_2	-9	-9			

Table 4: Initialization step of Algorithm 4: values of First(.,.) and next[.]

		First(j,d)				
d	$next[d]$	j=0	j=1	j=2	j=3	j=4
-12	-11_5	-6	-11	-11	-11	-11
-11	6_1	-5	-5	-5	-5	
-10	-10_1	5	5	5	5	
-9	-8_2	-4	-8			
-8	-7_1	-3	-3	-3		
-7	5_4	10	5	5	5	
-6	-9_2	-12	-9			
-5	-12_1	6				
-4	-6_1	12				
-3	11_5	11	11	11	11	11
-2	-2_1	9	9			
-1	2_1	8	8			
1	10_1	7				
2	7_1	-1				
3	1_1	-7				
4	-3_3	-8	-8	-3		
5	3_3	-10	3	3		
6	-11_5	-11	-11	-11	-11	-11
7	8_2	1	8			
8	9_2	2	9			
9	-1_1	-2				
10	11_5	3	3	3	11	11
11	12_1	4				
12	-4_1	-9	-9			

Table 5: First step of Algorithm 4

		First(j,d)				
d	$next[d]$	j=0	j=1	j=2	j=3	j=4
-12	-11_5	-6	-11	-11	-11	-11
-11	-9_2	-5	-5	-5	-5	
-10	11_5	5	5	5	5	11
-9	-7_1	-4	-8	-3		
-8	8_2	-3	-3	-3		
-7	3_3	10	5	5	5	
-6	-8_2	-12	-9			
-5	-4_1	6	-9			
-4	-11_5	12	11	11	11	11
-3	11_5	11	11	11	11	11
-2	7_1	9	9	-3		
-1	-2_1	8	8			
1	-10_1	7	5	5	5	
2	5_4	-1	5	5	5	
3	2_1	-7	8			
4	1_1	-8	-8	-3		
5	11_5	-10	3	3	11	11
6	-11_5	-11	-11	-11	-11	-11
7	9_2	1	8	-3		
8	-1_1	2	9			
9	10_1	-2				
10	11_5	3	3	3	11	11
11	-11_5	4	-11	-11	-11	-11
12	-3_3	-9	-9	-3		

Table 6: Second step of Algorithm 4

		First(j,d)				
d	$next[d]$	j=0	j=1	j=2	j=3	j=4
-12	-11_5	-6	-11	-11	-11	-11
-11	-7_1	-5	-5	-5	-5	
-10	11_5	5	5	5	5	11
-9	9_2	-4	-8	-3		
-8	-1_1	-3	-3	-3		
-7	11_5	10	5	5	5	11
-6	8_2	-12	-9	-3		
-5	1_1	6	-9	-3		
-4	-11_5	12	-11	-11	-11	-11
-3	11_5	11	11	11	11	11
-2	3_3	9	9	-3	5	
-1	5_4	8	8	5	5	
1	11_5	7	5	5	5	11
2	11_5	-1	5	5	5	11
3	7_1	-7	8	-3		
4	-2_1	-8	-8	-3		
5	11_5	-10	3	3	11	11
6	-11_5	-11	-11	-11	-11	-11
7	10_1	1	8	-3		
8	-10_1	2	9	5	5	
9	11_5	-2	5	5	11	11
10	11_5	3	3	3	11	11
11	-11_5	4	-11	-11	-11	-11
12	2_1	-9	-9	-3		

Table 7: Third step of Algorithm 4

		First(j,d)				
d	$next[d]$	j=0	j=1	j=2	j=3	j=4
-12	-11_5	-6	-11	-11	-11	-11
-11	10_1	-5	-5	-5	-5	
-10	11_5	5	5	5	5	11
-9	11_5	-4	-8	-3	11	11
-8	11_5	-3	-3	-3	5	11
-7	11_5	10	5	5	5	11
-6	-10_1	-12	-9	-3	5	
-5	5_4	6	-9	-3	5	
-4	-11_5	12	-11	-11	-11	-11
-3	11_5	11	11	11	11	11
-2	11_5	9	9	-3	5	11
-1	11_5	8	8	5	5	11
1	11_5	7	5	5	5	11
2	11_5	-1	5	5	5	11
3	11_5	-7	8	-3	5	11
4	11_5	-8	-8	-3	5	11
5	11_5	-10	3	3	11	11
6	-11_5	-11	-11	-11	-11	-11
7	11_5	1	8	-3	5	11
8	11_5	2	9	5	5	11
9	11_5	-2	5	5	11	11
10	11_5	3	3	3	11	11
11	-11_5	4	-11	-11	-11	-11
12	3_3	-9	-9	-3	5	

Table 8: Fourth step of Algorithm 4

		First(j,d)				
d	$next[d]$	j=0	j=1	j=2	j=3	j=4
-12	-11_5	-6	-11	-11	-11	-11
-11	11_5	-5	-5	-5	-5	11
-10	11_5	5	5	5	5	11
-9	11_5	-4	-8	-3	11	11
-8	11_5	-3	-3	-3	5	11
-7	11_5	10	5	5	5	11
-6	11_5	-12	-9	-3	5	11
-5	11_5	6	-9	-3	5	11
-4	-11_5	12	-11	-11	-11	-11
-3	11_5	11	11	11	11	11
-2	11_5	9	9	-3	5	11
-1	11_5	8	8	5	5	11
1	11_5	7	5	5	5	11
2	11_5	-1	5	5	5	11
3	11_5	-7	8	-3	5	11
4	11_5	-8	-8	-3	5	11
5	11_5	-10	3	3	11	11
6	-11_5	-11	-11	-11	-11	-11
7	11_5	1	8	-3	5	11
8	11_5	2	9	5	5	11
9	11_5	-2	5	5	11	11
10	11_5	3	3	3	11	11
11	-11_5	4	-11	-11	-11	-11
12	11_5	-9	-9	-3	5	11

Table 9: Fifth step of Algorithm 4

8 Appendix

8.1 Proof of validity of Algorithm 1

Let us decompose the different actions performed by Algorithms 1 and 2 (pages 59 and 60) according to the different cases described by Theorems 2 to 5.

8.1.1 If Theorem 2 applies

Algorithm 1 is applied with parameters i and d such that $i < level(d)$ and $CDS_i(d) = d$. Let us denote by d' the dart $\varphi(d)$ if K_i is a contraction kernel and $\sigma(d)$ if K_i is a removal kernel. Algorithm 1 initializes the array *prec* to d and calls Algorithm 2 with the dart d'. Note that d' is equal $\varphi_i(d)$ or $\sigma_i(d)$ according to the type of K_i (TR-63 Proposition 15). The level of d' is thus greater or equal to $i + 1$.

The first loop of Algorithm 2(line 3) is thus performed from 1 to $i + 1$. Moreover, since *prec* is initialized to d, $\Sigma(j - 1, \alpha(d)) = \varphi_{j-1}(d)$ will be initialized to $\varphi(d) = d'$ if K_i is a contraction kernel (line 8) while $\Sigma(j-1, d) = \sigma_{j-1}(d)$ will be initialized to $\sigma(d) = d'$ if K_i is a removal kernel(line 10) with $j \in \{1, \dots, i+1\}$. In both cases, the initialization are validated by Theorem 2 and all the σ or φ successors of d up to level i are initialized. Note that since $level(d') > i$, Algorithm 1 terminates after this call on lines 16-17. This algorithm can be slightly improved by moving lines 16 and 17 to line 12 (before the loop). The present presentation has been chosen to keep all the lines concerned with the traversal of the connecting dart sequences at the end of Algorithm 2(lines 16 to 22).

8.1.2 If Theorem 3 applies

Then Algorithm 1 is applied with two parameters i and d such that $i < level(d)$ and $CDS^*_i(d) \neq \emptyset$. As in Section 8.1.1, the array *prec* is initialized to d and Algorithm 2 is called with the second dart of $CDS_i(d)$ (TR-63 Proposition 16). Let us consider a given k in $\{1, \dots, L_i(d)\}$ and $d' = d_{l_{i,d}(k)}$. Since d' belongs to $CDS_i(d)$ it will be traversed by Algorithm 2. Moreover, since d' is the first dart with a level greater than $k < i+1$, the loop described on line 13 of Algorithm 2 do not reach k until Algorithm 2 is called with d'. Therefore $prec[k]$ is not reassigned until Algorithm 2 is called with parameters i, d' and *prec* with $prec[k] = d$. Then since $k \leq level(d') < i + 1$, the loop described on line 3 consider $j = k$, and $d' = d_{l_{i,d}(k)}$ is assigned to $\Sigma(k-1, \alpha(d)) = \varphi_{k-1}(d)$ if K_i is a contraction kernel and $\Sigma(k-1, d) = \sigma_{k-1}(d)$ otherwise. In both cases, the assignments are validated by Theorem 3.

Let us now consider a given k in $\{L_i(d)+1, \ldots, i+1\}$. Note that this set contains at least $i+1$ since $L_i(d) \leq i$ (Proposition 4). By definition of $L_i(d)$, k is greater than all the dart's level contained in $CDS_i^*(d)$. Therefore, *prec*$[k]$ is not reassigned until the last dart $CDS_i(d)$ is traversed by Algorithm 2(lines 13 and 14). Since Algorithm 2 and algorithm *survive* traverse the same sequence of darts, the next encountered dart d' is equal to $\varphi_i(d)$ if K_i is a contraction kernel and $\sigma_i(d)$ otherwise(TR-63 Proposition 25). Since $level(d') \geq i+1$, the loop defined on line 3 of Algorithm 2 is performed until $i+1$ and $k \leq i+1$ is considered in the loop. Since $prec[k] = d$ and $level(d)$ is strictly greater than i, $d' = \varphi_i(d)$ is assigned to $\Sigma(k-1, \alpha(d)) = \varphi_{k-1}(d)$ if K_i is a contraction kernel and $d' = \sigma_i(d)$ is assigned to $\Sigma(k-1, d) = \sigma_{k-1}(d)$ if K_i is a removal kernel. These assignments are validated by Corollary 2 and Remark 3.

8.1.3 If Theorems 4 or 5 apply

Let us consider a dart d contracted or removed before level i. If Algorithm 1 is applied at level i with the dart d^i, Algorithm 2 will traverse $d \in CDS_i^*(d^i)$. Since $level(d) \leq i$, Algorithm 2, called with parameters i, d and *prec* will fill the array *prec* until $level(d)$ (loop on line 13).

