On defining linear orders by automata

Irène Durand, Bruno Courcelle, Michael Raskin

On defining linear orders by automata

Irne Durand[1], Bruno Courcelle[1], and Michael Raskin[2]

[1]LaBRI, Bordeaux University and CNRS
[2]Technical University Munich

November 26, 2019

Abstract

We define linear orders $\leq_Z$ on product sets $Z := X_1 \times X_2 \times ... \times X_n$ and on subsets Z of $X_1 \times X_2$ where each composing set X_i is $[0, p]$ or $\mathbb{N}$, and ordered in the natural way. We require that $(Z, \leq_Z)$ be isomorphic to $(\mathbb{N}, \leq)$ if it is infinite. We want linear orderings of Z such that, in two consecutive tuples $\boldsymbol{z} = (z_1, ..., z_n)$ and $\boldsymbol{z}' = (z'_1, ..., z'_n)$, we have $|z_i - z'_i| \leq 1$ for each i. Furthermore, we define their *distance* $d(\boldsymbol{z}, \boldsymbol{z}')$ as the number of indices i such that $z_i \neq z'_i$. We will consider orderings where the distance of two consecutive tuples is at most 2. We are interested in algorithms that determine the tuple in Z following $\boldsymbol{z}$ by using local information, where "local" is meant with respect to graphs associated with Z, and that work as well for finite and infinite components X_i, without knowing whether the components X_i are finite or not. We will formalize these algorithms by *deterministic graph-walking automata*.

Introduction

Motivated by enumeration problems[1] (cf. [2]), we define linear orders $\leq_Z$ on product sets $Z := X_1 \times X_2 \times ... \times X_n$ and on subsets Z of $X_1 \times X_2$ where each composing set X_i is linearly ordered with order type ω, that of $\mathbb{N}$, if it is infinite. We require that $(Z, \leq_Z)$ be isomorphic to $(\mathbb{N}, \leq)$ if it is infinite. Otherwise, it is isomorphic to $([0, |Z| - 1], \leq)$. Our orders extend the classical diagonal enumeration of $\mathbb{N} \times \mathbb{N}$ establishing a bijection of this set with $\mathbb{N}$.

Each ordered set X_i will be taken equal to $[0, p]$ or $\mathbb{N}$, and ordered in the natural way. We want linear orderings of Z such that, in two consecutive tuples $\boldsymbol{z} = (z_1, ..., z_n)$ and $\boldsymbol{z}' = (z'_1, ..., z'_n)$, we have $|z_i - z'_i| \leq 1$ for each i. Furthermore, we define their *distance* $d(\boldsymbol{z}, \boldsymbol{z}')$ as the number of indices i such that $z_i \neq z'_i$. We have a *dk-ordering* if this distance is always at most k. We will only consider d1- and d2-orderings.

[1]Enumeration is taken in the sense of "listing" not in that of "counting", as in enumerative combinatorics.

These requirements can be expressed in terms of graphs. If $Z \subseteq X_1 \times X_2 \times ... \times X_n$, we define two associated graphs (kinds of "hypercubes"):

> $G_1(Z)$ has vertex set Z and an edge between $\boldsymbol{z} = (z_1, ..., z_n)$ and $\boldsymbol{z'} = (z'_1, ..., z'_n)$ if and only if $|z_i - z'_i| \leq 1$ for each i and $d(\boldsymbol{z}, \boldsymbol{z'}) = 1$, and
>
> $G_2(Z)$ is similar with an edge between $\boldsymbol{z}$ and $\boldsymbol{z'}$ if and only if $|z_i - z'_i| \leq 1$ for each i and $d(\boldsymbol{z}, \boldsymbol{z'})$ is 1 or 2.

Hence, $G_1(X_1 \times X_2)$ is a planar rectangular grid and $G_2(X_1 \times X_2)$ is $G_1(X_1 \times X_2)$ augmented with diagonal edges in each square. A d1-ordering (resp. a d2-ordering) of $Z \subseteq X_1 \times X_2 \times ... \times X_n$ is a Hamiltonian path in $G_1(Z)$ (resp. in $G_2(Z)$) starting at (0,0,...,0).

We are interested in algorithms that determine the tuple in Z following $\boldsymbol{z}$ by using local information ("local" is meant with respect to $G_1(Z)$ or $G_2(Z)$), and work as well for finite and infinite components X_i. They will formalize these algorithms by means of *deterministic graph-walking automata*, whose runs define *walks* (a walk is like a path, but vertices can be visited several times). We will actually use such automata that only define paths, but the general definition cannot guarantee that an automaton defines a path rather than a walk. These automata traverse graphs equipped with an edge labelling where adjacent edges have different labels[2] among a fixed finite set, which defines a bound on the degree. They may have infinitely many *states*. At a vertex reached by a walk, the automaton determines the color of the next edge to be traversed from the state and some knowledge of a finite neighbourhood, for example the set of colors of the incident edges. However, this neighbourhood can be a ball of radius $r > 1$ (and thus of bounded size) in a more complex model. After the traversal via the next edge, the state may be changed, according to the used transition rule.