- If d is not the last dart of $CDS_i(d^i)$, $i \in \mathcal{DS}_d$ and $\mathcal{L}_i(d)$ is defined. Let us consider a given k in $\{1, \ldots, \mathcal{L}_i(d)\}$ and $d' = d_{succ_{i,d}(k)}$. Since $\mathcal{L}_i(d) \leq level(d)$ (Definition 8), Algorithm 2 assigns d to *prec*$[k]$ when it is called with the parameter d. Moreover, $d' \in CDS_i^*(d^i)$ has an index in $CDS_i(d^i)$ greater than d by definition of $succ_{i,d}(k)$, therefore it will be traversed by Algorithm 2 after the dart d. Since d' is the first dart with a level greater than k encountered since Algorithm 2 has traversed d, *prec*$[k]$ is still assigned to d when Algorithm 2 is called with d' as second parameter. Then since $k \leq level(d') < i+1$, the loop defined on line 3 of Algorithm 2 will consider $j = k$ and thus assigns $d' = d_{succ_{i,d}(k)}$ to $\Sigma(k-1, \alpha(d)) = \varphi_{k-1}(d)$ if d is contracted and to $\Sigma(k-1, d) = \sigma_{k-1}(d)$ otherwise. These assignments are validated by Theorems 4 or 5 according to $CDS_{level(d)-1}^*(d)$.

  Let us now suppose that $\mathcal{L}_i(d) < l = level(d)$, and let us consider a given k in $\{\mathcal{L}_i(d)+1,\dots,l+1\}$. Note that this set contains at least $l+1$. The first dart with a level greater or equal to k traversed by Algorithm 2, is $\varphi_i(d^i)$, if K_i is a contraction kernel and $\sigma_i(d^i)$ if K_i is a removal kernel. Therefore, as previously, $\varphi_i(d^i)$ is assigned to $\Sigma(k-1,\alpha(d)) = \varphi_{k-1}(d)$ if K_i is a contraction kernel, and $\sigma_i(d^i)$ will be assigned to $\Sigma(k-1,d) = \sigma_{k-1}(d)$ if K_i is a removal kernel. Since $\mathcal{L}_i(d) < \mathit{level}(d)$ we have $i \in J_d$ (Proposition 13). Moreover, since d is not the last dart of $CDS_i(d^i)$, $CDS^*_{l-1}(\alpha(d))$ must be non-empty. Therefore, the above assignments are validated by Corollary 5 (see also Remark 7).

- If d is the last dart of $CDS_i(d^i)$, the loop defined on line 13 of Algorithm 2 initializes the array *prec* to d from 1 up to $\mathit{level}(d)$. Then the next call to Algorithm 2 traverses $\varphi_i(d^i)$ if K_i is a contraction kernel and $\sigma_i(d^i)$ otherwise. Since $\mathit{level}(d) < i+1$, the loop on line 3 of Algorithm 1 considers the indices from 1 to $\mathit{level}(d)$ and assigns for each $j \in \{1,\dots,\mathit{level}(d)\}$, $\varphi_i(d^i)$ to $\Sigma(j-1,\alpha(d)) = \varphi_{j-1}(d)$ if K_i is a contraction kernel and $\sigma_i(d^i)$ to $\Sigma(j-1,d) = \sigma_{j-1}(d)$ otherwise. These assignments are validated by Proposition 8.

8.2 Proof of validity of Algorithm 3

The following proposition studies the main properties of the series $(\mathit{next}_k[d])$ with k in $\{0,\dots,p\}$ where p denotes the index of the last iteration of Algorithm 4. As shown in this proposition the sequence $(\mathit{next}_k[d])_{k\in\{0,\dots,p\}}$ traverses a connecting dart sequence defined at level i and reaches the φ_i or σ_i successor of the surviving dart which defines the connecting dart sequence. Given a dart $d \in \mathcal{SD}_i$ we will thus have to consider the sequence $S_i(d)$ defined by:

$$S_i(d) = \begin{cases} CDS_i(d)\varphi_i(d) & \text{If } K_i \text{ is a contraction kernel} \\ CDS_i(d)\sigma_i(d) & \text{If } K_i \text{ is a removal kernel} \end{cases}$$

Such a sequence will be often used in the following demonstrations. Moreover, the notation $\mathit{Ind}_i(d)$ will be used to denote the index of d in $S_i(d^i)$ instead of $CDS_i(d^i)$ (Definition 8) when no ambiguity occurs.

Proposition 14 *Let us consider a dart $d \in \boldsymbol{\mathcal{D}}$, and the sequence $\mathit{next}_0[d],\dots,$ $\mathit{next}_p[d]$ which denotes the successive values taken by $\mathit{next}[d]$ in Algorithm 4. This sequence satisfies $\mathit{next}_0[d] = \sigma(d)$ and:*

- *If d is removed before level $i+1$ or if $state(i) = Removed$:*

$$\forall j \in \{0,\dots,p-1\} \quad \left\{ \begin{array}{l} next_j[d] \in CDS_i(d^i) \textit{ and} \\ Ind_i(next_j[d]) - Ind_i(d) = 2^j \end{array} \right.$$

Moreover,

$$next_p[d] = \left\{ \begin{array}{ll} \varphi_i(d^i) & \textit{If } K_i \textit{ is a contraction kernel} \\ \sigma_i(d^i) & \textit{If } K_i \textit{ is a removal kernel} \end{array} \right.$$

- *If d is contracted before level $i+1$ or if $state(i) = Contracted$:*

$$\forall j \in \{0,\dots,p-1\} \quad \left\{ \begin{array}{l} next_j[d] \in CDS_i(\alpha(d)^i) \textit{ and} \\ Ind_i(next_j[d]) - Ind_i(\alpha(d)) = 2^j \end{array} \right.$$

Moreover,

$$next_p[d] = \left\{ \begin{array}{ll} \varphi_i(\alpha(d)^i) & \textit{If } K_i \textit{ is a contraction kernel} \\ \sigma_i(\alpha(d)^i) & \textit{If } K_i \textit{ is a removal kernel} \end{array} \right.$$

Proof:

Since Algorithm 3 initializes $next[d]$ to $\sigma[d]$ we have $next_0[d] = \sigma(d)$. Let us show that the remaining properties hold for $j = 0$:

- After the initialization step:
  - If d is removed before level $i+1$ or if $state(i) = Removed$.
    - ∗ If $i \in \mathcal{DS}_d$, d is not the last nor the only dart of $CDS_i(d^i)$, therefore $\sigma(d) \in CDS_i(d^i)$ (TR-63, Theorem 1) and using Proposition 4: $level(\sigma(d)) \leq L_i(d^i) \leq i$. Moreover, in this case $\sigma(d)$ follows d in $CDS_i(d^i)$ and thus:

$$Ind_i(\sigma(d)) - Ind_i(d) = 1 = 2^0$$

    - ∗ Otherwise, d is either the last or the only dart of $CDS_i(d^i)$. In this case, $next_0[d] = \sigma(d)$ is equal to $\sigma_i(d^i)$ or $\varphi_i(d^i)$ according to K_i(TR-63, Propositions 15 and 17). We have thus $level(\sigma(d)) > i$ and Algorithm 4 does not enter in the while loop (line 13). Therefore:

$$next_0[d] = next_p[d] = \left\{ \begin{array}{ll} \varphi_i(d^i) & \text{If } K_i \text{ is a contraction kernel} \\ \sigma_i(d^i) & \text{If } K_i \text{ is a removal kernel} \end{array} \right.$$

- If d is contracted,before level $i+1$ or if $state(i) = Contracted$, $\alpha(d)$ satisfies the same property.
  - If $i \in \mathcal{DS}_{\alpha(d)}$, $\alpha(d)$ is not the last nor the only dart of $CDS_i(\alpha(d)^i)$, therefore, $\sigma(d) \in CDS_i(\alpha(d)^i)$ and $level(\sigma(d)) \leq L_i(\alpha(d)^i) \leq i$(Proposition 4). Moreover, in this case $next_0[d] = \sigma(d) = \varphi(\alpha(d))$ follows $\alpha(d)$ in $CDS_i(\alpha(d)^i)$ and we have:

$$Ind_i(\varphi(\alpha(d))) - Ind_i(\alpha(d)) = 1 = 2^0$$

  - Otherwise, $\sigma(d)$ is equal to $\sigma_i(\alpha(d)^i)$ or $\varphi_i(\alpha(d)^i)$ according to K_i(TR-63, Propositions 15 and 17). In both cases, $level(\sigma(d)) > i$ and Algorithm 4 does not enter in the while loop. We have thus:

$$next_0[d] = next_p[d] = \begin{cases} \varphi_i(\alpha(d)^i) & \text{If } K_i \text{ is a contraction kernel} \\ \sigma_i(\alpha(d)^i) & \text{If } K_i \text{ is a removal kernel} \end{cases}$$

The property is thus true before the first iteration for all darts. Let us suppose it true until a given iteration k for any $d \in \boldsymbol{\mathcal{D}}$ and let us show that the property holds after the $(k+1)^{th}$ iteration.

- If d is removed before level $i+1$ or if $state(i) = Removed$.

According to our recurrence hypothesis, $next_k[d]$ either belongs to $CDS_i(d^i)$ or is equal to $\sigma_i(d^i)$ or $\varphi_i(d^i)$ according to K_i. In this latter case $level(next_k[d]) > i$ and Algorithm 4 terminates on $next_k[d]$(line 13). We have thus $k = p$ with:

$$next_p[d] = \begin{cases} \varphi_i(d^i) & \text{If } K_i \text{ is a contraction kernel} \\ \sigma_i(d^i) & \text{If } K_i \text{ is a removal kernel} \end{cases}$$

If $next_k[d] \in CDS_i(d^i)$, $level(next_k[d]) \leq L_i(d^i) \leq i$ and Algorithm 4 performs an additional iteration (line 13). Moreover, using our recurrence hypothesis, the dart $next_k[d]$ satisfies:

$$\begin{cases} next_k[d] \in CDS_i(d^i) \text{ and} \\ Ind_i(next_k[d]) - Ind_i(d) = 2^k \end{cases}$$

Note that we have in this case $next_k[d]^i = d^i$ (since $next_k[d] \in CDS_i(d^i)$).

– If $next_k[d]$ is removed we have $next_{k+1}[d] = next_k[next_k[d]]$.

Since $next_{k+1}[d] = next_k[next_k[d]]$, we can apply our recurrence hypothesis on $next_k[d]$. Therefore, $next_{k+1}[d]$ either belongs to $CDS_i(next_k[d]^i)$ or is equal to $\varphi_i(next_k[d]^i)$ or $\sigma_i(next_k[d]^i)$ according to K_i.

If $next_{k+1}[d] \in \{\varphi_i(next_k[d]^i), \sigma_i(next_k[d]^i)\}$ we have:

$$next_{k+1}[d] = \left\{ \begin{array}{lcll} \varphi_i(next_k[d]^i) & = & \varphi_i(d^i) & \text{If } K_i \text{ is a contraction kernel} \\ \sigma_i(next_k[d]^i) & = & \sigma_i(d^i) & \text{If } K_i \text{ is a removal kernel} \end{array} \right.$$

In this case $level(next_{k+1}[d]) > i$ and Algorithm 4 exit from the while loop at this iteration (line 13). We have thus $p = k + 1$.

If $next_{k+1}[d] \in CDS_i(next_k[d]^i)$, we have by our recurrence hypothesis:

$$\left\{ \begin{array}{l} next_{k+1}[d] \in CDS_i(next_k[d]^i) = CDS_i(d^i) \text{ and} \\ Ind_i(next_{k+1}[d]) - Ind_i(next_k[d]) = 2^k \end{array} \right.$$

Thus:

$$\begin{aligned} & Ind_i(next_{k+1}[d]) - Ind_i(d) = \\ & Ind_i(next_{k+1}[d]) - Ind_i(next_k[d]) + Ind_i(next_k[d]) - Ind_i(d) \\ & \qquad = 2.2^k \\ & \qquad = 2^{k+1} \end{aligned} \tag{21}$$

Moreover, $level(next_{k+1}[d]) \leq L_i(d^i) \leq i$ and Algorithm 4 performs an additional iteration.

– If $next_k[d]$ is contracted, $\alpha(next_k[d])$ is also contracted and $next_{k+1}[d] = next_k[\alpha(next_k[d])]$.

Since $next_{k+1}[d] = next_k[\alpha(next_k[d])]$ we can apply our recurrence hypothesis on $\alpha(next_k[d])$. Therefore, $next_{k+1}[d]$ either belongs to $CDS_i(\alpha(\alpha(next_k[d]))^i) = CDS_i(next_k[d]^i)$ or is equal to $\sigma_i(\alpha(\alpha(next_k[d]))^i)$ or $\varphi_i(\alpha(\alpha(next_k[d]))^i)$ according to K_i.

  * If $next_{k+1}[d] \in \{\varphi_i(\alpha(\alpha(next_k[d]))^i), \sigma_i(\alpha(\alpha(next_k[d]))^i)\}$ we

have by our recurrence hypothesis:

$$next_{k+1}[d] = \begin{cases} \text{If } K_i \quad \text{is a contraction kernel} \\ \qquad \varphi_i(\alpha(\alpha(next_k[d]))^i) = \varphi_i(next_k[d]^i) = \varphi_i(d^i) \\ \text{If } K_i \quad \text{is a removal kernel} \\ \qquad \sigma_i(\alpha(\alpha(next_k[d]))^i) = \sigma_i(next_k[d]^i) = \sigma_i(d^i) \end{cases}$$

In this case $level(next_{k+1}[d]) > i$ and Algorithm 4 exit from the while loop at this iteration (line 13). We have thus $p = k+1$.

$*$ If $next_{k+1}[d] \in CDS_i(next_k[d]^i)$ we have:

$$\begin{cases} next_{k+1}[d] = next_k[\alpha(next_k[d])] \in CDS_i(next_k[d]^i) = CDS_i(d^i) \text{ and} \\ Ind_i(next_{k+1}[d]) - Ind_i(next_k[d]) = 2^k \end{cases}$$

We obtain as previously (equation 21):

$$Ind_i(next_{k+1}[d]) - Ind_i(d) = 2^{k+1}$$

- If d is contracted before level $i+1$ or if $state(i) = Contracted$:

As previously, if $next_k[d] \notin CDS_i(\alpha(d)^i)$ we have $k = p$ with:

$$next_p[d] = \begin{cases} \varphi_i(\alpha(d)^i) & \text{If } K_i \text{ is a contraction kernel} \\ \sigma_i(\alpha(d)^i) & \text{If } K_i \text{ is a removal kernel} \end{cases}$$

If $next_k[d] \in CDS_i(\alpha(d)^i)$, $level(next_k[d]) \leq i$ and Algorithm 4 performs thus an additional iteration (line 13). Using our recurrence hypothesis the dart $next_k[d]$ satisfies:

$$\begin{cases} next_k[d] \in CDS_i(\alpha(d)^i) \text{ and} \\ Ind_i(next_k[d]) - Ind_i(\alpha(d)) = 2^k \end{cases}$$

We have thus $next_k[d]^i = \alpha(d)^i$.

– If $next_k[d]$ is removed we have $next_{k+1}[d] = next_k[next_k[d]]$.
As previously, using our recurrence hypothesis on $next_k[d]$, $next_{k+1}[d]$ belongs either to $CDS_i(next_k[d]^i)$ or $\{\varphi_i(next_k[d]), \sigma_i(next_k[d])\}$:

* If $next_{k+1}[d] \in \{\varphi_i(next_k[d]), \sigma_i(next_k[d])\}$, we have:

$$next_{k+1}[d] = \left\{ \begin{array}{lcll} \varphi_i(next_k[d]^i) & = & \varphi_i(\alpha(d)^i) & \text{If } K_i \text{ is a contraction kernel} \\ \sigma_i(next_k[d]^i) & = & \sigma_i(\alpha(d)^i) & \text{If } K_i \text{ is a removal kernel} \end{array} \right.$$

  In this case $level(next_{k+1}[d]) > i$ and Algorithm 4 exit from the while loop at this iteration (line 13). We have thus $p = k+1$.

* If $next_{k+1}[d] \in CDS_i(next_k[d]^i)$, we have by our recurrence hypothesis on $next_k[d]$:

$$\left\{ \begin{array}{l} next_{k+1}[d] \in CDS_i(next_k[d]^i) = CDS_i(\alpha(d)^i) \text{ and} \\ Ind_i(next_{k+1}[d]) - Ind_i(next_k[d]) = 2^k \end{array} \right.$$

  Therefore:

$$\begin{array}{r} Ind_i(next_{k+1}[d]) - Ind_i(\alpha(d)) = \\ Ind_i(next_{k+1}[d]) - Ind_i(next_k[d]) + Ind_i(next_k[d]) - Ind_i(\alpha(d)) \\ = 2.2^k \\ = 2^{k+1} \end{array} \tag{22}$$

- If $next_k[d]$ is contracted, then $\alpha(next_k[d])$ is also contracted and we have $next_{k+1}[d] = next_k[\alpha(next_k[d])]$.

  According to our recurrence hypothesis on $\alpha(next_k[d])$, $next_{k+1}[d]$ belongs either to $CDS_i(\alpha(\alpha(next_k[d]))^i) = CDS_i(next_k[d]^i)$ or $\{\varphi_i(next_k[d]^i), \sigma_i(next_k[d]^i)\}$ (this last relation was obtained by using the relation $\alpha(\alpha(next_k[d]))^i = next_k[d]^i$)

  * If $next_{k+1}[d] \in \{\varphi_i(next_k[d]^i), \sigma_i(next_k[d]^i)\}$ we have:

$$\begin{array}{l} next_{k+1}[d] = \\ \left\{ \begin{array}{lcll} \varphi_i(next_k[d]^i) & = & \varphi_i(\alpha(d)^i) & \text{If } K_i \text{ is a contraction kernel} \\ \sigma_i(next_k[d]^i) & = & \sigma_i(\alpha(d)^i) & \text{If } K_i \text{ is a removal kernel} \end{array} \right. \end{array} \tag{23}$$

    As previously, we have in this case $p = k+1$.

* If $next_{k+1}[d] \in CDS_i(next_k[d]^i)$ we have by our recurrence hypothesis on $\alpha(next_k[d])$:

$$\begin{array}{l} next_{k+1}[d] \in CDS_i(next_k[d]^i) = CDS_i(\alpha(d)^i) \\ Ind_i(next_{k+1}[d]) - Ind_i(next_k[d]) = 2^k \end{array}$$

  As previously:

$$Ind_i(next_{k+1}[d]) - Ind_i(\alpha(d)) = 2^{k+1}$$

□

Corollary 6 *With the same notations and hypothesis as Proposition 14 we have:*

$$\begin{array}{rrcl} & level(next_p[d]) & > & i \\ \forall j \in \{0,\dots,p-1\} & level(next_j[d]) & \leq & i \end{array}$$

Proof:

Since $next_p[d]$ is the φ_i or σ_i successor of either d^i or $\alpha(d)^i$(Proposition 14), we have $next_p[d] \in \mathcal{SD}_i$ and thus $level(d_p) > i$. Moreover, using Proposition 14 we have:

- If d is removed before level $i+1$ or if $state(i) = Removed$:

$$\forall j \in \{0,\dots,p-1\} \quad \begin{cases} next_j[d] \in CDS_i(d^i) \text{ with} \\ Ind_i(next_j[d]) - Ind_i(d) > 0 \end{cases}$$

  Therefore, given an index $j \in \{0,\dots,p-1\}$, if $next_j[d] = d^i$ we have $Ind_i(next_j[d]) = Ind_i(d^i) \leq Ind_i(d)$. This last hypothesis being refused by Proposition 14, we have $next_j[d] \neq d^i$ and thus $next_j[d] \in CDS_i^*(d^i)$. Thus (Proposition 4):

$$\forall j \in \{0,\dots,p-1\} \quad level(next_j[d]) \leq L_i(d^i) \leq i$$

- If d is contracted before level $i+1$ or if $state(i) = Contracted$:

$$\forall j \in \{0,\dots,p-1\} \quad \begin{cases} next_j[d] \in CDS_i(\alpha(d)^i) \text{ with} \\ Ind_i(next_j[d]) - Ind_i(\alpha(d)) > 0 \end{cases}$$

  As previously, given $j \in \{0,\dots,p-1\}$, the hypothesis $next_j[d] = \alpha(d)^i$ contradict Proposition 14. Therefore, $next_j[d] \in CDS_i^*(\alpha(d)^i)$ and:

$$\forall j \in \{0,\dots,p-1\} \quad level(next_j[d]) \leq L_i(\alpha(d)^i) \leq i$$

□

Remark 8 *Note that Corollary 6 may be used to characterize* $next_p[d]$ *since this dart is the first one in the sequence* $(next_0[d], \ldots, next_p[d])$ *with a level strictly greater than* i*.*

As an example, let us consider the dart -11 in Table 4 (line 1). The successive values taken by $next[-11]$ are equal to $-5, 6, -9, -7, 10, 11$(Tables 4 to 9(line 1)). The dart -11 is not contracted nor removed at level 4 but since K_4 is a removal kernel the dart -11 fulfills the first requirement of Proposition 14. Therefore, the series $-5, 6, -9, -7, 10, 11$ is included in $CDS_4(-11)$ (Figure 16 or 5). The index of these darts within $CDS_4(-11)$ is respectively equal to: $1, 2, 4, 8, 16, 20$ (we supposed that the index of -11 is 0). The exponential growth of the index of $next[-11]$ is thus verified.

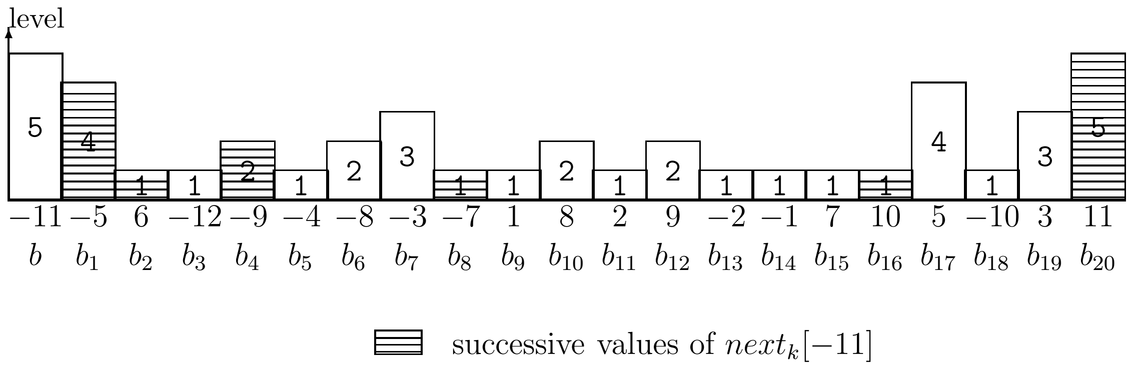

level
5
4
1
1
2
1
2
3
1
1
2
1
2
1
1
1
1
4
1
3
5
−11 −5 6 −12 −9 −4 −8 −3 −7 1 8 2 9 −2 −1 7 10 5 −10 3 11
b b_1 b_2 b_3 b_4 b_5 b_6 b_7 b_8 b_9 b_{10} b_{11} b_{12} b_{13} b_{14} b_{15} b_{16} b_{17} b_{18} b_{19} b_{20}
▤ successive values of $next_k[-11]$

Figure 16: The connecting dart sequence $CDS_4(-11)$ with the successive values taken by $next_k[-11]$ during Algorithm 4

In the same way, the dart 3 is contracted at level 3. The successive values of $next[3]$ are equal to $-7, 1, 2, 7, 11$(Tables 4 to 9(line 15)). Since 3 is contracted before level 4, this sequence of darts belongs to $CDS_4(-3^4) = CDS_4(-11)$ (in this this case, the darts 3 and -3 belong to the same connecting dart sequence at level 4). The indices of $-7, 1, 2, 7, 11$ in $CDS_4(-11)$ are respectively equal to $8, 9, 11, 15, 20$. Since $Ind_4(-3) = 7$, the exponential growth of $Ind_4(next[-3]) - Ind_4(-3)$ is verified.

Proposition 15 *Using the same hypothesis and notations as Proposition 14, given a dart $d \in \boldsymbol{\mathcal{D}}$:*

- *If $d \in \mathcal{SD}_i$ and if one of the two following proposition is true:*

    1. *K_i is a removal kernel and $CDS_i(d) = d$*
    2. *K_i is a contraction kernel and $CDS_i(\alpha(d)) = \alpha(d)$*

    Then:
    $$\forall j \in \{0, \dots, i\} \quad First(j, d) = \sigma_i(d)$$

- *If $d \in \cup_{j=0}^{i} K_j$ and $i \in I_d$*

    - *If d is removed:*
    $$\forall j \in \{0, \dots, i\}\, First(j, d) = \begin{cases} \varphi_i(d^i) & \textit{If } K_i \textit{ is a contraction kernel} \\ \sigma_i(d^i) & \textit{If } K_i \textit{ is a removal kernel} \end{cases}$$

    - *If d is contracted:*
    $$\forall j \in \{0, \dots, i\}\, First(j, d) = \begin{cases} \varphi_i(\alpha(d)^i) & \textit{If } K_i \textit{ is a contraction kernel} \\ \sigma_i(\alpha(d)^i) & \textit{If } K_i \textit{ is a removal kernel} \end{cases}$$

In all cases, Algorithm 4 does not enter in the while loop and the sequence of successive values of $next[d]$ is reduced to $next_0[d]$.

Proof:

Let us follow the decomposition of the proposition.

- If $d \in \mathcal{SD}_i$ and $CDS_i(d) = d$

    If K_i is a removal kernel and $CDS_i(d) = d$, we have by Theorem 2:
    $$\forall j \in \{0, \dots, i\} \quad \sigma_j(d) = \sigma(d)$$
    and more particularly $\sigma_i(d) = \sigma(d)$.

    In the same way, if K_i is a contraction kernel and $CDS_i(\alpha(d)) = \alpha(d)$, we have by Theorem 2:
    $$\forall j \in \{0, \dots, i\} \quad \varphi_j(\alpha(d)) = \sigma_j(d) = \varphi(\alpha(d)) = \sigma(d)$$
    and thus $\sigma_i(d) = \sigma(d)$.

In both cases Algorithm 4 is called with a parameter $d' = \sigma(d)$ whose level is greater than i. The variable *max_level* (Algorithm 4, line 4) is thus initialized to $i+1$ and the array $First(.,d)$ is filled by $\sigma(d) = \sigma_i(d)$ from level 0 to level i(Algorithm 4, lines 7,8). Moreover, since $level(\sigma(d)) > i$, Algorithm 4 does not enter in the while loop (line 13) and $next_0[d] = \sigma(d)$ is the first and last value of $next[d]$.

- If $d \in \cup_{j=0}^{i} K_j$ and $i \in I_d$.
  - If d is removed, we have by Proposition 8:
    $$\sigma(d) = \begin{cases} \varphi_i(d^i) & \text{If } K_i \text{ is a contraction kernel} \\ \sigma_i(d^i) & \text{If } K_i \text{ is a removal kernel} \end{cases}$$
    Therefore, in both cases, Algorithm 4 is called with a parameter $d' = \sigma(d)$ with a level strictly greater than i. As previously, Algorithm 4 fills the array $First(.,d)$ by $\sigma(d)$ from level 0 to level i and terminates without entering in the while loop.
  - If d is contracted, $\alpha(d)$ is also contracted and Proposition 8 applied to $\alpha(d)$ gives:
    $$\varphi(\alpha(d)) = \sigma(d) = \begin{cases} \varphi_i(\alpha(d)^i) & \text{If } K_i \text{ is a contraction kernel} \\ \sigma_i(\alpha(d)^i) & \text{If } K_i \text{ is a removal kernel} \end{cases}$$
    In both cases, Algorithm 4 is called with a parameter $d' = \sigma(d)$ such that $level(\sigma(d)) > i$. As previously, Algorithm 4 fills the array $First(.,d)$ by $\sigma(d)$ from level 0 to level i and terminates without entering in the while loop.

□

Proposition 15 considers the cases where a dart d is either the only or the last dart of the connecting dart sequence which contains it. In these cases, Algorithm 4 does not enter in the while loop (line 13).

If Algorithm 4 does enter in the while loop for each iteration such that $min\_level[d'']$ is greater than $min\_level[d]$, the algorithm fills the array $First(.,d)$ from $min\_level[d]$ to $min\_level[d'']$ (lines 15 and 21). Let us denote by $min_k[d]$ the value of $min\_level[d]$ at iteration k. Intuitively, $min_k[d]$ encodes the level of the next dart required by the processor associated to d in order to add an entry to the array $First(.,d)$ (comments in Algorithm 4).

The following proposition establishes the main properties of the series $min_k[d]$.

Proposition 16 *With the same notations as Proposition 14, given a dart $d \in \boldsymbol{\mathcal{D}}$, $min_0[d], \ldots, \min_p[d]$ denote the values of $min\_level[d]$ during the successive iterations of Algorithm 4, we have:*

$$\begin{cases} \forall j \in \{0, \ldots, p-1\} \quad \min(level(next_j[d]), i+1) + 1 \leq min_j[d] < i+2 \\ min_p[d] = i+2 \end{cases}$$

Proof:

Let us first show that the condition is true before the loop. For each dart $d \in \boldsymbol{\mathcal{D}}$, we have $next_0[d] = \sigma(d)$ and $min_0[d]$ is set to (lines 4 and 10):

$$min_0[d] = \min(level(next_0[d]), i+1) + 1$$

- If $level(next_0[d]) \leq i$, $min_0[d] = level(next_0[d]) + 1 < i+2$. Moreover, Algorithm 4 performs in this case at least one iteration (Remark 8). Therefore $next_0[d]$ is not the last dart of the sequence $next_0[d] \ldots next_p[d]$.
- If $level(next_0[d]) > i$, $min_0[d] = i+2$. In this case, we have $p = 0$(line 13).

The condition is thus true for each dart $d \in \boldsymbol{\mathcal{D}}$ before the while loop. Let us now suppose that the condition is true for all darts until a given iteration k.

Given one dart $d \in \boldsymbol{\mathcal{D}}$ such that the value of $next[d]$ at iteration k is not the last one ($k \neq p$), we have by our recurrence hypothesis:

$$\min(level(next_k[d]), i+1) + 1 \leq min_k[d] < i+2$$

Let us denote by d'' the dart $next_k[d]$ if $next_k[d]$ is removed and $\alpha(next_k[d])$ if $next_k[d]$ is contracted (Algorithm 4, line 15). Note that we have $next_{k+1}[d] = next_k[d'']$(line 24). In the same way, let us denote by p'' the index of the last iteration of Algorithm 4 associated to d''. The successive values taken by $next[d'']$ are thus equal to $(next_j[d''])_{j \in \{0, \ldots, p''\}}$.

If $p'' \leq k$, the sequence $(next_j[d''])_{j \in \{0, \ldots, p''\}}$ terminates at or before iteration k. Using our recurrence hypothesis we have $min_{p''}[d''] = i+2$. Moreover, since $min_k[d] < i+2$ by hypothesis we have $min_k[d] < min_{p''}[d'']$ and $min_{k+1}[d]$ is set to $min_{p''}[d''] = i+2$ (line 18 and 22).

Otherwise, using our recurrence hypothesis on d'', and the fact that $next_{k+1}[d] = next_k[d'']$:

$$\min(level(next_{k+1}[d]), i+1) + 1 \leq min_k[d''] < i+2$$

- If $min_k[d''] \leq min_k[d]$, $min_{k+1}[d]$ is set to $min_k[d]$.

$$\min(level(next_{k+1}[d]), i+1) + 1 \leq min_k[d''] \leq min_{k+1}[d] < i+2$$

- If $min_k[d''] > min_k[d]$ we have $min_{k+1}[d] = min_k[d'']$(line 22). Thus:

$$\min(level(next_{k+1}[d]), i+1) + 1 \leq min_k[d''] = min_{k+1}[d] < i+2$$

The condition holds thus at iteration $k+1$. $\square$

Note that since the array $First(.,d)$ is filled up to $min_k[d]-2$ Algorithm 4 (lines 7,8 and 20,21) the value of $min_k[d]$ may be deduced from Tables 4 to 9 by adding 2 to the Column's index of the last filled entries on each line. Let us consider the dart -11 (line 2) on Table 4. The value of $next_0[d]$ is indicated on the second column and equal to -5 for the dart -11. Since $First(.,-11)$ is filled up to index 3 we have $min_0[-11] = 3+2 = 5$. Moreover, the level of $next_0[-11] = -5$ being equal to 4 we have:

$$\min(level(-5), 4+1) + 1 = 5 \leq min_0[-11] = 5 < 6$$

Since the dart -5 is removed at level 4, the value of d'' in Algorithm 4 is equal to -5 with $min_0[-5] = 0+2 = 2$. We have thus: $next_1[-11] = next_0[-5] = 6$. Since $min_0[-5] \leq min_0[-11]$, the value of $min_0[-11]$ remains unchanged at iteration 1 and:

$$\min(level(next_1[-11]), 4{+}1){+}1 = 2 \leq min_0[-5] \leq min_0[-11] = min_1[-11] < 6$$

Conversely, let us consider the dart 7 on Table 4. since the dart $next_0[7] = 1$ is contracted at level 1, the value of d'' in Algorithm 4 is equal to -1 with $min_0[7] = 2 < min_0[-1] = 3$ and $next_1[7] = next_0[-1] = 8$. The value of $min_1[7]$ is thus set to $min_0[-1]$ and we have:

$$\min(level(next_1[7]), 4+1) + 1 = 3 \leq min_0[-1] = min_1[7] < 6$$

Lemma 1 *With the same notations as Proposition 14, given a dart $d \in \boldsymbol{\mathcal{D}}$, if $next_0[d], \ldots, next_p[d]$ and $min_0[d], \ldots, \min_p[d]$ denote respectively the values of $next[d]$ and $min\_level[d]$ during the successive iterations of Algorithm 4, the following conditions are invariant for each iteration:*

- *If d is removed before level $i+1$ or if $state(i) = Removed$.*

1. *For each $k \in \{0, \dots, p-1\}$, any dart in the sequence $]d, \dots, next_k[d]] \trianglelefteq CDS_i(d^i)$ has a level strictly lower than $min_k[d]$:*

   $$\forall k \in \{0, \dots, p-1\}\ \forall b \in]d, \dots, next_k[d]] \quad level(b) < min_k[d]$$

2. *For each $k \in \{0, \dots, p\}$ and each $j \in \{0, \dots, min_k[d]-2\}$, $First(j, d)$ is the first dart of the sequence $]d, \dots, next_k[d]]$ with a level strictly greater than j:*

   $$\forall j \in \{0, \dots, min_k[d]-2\} \quad \begin{cases} level(First(j, d)) > j \text{ and} \\ \forall b \in]d, \dots, First(j, d)[\quad level(b) \leq j \end{cases}$$

- *If d is contracted before level $i + 1$ or if $state(i) = Contracted$*

  1. *For each $k \in \{0, \dots, p-1\}$, any dart in the sequence $]\alpha(d), \dots, next_k[d]] \trianglelefteq CDS_i(\alpha(d)^i)$ has a level strictly less than $min_k[d]$.*

     $$\forall k \in \{0, \dots, p-1\}\ \forall b \in]\alpha(d), \dots, next_k[d]] \quad level(b) < min_k[d]$$

  2. *For each $k \in \{0, \dots, p\}$ and each $j \in \{0, \dots, min_k[d]-2\}$, $First(j, d)$ is the first dart of the sequence $]\alpha(d), \dots, next_k[d]]$, with a level strictly greater than j:*

     $$\begin{array}{l} level(First(j, d)) > j \text{ and} \\ \forall b \in]\alpha(d), \dots, First(j, d)[\quad level(b) \leq j \end{array}$$

Proof:

The following demonstration will be restricted to the case d is removed before level $i + 1$ or $state(i) = Removed$, since the other case may be easily adapted from it.

After the initialization step, since d is removed, $next_0[d] = \sigma(d)$ either belongs to $CDS_i(d^i)$ or is equal to $\varphi_i(d^i)$ or $\sigma_i(d^i)$ according to K_i (Proposition 14).

Case 1: $next_0[d] = \sigma(d) \in \{\sigma_i(d^i), \varphi_i(d^i)\}$.

We have $level(\sigma(d)) > i$(Corollary 6) and $min_0[d] = i+2$(Proposition 16). Moreover, the variable *max_level* (Algorithm 4, line 4) is initialized to $i + 1$ and the array $First(., d)$ is filled by $\sigma(d)$ from level 0 to level i (Algorithm 4, lines 7,8). The condition $level(First(j, d) = \sigma(d)) > j$ for each j in $\{0, \dots, i\}$ is thus insured.

Since $\sigma(d)$ follows d in $S_i(d^i)$, the interval $]d, \dots, Fist(j, d)[$ is empty for any $j \in \{0, \dots, i\}$. The second property of the second invariant is thus trivially satisfied. Nothing remains to be demonstrated in this case since $p = 0$.

Case 2: $next_0[d] = \sigma(d) \in CDS_i(d^i)$.

We have $level(\sigma(d)) \leq i$(Corollary 6).The variable max_level (Algorithm 4, line 4) is thus initialized to $level(\sigma(d))$ and the array $First(., d)$ is filled up to $level(\sigma(d))$(lines 7,8) by $\sigma(d)$. Note that we have in this case $min_0[d] = max\_level + 1 = level(\sigma(d)) + 1$(line 10), moreover:

$$\forall j \in \{0, \dots, level(\sigma(d)) - 1\}\ level(First(j, d)) = level(\sigma(d)) > j$$

The first property of the second invariant is thus satisfied.

Since $\sigma(d)$ follows d in $CDS_i(d^i)$, the interval $]d, \dots, next_0[d] = \sigma(d)]$ is reduced to $\{next_0[d]\}$ with:

$$level(next_0[d]) = level(\sigma(d)) < min_0[d] = level(\sigma(d)) + 1$$

The first invariant is thus satisfied.

As previously, the interval $]d, \dots, First(j, d)[$ is empty for any $j \in \{0, \dots, level(\sigma(d)\}$. The second property of the second invariant is thus trivially true.

The conditions are thus true before the loop. Let us suppose them true until level k. If $k = p$ nothing remains to be demonstrated. Otherwise, Using our recurrence hypothesis we have:

$$\forall b \in]d, \dots, next_k[d]] \quad level(b) < min_k[d] \tag{24}$$

Case 3: $next_k[d]$ is removed before level $i + 1$ or $state(i) = Removed$.

Using Proposition 14, $next_k[d] \in CDS_i(d^i)$. Moreover, by definition of the sequence $(next_j[d])_{j\in\{0,\dots,p\}}$ we have $next_{k+1}[d] = next_k[next_k[d]]$. Let us first suppose that $level(next_{k+1}[d]) \leq i$. In this case, the Algorithm 4 associated to $next_k[d]$ performs at least $k + 1$ iteration (Corollary 6). Using our recurrence hypothesis on $next_k[d]$ we obtain:

$$\forall b \in]next_k[d], \dots, next_k[next_k[d]] = next_{k+1}[d]]\ level(b) < min_k[next_k[d]]$$

- If $min_k[next_k[d]] \leq min_k[d]$, using equation 24 we have:

$$\forall b \in]d, \ldots, next_{k+1}[d]] \quad level(b) < min_k[d]$$

Therefore, since $min_k[d]$ and the array $First(., d)$ are not modified (line 18) both invariants remain true at iteration $k+1$.

- If $min_k[next_k[d]] > min_k[d]$, then $min_{k+1}[d]$ is set to $min_k[next_k[d]]$ (line 22). Using equation 24 we obtain:

$$\left.\begin{array}{ll} \forall b \in]d, \ldots, next_k[d]] & level(b) < min_k[d] < min_{k+1}[d] \\ \forall b \in]next_k[d], \ldots, next_{k+1}[d]] & level(b) < min_k[next_k[d]] = min_{k+1}[d] \end{array}\right\}$$
$$\Longrightarrow \forall b \in]d, \ldots, next_{k+1}[d]]\ level(b) < min_{k+1}[d]$$

Moreover, the array $First(., next_k[d])$ is copied in $First(., d)$ from the index $min_k[d]-1$ to $min_k[next_k[d]]-2$(lines 20,21). Since this copy preserves the indexes we have:

$$\begin{array}{r} \forall j \in \{min_k[d]-1, \ldots, min_k[next_k[d]]-2\}, \\ First(j, next_k[d]) = First(j, d) > j \end{array}$$

The above property is also true for any $j \in \{0, \ldots, min_k[d] - 2\}$ by our recurrence hypothesis. Therefore, since $min_{k+1}[d] = min_k[next_k[d]]$:

$$\forall j \in \{0, \ldots, min_{k+1}[d]-2\}\ First(j, d) > j$$

The first property of the second invariant is thus preserved at level $k+1$.

Using the second property of the second invariant on $next_k[d]$ we have:

$$\left.\begin{array}{ll} \forall j \in & \{min_k[d]-1, \ldots, min_k[next_k[d]]-2\} \\ \forall b \in &]next_k[d], \ldots, First(j, next_k[d])[\end{array}\right\} level(b) \leq j \tag{25}$$

Note that $First(j, next_k[d])$ is equal to $First(j, d)$ due to the copy of $First(j, next_k[d])$ into $First(j, d)$ for $j \in \{min_k[d] - 1, \ldots, min_k[next_k[d]] - 2\}$(Algorithm 4, lines 21,22 and above).

Using the first invariant on d we obtain:

$$\forall b \in]d, \ldots, next_k[d]]\ level(b) < min_k[d]$$

Thus:

$$\left.\begin{array}{ll}\forall j \in & \{min_k[d]-1,\ldots,min_k[next_k[d]]-2\} \\ \forall b \in &]d,\ldots,next_k[d]]\end{array}\right\}$$
$$level(b) \le min_k[d]-1 \le j$$

Therefore, using the above equation and equation 25 we obtain:

$$\left.\begin{array}{ll}\forall j \in & \{min_k[d]-1,\ldots,min_k[next_k[d]]-2\} \\ \forall b \in &]d,\ldots,First(j,d)[\end{array}\right\} level(b) \le j$$

Since this property is also valid for $j \in \{0,\ldots,min_k[d]-2\}$ by hypothesis, the second invariant is true at level $k+1$.

If $level(next_{k+1}[d]) > i$, the Algorithm 4 associated to d terminates on $next_{k+1}[d]$ (Corollary 6) and we have $k+1 = p$. In this case, the processor running Algorithm 4 on $next_k[d]$ has terminated before iteration k. We have thus $min_k[next_k[d]] = i+2$(Proposition 16). Since $min_k[d] < i+2$ we have $min_k[d] < min_k[next_k[d]]$ and $min_{k+1}[d]$ is set to $min_k[next_k[d]] = i+2$. As previously, the array $First(.,next_k[d])$ is copied in $First(.,d)$ from index $min_k[d]-1$ to $min_k[next_k[d]]-2 = i$. Since this copy preserves the indexes we have:

$$\forall j \in \{0,\ldots,i\}\ level(First(j,d)) > j$$

The first property of the second invariant is thus preserved at level $k+1 = p$. Since the demonstration of the second property of the second invariant do not use the first invariant on $next_k[d]$ (which is not valid for the last iteration of the algorithm associated to $next_k[d]$) it may be demonstrated as previously by using equation 25. We obtain thus:

$$\forall j \in \{0,\ldots,i\} \quad \left\{\begin{array}{l} level(First(j,d)) > j \text{ and} \\ \forall b \in]d,\ldots,First(j,d)[\quad level(b) \le j\end{array}\right.$$

Case 4: $next_k[d]$ is contracted before level $i+1$ or $state(i) = Contracted$

The proof of this case being similar to the one of the case (3), we will just outline the main parts of the proof.

Using Proposition 14, $next_k[d] \in S_i(\alpha(d)^i)$. Moreover, by definition of the sequence $(next_j[d])_{j\in\{0,\ldots,p\}}$ we have $next_{k+1}[d] = next_k[\alpha(next_k[d])]$.

If $level(next_{k+1}[d]) \le i$, Algorithm 4 associated to $\alpha(next_k[d])$ performs at least $k+1$ iteration. Using our recurrence hypothesis on $\alpha(next_k[d])$ we obtain:

$$\forall b \in]\alpha(\alpha(next_k[d])) = next_k[d], \dots, next_{k+1}[d]] \; level(b) < min_k[\alpha(next_k[d])]$$

- If $min_k[\alpha(next_k[d])] \le min_k[d]$, we obtain by using equation 24:

  $$\forall b \in]d, \dots, next_{k+1}[d]] \quad level(b) < min_k[d]$$

  Moreover, since in this case, the array $First(., d)$ and the variable $min_k[d]$ remain unchanged(line 18) the invariants remain true at iteration $k+1$.

- If $min_k[\alpha(next_k[d])] > min_k[d]$ then $min_{k+1}[d]$ is set to $min_k[\alpha(next_k[d])]$ (line 22). Using equation 24 we obtain:

  $$\forall b \in]d, \dots, next_{k+1}[d]] \; level(b) < min_{k+1}[d]$$

  As previously, the copy of $First(j, \alpha(next_k[d]))$ to $First(j, d)$ for $j \in \{min_k[d]-1, \dots, min_{k+1}[d]\}$ preserves the indexes. We have thus:

  $$\forall j \in \{0, \dots, min_{k+1}[d]-2\} \; level(First(j, d)) > j$$

  The first property of the second invariant is thus preserved at iteration $k+1$. Using the second property of the second invariant on $\alpha(next_k[d])$ we have:

  $$\left.\begin{array}{ll} \forall j \in & \{min_k[d]-1, \dots, min_k[\alpha(next_k[d])]-2\} \\ \forall b \in &]\alpha(\alpha(next_k[d])) = next_k[d], \dots, First(j, d)[\end{array}\right\} level(b) \le j$$

  Since $min_k[\alpha(next_k[d])] = min_{k+1}[d]$, using the above equation and the first invariant on d we obtain as previously:

  $$\left.\begin{array}{ll} \forall j \in & \{0, \dots, min_{k+1}[d]-2\} \\ \forall b \in &]d, \dots, First(j, d)[\end{array}\right\} level(b) \le j$$

  The second invariant remains thus true at level $k+1$.

If $level(next_{k+1}[d]) > i$ we have $k+1 = p$ (Corollary 6) and $min_p[d] = i+2$. Since the demonstration of the second invariant does not require the use of the first invariant on $\alpha(next_k[d])$ (which is not valid) the same demonstration as in the case $level(next_{k+1}[d]) \le i$ holds.

□

The propositions seen in this section were mainly devoted to the properties of the objects used by Algorithm 4. The following propositions use these properties to connect the objects of Algorithm 4 with the ones used by Theorems 2 to 5. This section concludes with Proposition 19 which shows that the arrays $First(j, d)$ contain the σ_j successors of the darts d at the end of the algorithm.

Proposition 17 *Using the same hypothesis and notations as Proposition 14, given a dart $d \in \boldsymbol{\mathcal{D}}$:*

1. *If $i \in \mathcal{DS}_d$ and d is removed before level $i+1$ or $state(i) = Removed$:*

$$\forall j \in \{0, \ldots, \mathcal{L}_{\,i}(d) - 1\} \quad First(j, d) = d_{succ_{i,d}(j+1)}$$

2. *If $i \in \mathcal{DS}_{\alpha(d)}$ and d is contracted before level $i+1$ or $state(i) = Contracted$:*

$$\forall j \in \{0, \ldots, \mathcal{L}_{\,i}(\alpha(d)) - 1\} \quad First(j, d) = d_{succ_{i,\alpha(d)}(j+1)}$$

Proof:

Property 1: Since $i \in \mathcal{DS}_d$, $\mathcal{L}_{\,i}(d)$ and the function $succ_{i,d}$ are defined. The function $succ_{i,d}$ is defined by (Definition 8):

$$\left.\begin{array}{ll} \forall d \in & \boldsymbol{\mathcal{D}} \\ \forall j \in & \{1, \ldots, \mathcal{L}_{\,i}(d)\} \end{array}\right\} \quad succ_{i,d}(j) = \min\{k \in \{Ind_i(d)+1, \ldots, p\} \mid level(d_k) \geq j\} \quad (26)$$

with $CDS_i(d^i) = d_1, \ldots, d_p$. This last definition is equivalent to:

$$\left.\begin{array}{l} \forall d \in \boldsymbol{\mathcal{D}} \\ \forall j \in \{0, \ldots, \mathcal{L}_{\,i}(d) - 1\} \end{array}\right\} \quad succ_{i,d}(j+1) = \min\{k \in \{Ind_i(d)+1, \ldots, p\} \mid level(d_k) > j\} \quad (27)$$

We have by Lemma 1, $]d \ldots, First(j, d)[\trianglelefteq S_i(d)$ Moreover,

$$\forall b \in]d \ldots, First(j,d))[\trianglelefteq S_i(d),\ level(b) \le j$$

Therefore, given one $j \in \{0, \ldots, \mathcal{L}\ _i(d)-1\}$, $succ_{i,d}(j+1)$ is defined and $d.\ldots.First(j,d) \preceq_{pre} d.\ldots.d_{succ_{i,d}(j+1)}$ .Therefore, since $d_{succ_{i,d}(j+1)} \in CDS_i(d^i)$, $First(j,d) \in CDS_i(d^i)$. Moreover, using the first property of the second invariant of Lemma 1: $level(First(j,d)) > j$. Thus:

$$d_{succ_{i,d}(j+1)} = First(j,d)$$

Property 2: Since $i \in \mathcal{DS}_{\alpha(d)}$, $\mathcal{L}\ _i(\alpha(d))$ and the function $succ_{i,\alpha(d)}$ are defined. For any $j \in \{0, \ldots, \mathcal{L}\ _i(\alpha(d))-1\}$ the value of $succ_{i,\alpha(d)}(j+1)$ is defined as the index of the first dart in $CDS_i(\alpha(d)^i)$ with a level strictly greater than j. Using Lemma 1:

$$\forall b \in]\alpha(d), \ldots, First(j,d)[\trianglelefteq S_i(\alpha(d)^i),\ level(b) \le j$$

We have thus:

$$\alpha(d).\ldots.First(j,d) \preceq_{pre} \alpha(d).\ldots.d_{succ_{i,,\alpha(d)}(j+1)} \trianglelefteq S_i(\alpha(d)^i)$$

Since $succ_{i,\alpha(d)}(j+1)$ is defined, $d_{succ_{i,,\alpha(d)}(j+1)} \in CDS_i(\alpha(d^i))$ and $First(j,d) \in CDS_i(\alpha(d)^i)$. Moreover, since $level(First(j,d)) > j$:

$$d_{succ_{i,,\alpha(d)}(j+1)} = First(j,d)$$

□

Proposition 18 *Using the same hypothesis and notations as Proposition 14, given a dart $d \in \boldsymbol{\mathcal{D}}$:*

1. *If $i \in \mathcal{DS}_d$ and d is removed before level $i+1$ or $state(i) = Removed$:*

$$\forall j \in \{\mathcal{L}\ _i(d), \ldots, i\} \quad First(j,d) = \begin{cases} \varphi_i(d^i) & \text{If } K_i \text{ is a contraction kernel} \\ \sigma_i(d^i) & \text{If } K_i \text{ is a removal kernel} \end{cases}$$

2. *If $i \in \mathcal{DS}_{\alpha(d)}$ and d is contracted before level $i+1$ or $state(i) = Contracted$:*

$$\forall j \in \{\mathcal{L}\ _i(\alpha(d)), \ldots, i\}\, First(j,d) = \begin{cases} \varphi_i(\alpha(d)^i) & \text{If } K_i \text{ is a contraction kernel} \\ \sigma_i(\alpha(d)^i) & \text{If } K_i \text{ is a removal kernel} \end{cases}$$

Proof:

Property 1: According to the second invariant of Lemma 1 at iteration $k = p$:

$$\forall j \in \{0, \dots, i\} \quad \left\{ \begin{array}{l} level(First(j,d)) > j \text{ and} \\ \forall b \in]d, \dots, First(j,d)[\quad level(b) \leq j \end{array} \right.$$

Given one $j \in \{0, \dots, i\}$, according to Lemma 1, $First(j,d)$ has an index in $S_i(d^i)$ strictly greater than the one of d. Therefore, if Ind denotes the index of each dart in $S_i(d^i)$ we have $Ind(d^i) \leq Ind(d) < Ind(First(j,d))$. Therefore, $First(j,d)$ cannot be equal to d^i. Moreover, according to the definition of $\mathcal{L}_{\ i}(d)$ we have:

$$\forall b \in CDS_i^*(d^i) \quad level(b) \leq \mathcal{L}_{\ i}(d)$$

Therefore, since $S_i(d^i)$ is equal to $CDS_i(d^i)\sigma_i(d^i)$ or $CDS_i(d^i)\varphi_i(d^i)$ according to K_i, the only two darts of $S_i(d^i)$ whose level is greater than $\mathcal{L}_{\ i}(d)$ are d^i and $\varphi_i(d^i)$ or $\sigma_i(d^i)$. Therefore:

$$\forall j \in \{\mathcal{L}_{\ i}(d), \dots, i\} \quad First(j,d) = \left\{ \begin{array}{ll} \varphi_i(d^i) & \text{If } K_i \text{ is a contraction kernel} \\ \sigma_i(d^i) & \text{If } K_i \text{ is a removal kernel} \end{array} \right.$$

Property 2: According to the second invariant of Lemma 1 at iteration $k = p$:

$$\forall j \in \{0, \dots, i\} \quad \left\{ \begin{array}{l} level(First(j,d)) > j \text{ and} \\ \forall b \in]\alpha(d), \dots, First(j,d)[\quad level(b) \leq j \end{array} \right.$$

As previously, $First(j,d)$ cannot be equal to $\alpha(d)^i$ for any $j \in \{0, \dots, i\}$. Moreover, since:

$$\forall b \in CDS_i^*(\alpha(d)^i) \quad level(b) \leq \mathcal{L}_{\ i}(\alpha(d))$$

$First(j,d)$ must be equal to $\varphi_i(\alpha(d)^i)$ or $\sigma_i(\alpha(d)^i)$ according to K_i for any $j \in \{\mathcal{L}_{\ i}(\alpha(d)), \dots, i\}$.

□

Let us consider the state of the variables $next[d]$ and $First(., d)$ during the third iteration of Algorithm 4 displayed in Table 7 respectively in Columns 2 and 3 to 7. For example, the value of $next[-9]$(line 4, column 2) during this iteration is equal to 9 and the array $First$ is initialized up to index 2 with:

$$First(0,-9) = -4,\ First(1,-9) = -8,\ First(2,-9) = -3$$

The value of $min_3[-9]$ is thus equal to $2+2=4$.

The dart -9 is removed at level 2 and belongs to $CDS_4(-11)$. According to Lemma 1, all the darts between -9 and 9 in $CDS_4(-11)$ have a level less than 4(see Figure 16). In the same way, the sequences defined by the array $First(.,-9)$ are equal to:

$$\begin{array}{rcccl}]-9,\ldots,First(0,-9)[& = &]-9,\ldots,-4[& = & \varepsilon \\ ]-9,\ldots,First(1,-9)[& = &]-9,\ldots,-8[& = & (-4) \\ ]-9,\ldots,First(2,-9)[& = &]-9,\ldots,-3[& = & (-4,-8) \end{array}$$

Since $level(-4) = 1$ and $level(-8) = 2$, the second invariant of Algorithm 4 is satisfied.

During the fourth iteration $next[-9]$ is set to $next[9] = 11$ with $min_4[9] = 6$ (the line associated to the dart 9 is completely filled). Since $min_4[9] \geq min_4[-9]$, $min_4[-9]$ is set to $min_4[9] = 6$ and the array $First(.,9)$ is copied in $First(.,-9)$ from the index $min_4[-9]-1 = 3$ to $min_4[9]-2 = 4$ (Table 8, line 4).

The line associated to -9 is thus completely filled during the fourth iteration and the processor associated to this dart stop at this iteration. In the same way, the line associated to -11 (line 2) is filled during the fifth iteration (Table 9) and its array $First(.,-11)$ is equal to:

$$\begin{array}{lll} First(0,-11) = -5, & First(1,-11) = -5, & First(2,-11) = -5 \\ First(3,-11) = -5, & First(4,-11) = 11 & \end{array}$$

Note that we have (Figure 4):

$$\sigma_0(-11) = -5,\ \sigma_1(-11) = -5,\ \sigma_2(-11) = -5,\ \sigma_3(-11) = -5,\ \sigma_4(-11) = 11$$

This equality between $First(j,d)$ and $\sigma_j(d)$ is justified by the following proposition:

Proposition 19 *With the same notations as Proposition 14 the array First initialized by Algorithm 4 satisfies the following property:*

$$\forall d \in \boldsymbol{\mathcal{D}}, \forall j \in \{0,\ldots,\min(level(d)-1,i)\} \quad First(j,d) = \sigma_j(d)$$

Proof:

Let us first consider a dart $d \in \mathcal{SD}_i$. We have thus $level(d) > i$ and $\min(level(d)-1, i) = i$.

Case 1: K_i is a removal kernel ($state(i) = Removed$).

If $CDS_i(d) = d$, then $S_i(d) = d.\sigma_i(d)$, moreover according to Theorem 2:

$$\forall j \in \{0, \dots, i\} \quad \sigma_j(d) = \sigma_i(d)$$

The proof is in this case provided by Proposition 15.

If $CDS_i(d) \neq (d)$, $L_i(d)$ is defined (Definition 4). Moreover, we have in this case (Propositions 12 and 17):

$$\forall j \in \{1, \dots, L_i(d)\} \quad d_{l_{i,d}(j)} = First(j-1, d)$$

Using Theorem 3 we have:

$$\forall j \in \{0, \dots, L_i(d)-1\} \quad \sigma_j(d) = d_{l_{i,d}(j+1)} = First(j, d)$$

Using Proposition 18 we have:

$$\forall j \in \{L_i(d), \dots, i\} \quad First(j, d) = \sigma_i(d)$$

Note that in this case $\mathcal{L}\ _i(d) = L_i(d)$ (Proposition 11) and $d^i = d$.

Using Corollary 2:

$$\forall j \in \{L_i(d), \dots, i\} \quad \sigma_j(d) = \sigma_i(d) = First(j, d)$$

Case 2: K_i is a contraction kernel ($state(i) = Contracted$)

We have in this case $S_i(\alpha(d)) = CDS_i(\alpha(d))\varphi_i(\alpha(d)) = CDS_i(\alpha(d))\sigma_i(d)$. If $CDS_i(\alpha(d)) = \alpha(d)$, we have $S_i(\alpha(d)) = \alpha(d)\sigma_i(d)$. Moreover, according to Theorem 2:

$$\forall j \in \{0, \dots, i\} \quad \varphi_j(\alpha(d)) = \sigma_j(d) = \varphi_i(\alpha(d)) = \sigma_i(d)$$

The proof is in this case provided by Proposition 15.

If $CDS_i(\alpha(d)) \neq (\alpha(d))$, $L_i(\alpha(d))$ and the functions $l_{i,\alpha(d)}$ are defined. Using Proposition 17 we have:

$$\forall j \in \{1, \dots, L_i(\alpha(d))\} \quad d_{l_{i,\alpha(d)}(j)} = First(j-1, d)$$

Using Theorem 3:

$$\forall j \in \{0, \dots, L_i(\alpha(d)) - 1\} \quad \varphi_j(\alpha(d)) = \sigma_j(d) = d_{l_{i,\alpha(d)}(j+1)} = First(j, d)$$

As previously, using Proposition 18 we have:

$$\forall j \in \{L_i(\alpha(d)), \dots, i\} \quad First(j, d) = \sigma_i(d)$$

Using Corollary 2:

$$\forall j \in \{L_i(d), \dots, i\} \quad \varphi_j(\alpha(d)) = \sigma_j(d) = \varphi_i(\alpha(d)) = \sigma_i(d) = First(j, d)$$

Let us now consider a dart d contracted or removed before level $i+1$. The sequence $S_i(d^i)$ is thus equal to $CDS_i(d^i)\sigma_i(d^i)$ if K_i is a removal kernel or $CDS_i(d^i)\varphi_i(d^i)$ if K_i is a contraction kernel. Moreover, since $level(d) \leq i$ we have $min(level(d) - 1, i) = level(d) - 1$.

Case d is removed: If $i \in I_d$, d is the last dart of $CDS_i(d^i)$ and we have by Proposition 8:

$$\forall j \in \{0, \dots, level(d) - 1\} \ \sigma_j(d) = \begin{cases} \varphi_i(d^i) & \text{If } K_i \text{ is a contraction kernel} \\ \sigma_i(d^i) & \text{If } K_i \text{ is a removal kernel} \end{cases}$$

The proof is then provided by Proposition 15.

If $i \notin I_d$, $\mathcal{L}_i(d)$ and the functions $succ_{i,d}$ are defined. Using either Theorem 4 or Theorem 5 according to $CDS_{level(d)-1}(\alpha(d))$ we obtain:

$$\forall j \in \{0, \dots, \mathcal{L}_i(d) - 1\} \quad \sigma_j(d) = d_{succ_{i,d}(j+1)}$$

We have thus by Proposition 17:

$$\forall j \in \{0, \dots, \mathcal{L}_i(d) - 1\} \quad \sigma_j(d) = d_{succ_{i,d}(j+1)} = First(j, d)$$

If $\mathcal{L}_i(d) = level(d)$ nothing remains to be demonstrated. Otherwise $\mathcal{L}_i(d) < level(d)$ and we have (Proposition 13) $i \in J_d$ and $\mathcal{L}_i(d) = L_{level(d)-1}(\alpha(d))$. Note that since $i \in J_d$ and $i \notin I_d$ we must have $CDS^*_{level(d)-1}(\alpha(d)) \neq \varepsilon$ (Definition 6 and TR-63, Proposition 21). Therefore, using Corollary 5:

$$\forall j \in \{\mathcal{L}_i(d), \dots, level(d) - 1\} \ \sigma_j(d) = \begin{cases} \varphi_i(d^i) & \text{If } K_i \text{ is a contraction kernel} \\ \sigma_i(d^i) & \text{If } K_i \text{ is a removal kernel} \end{cases}$$

The proof is then provided by Proposition 18.

Case d is contracted: First note that $\alpha(d)$ is contracted at $level(d)$.

If $i \in I_{\alpha(d)}$, $\alpha(d)$ is the last dart of $CDS_i(\alpha(d)^i)$ and we have by Corollary 8:

$$\begin{aligned}&\forall j \in \{0, \dots, level(d) - 1\}\\ &\varphi_j(\alpha(d)) = \sigma_j(d) = \begin{cases} \varphi_i(\alpha(d)^i) & \text{If } K_i \text{ is a contraction kernel} \\ \sigma_i(\alpha(d)^i) & \text{If } K_i \text{ is a removal kernel}\end{cases}\end{aligned} \tag{28}$$

The proof is then provided by Proposition 15.

If $i \notin I_{\alpha(d)}$, $\mathcal{L}_i(\alpha(d))$ and the functions $succ_{i,\alpha(d)}$ are defined. Using either Theorem 4 or Theorem 5 according to $CDS_{level(d)-1}(\alpha(d))$ we obtain:

$$\forall j \in \{0, \dots, \mathcal{L}_i(\alpha(d)) - 1\} \quad \varphi_j(\alpha(d)) = \sigma_j(d) = d_{succ_{i,\alpha(d)}(j+1)}$$

As previously, we have by Proposition 17:

$$\forall j \in \{0, \dots, \mathcal{L}_i(\alpha(d)) - 1\} \quad \sigma_j(d) = d_{succ_{i,\alpha(d)}(j+1)} = First(j, d)$$

If $\mathcal{L}_i(\alpha(d)) = level(\alpha(d)) = level(d)$ nothing remains to be demonstrated. Otherwise $\mathcal{L}_i(\alpha(d)) < level(d)$ and we have by Corollary 5:

$$\begin{aligned}&\forall j \in \{\mathcal{L}_i(\alpha(d)), \dots, level(d) - 1\}\\ &\varphi_j(\alpha(d)) = \begin{cases} \varphi_i(\alpha(d)^i) & \text{If } K_i \text{ is a contraction kernel} \\ \sigma_i(\alpha(d)^i) & \text{If } K_i \text{ is a removal kernel}\end{cases}\end{aligned} \tag{29}$$

We have thus by Proposition 18:

$$\forall j \in \{\mathcal{L}_i(\alpha(d)), \dots, level(d) - 1\} \; \varphi_j(\alpha(d)) = \sigma_j(d) = First(j, d)$$

□

References

[1] M. Bister, J. Cornelis, and A. Rosenfeld. A critical view of pyramid segmentation algorithms. *Pattern Recognit Letter.*, 11(9):605–617, September 1990.

[2] J. P. Braquelaire, P. Desbarats, J.-P. Domenger, and C. Wüthrich. A topological structuring for aggregates of 3d discrete objects. In Walter Kropatsch and J.-M. Jolion, editors, 2^{nd} *IAPR-TC-15 Workshop on Graph-based Representations*, volume 126, pages 145–154, Haindorf, Austria, May 1999. Österreichische Computer Gesellschaft.

[3] Jean Pierre Braquelaire and Luc Brun. Image segmentation with topological maps and inter-pixel representation. *Journal of Visual Communication and Image representation*, 9(1):62–79, 1998.

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

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

[6] L. Brun and Walter. Kropatsch. Dual contraction of combinatorial maps. In Walter Kropatsch and J.-M. Jolion, editors, 2^{nd} *IAPR-TC-15 Workshop on Graph-based Representations*, volume 126, pages 145–154, Haindorf, Austria, May 1999. Österreichische Computer Gesellschaft.

[7] L. Brun and Walter Kropatsch. The construction of pyramids with combinatorial maps. Technical Report 63, Institute of Computer Aided Design, Vienna University of Technology, lstr. 3/1832,A-1040 Vienna AUSTRIA, June 2000.

[8] L. Brun and Walter Kropatsch. Irregular pyramids with combinatorial maps. In Adnan Amin, Francesc J. Ferri, Pavel Pudil, and Francesc J. Iñesta, editors, *Advances in Pattern Recognition, Joint IAPR International Workshops SSPR'2000 and SPR'2000*, volume Vol. 1451 of *Lecture Notes in Computer Science*, pages 256–265, Alicante, Spain, August 2000. Springer, Berlin Heidelberg, New York.

[9] Luc Brun, Jean Philipe Domenger, and Jean Pierre Braquelaire. Discrete maps : a framework for region segmentation algorithms. In *Workshop on Graph based representations*, Lyon, April 1997. published in Advances in Computing (Springer).

[10] Luc Brun 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.

[11] Luc Brun and Walter Kropatsch. Pyramids with combinatorial maps. Technical Report PRIP-TR-057, PRIP, TU Wien, 1999.

[12] Peter Burt, Tsai-Hong Hong, and Azriel Rosenfeld. Segmentation and estimation of image region properties through cooperative hierarchial computation. *IEEE Transactions on Sustems, Man and Cybernetics*, 11(12):802–809, December 1981.

[13] H. Elter and P. Lienhardt. Extension of the notion of map for the representation of the topology of cellular complexes. In *Proc. 4th Canad. Conf. Comput. Geom.*, pages 65–70, 1992.

[14] A. Jones Gareth and David Singerman. Theory of maps on orientable surfaces. *Proceedings of the London Mathematical Society*, 3(37):273–307, 1978.

[15] Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, and Werner Stuetzle. Mesh optimization. In James T. Kajiya, editor, *Computer Graphics (SIGGRAPH'93 Proceedings), volume 27*, pages 19–26, August 1993.

[16] Walter G. Kropatsch. Building Irregular Pyramids by Dual Graph Contraction. *IEE-Proc. Vision, Image and Signal Processing*, Vol. 142(No. 6):pp. 366–374, December 1995.

[17] Walter G. Kropatsch. Property Preserving Hierarchical Graph Transformations. In Carlo Arcelli, Luigi P. Cordella, and Gabriella Sanniti di Baja, editors, *Advances in Visual Form Analysis*, pages 340–349. World Scientific Publishing Company, 1998.

[18] Walter G. Kropatsch and Souheil BenYacoub. Universal Segmentation with P*IRR*amids. In Axel Pinz, editor, *Pattern Recognition 1996, Proc. of 20th ÖAGM Workshop*, pages 171–182. OCG-Schriftenreihe, Österr. Arbeitsgemeinschaft für Mustererkennung, R. Oldenburg, 1996. Band 90.

[19] Walter G. Kropatsch and Mark Burge. Minimizing the Topological Structure of Line Images. In Adnan Amin, Dov Dori, Pavel Pudil, and Herbert Freeman, editors, *Advances in Pattern Recognition, Joint IAPR International Workshops SSPR'98 and SPR'98*, volume Vol. 1451 of *Lecture Notes in Computer Science*, pages 149–158, Sydney, Australia, August 1998. Springer, Berlin Heidelberg, New York.

[20] Walter G. Kropatsch and Herwig Macho. Finding the structure of connected components using dual irregular pyramids. In *Cinquième Colloque DGCI*, pages 147–158. LLAIC1, Université d'Auvergne, ISBN 2-87663-040-0, September 1995.

[21] Pascal 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.

[22] Annick Montanvert, Peter Meer, and Azriel Rosenfeld. Hierarchical image analysis using irregular tessellations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(4):307–316, APRIL 1991.

[23] Jean-Gerard Pailloncy, 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.

[24] Azriel Rosenfeld, editor. *Multiresolution Image Processing and Analysis.* Springer Verlag, Berlin, 1984.

[25] W.T. Tutte. A census of planar maps. Canad.J.Math., 15:249–271, 1963.

[26] Dieter Willersinn and Walter G. Kropatsch. Dual graph contraction for irregular pyramids. In *International Conference on Pattern Recogntion D: Parallel Computing*, pages 251–256, Jerusalem, Israel, 1994. International Association for Pattern Recognition.

9 Index of Definitions and Notations

- Darts :

$\boldsymbol{\mathcal{D}}$	:	initial set of darts,
$\mathcal{SD}_i$	:	set of surviving darts at level i,
α	:	involution encoding the edges of the combinatorial maps. Remains constant along the pyramid in this technical report,
σ_i	:	permutation σ encoding the vertices of the combinatorial map defined at level i,
φ_i	:	permutation φ encoding the faces of the combinatorial map defined at level i,
d^i	:	surviving dart at level i whose connecting dart sequence contains d (section 4).

- Functions

$L_i(d)$	:	greatest level contained in $CDS^*_i(d)$ (Definition 4, page 19),
$l_{i,d}(j)$	:	index of the first dart in $CDS_i(d)$ with a level greater or equal to j (Definition 5, page 23),
$Ind_i(d)$	:	index d in the connecting dart sequence which contains it at level i (Definition 8, page 43),
$\mathcal{L}_i(d)$	:	minimum between $level(d)$ and the greater level of the darts which follow d in $CDS_i(d^i)$ (Definition 8, equation 14, page 43),
$succ_{i,d}(j)$	:	index of the first dart with a level greater than j which follows d in $CDS_i(d^i)$ (Definition 8, equation 15, page 43).

- Kernels :

Alternate sequence of kernels: sequence of kernels such that two successive kernels does not have the same type (Definition 1, page 9),

Equivalent sequence of kernels: conditions under which two sequence of kernels produce a same combinatorial pyramid (Definition 2, page 9).

- Levels :

J_d	:	set of levels included in $\{level(d), \ldots, n\}$ such that $dCDS^*_{level(d)-1}(\alpha(d))$ is a suffix of $CDS^*_i(d^i)$ (Definition 6, page 36),
I_d	:	set of levels such that d is the last dart of the connecting dart sequence which contains it (TR-63 Proposition 21),
$\mathcal{DS}_d$	:	set of levels such that d is neither the last nor the only dart of $CDS_i(d^i)$ (Definition 7, page 42).

- Sequences of darts

$CDS_i(d)$	:	connecting dart sequence of d defined at level i (TR-63 [7] Definition 11, and this report pages 8 and 15),
$CDS^*_i(d)$	:	connecting dart sequence of d at level i without its first dart d,
$CW_i(d)$	:	connecting walk of d at level i (TR-57 [11]),
$\trianglelefteq$	:	sub-word relationship between two sequences of darts (Definition 3, page 3),
$\triangleleft$	:	strict sub-word relationship between two sequences of darts (Definition 3, page 3),
$\preceq_{pre}$	:	prefix relationship between two sequences of darts (Definition 3, page 3),
$\prec_{pre}$	:	strict prefix relationship between two sequences of darts (Definition 3, page 3),
$\preceq_{suff}$	:	suffix relationship between two sequences of darts (Definition 3, page 3),
$\prec_{suff}$	:	strict suffix relationship between two sequences of darts (Definition 3, page 3),
ε	:	empty sequence of darts (Definition 3, page 3),
$\vert S\vert$	:	number of darts of the sequence S.