We will not develop a general theory of such automata (see [3] for graph-walking automata considered in relation with logic, or [4]), but we will define automata well-adapted to the graphs $G_1(Z)$ and $G_2(Z)$. One of our main theorems is the following one, informally stated.

Theorem 1. *For each n, there is an automaton with 2^{n-1} states that defines a d2-ordering respecting levels, on any set $Z = X_1 \times X_2 \times ... \times X_n$ such that each X_i is $\mathbb{N}$ or $[0, p]$ for some p.*

The *height* of a tuple of integers is the sum of values of its components. A level is the set of tuples of the same height. For d2-orderings, we want that levels be traversed consecutively by increasing order of height, we call them *d2-ℓ-orderings*. Related positive and negative results will be as follows:

Theorem 2. *There is no finite or infinite automaton that defines a d1-ordering in each set $X_1 \times X_2$ such that each X_i is $\mathbb{N}$ or $[0, p]$ for some p by looking at distance 1 of the current vertex. There is a finite one, that looks at distance 2.*

[2] These labels are related with edge directions.

We also characterize the affine and convex subsets of $\mathbb{N} \times \mathbb{N}$ having d2-ℓ-orderings defined by finite automata.

Acknowledgements: This research was supported by the GraphEn project of Agence Nationale pour la recherche and the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme, under grant agreement No 787367 (PaVeS).

1 Definitions and first results

We will order linearly sets $Z := X_1 \times X_2 \times ... \times X_n$ and subsets of $X_1 \times X_2$ where each composing set X_i is linearly ordered with order type ω, that of $\mathbb{N}$, in the case it is infinite. We require that $(Z, \leq_Z)$ be isomorphic to $(\mathbb{N}, \leq)$ if it is infinite.

Each ordered set X_i will be taken equal to $[0, p]$ or $\mathbb{N}$, and ordered in the natural way.

We want linear orderings of Z such that, in two consecutive tuples $\boldsymbol{z} = (z_1, ..., z_n)$ and $\boldsymbol{z'} = (z'_1, ..., z'_n)$, we have $|z_i - z'_i| \leq 1$ for each i.

Definition 1.1. *Distances, heights and levels.*

(a) The *distance* $d(\boldsymbol{z}, \boldsymbol{z'})$ of $\boldsymbol{z} = (z_1, ..., z_n)$ and $\boldsymbol{z'} = (z'_1, ..., z'_n)$ is the number of indices i such that $z_i \neq z'_i$.

(b) In a *dk-ordering*, the distance between any two consecutive tuples is at most k. We will actually consider d1- and d2-orderings.

(c) If $Z \subseteq X_1 \times X_2 \times ... \times X_n$, we define two graphs:

> $G_1(Z)$ has vertex set Z and an edge between $\boldsymbol{z} = (z_1, ..., z_n)$ and $\boldsymbol{z'} = (z'_1, ..., z'_n)$ if and only if $|z_i - z'_i| \leq 1$ for each i and $d(\boldsymbol{z}, \boldsymbol{z'}) = 1$, and
>
> $G_2(Z)$ is similar with an edge between $\boldsymbol{z}$ and $\boldsymbol{z'}$ if and only if $|z_i - z'_i| \leq 1$ for each i and $d(\boldsymbol{z}, \boldsymbol{z'})$ is 1 or 2.

Hence, a di-ordering of $Z \subseteq X_1 \times X_2 \times ... \times X_n$ where i is 1 or 2 is a Hamiltonian path in $G_i(Z)$ starting at (0,0,...,0).

(d) The *height* of a tuple of integers is the sum of the values of its components. The *level* k of $Z \subseteq X_1 \times X_2 \times ... \times X_n$ is the set of its tuples of height k.

(e) A d2-ℓ-ordering of Z, is d2-ℓ-ordering such that the levels are traversed consecutively by increasing order of height. □

We now review the classical diagonal enumeration of $\mathbb{N} \times \mathbb{N}$ and its extension to certain subsets of the form $X \times Y$ in order to present on an easy case our notion of *graph-walking automaton*.

Definition 1.2. *The diagonal d2-ℓ-ordering $\leq_\Delta$ of $\mathbb{N} \times \mathbb{N}$*

We define the *type* $\tau(i, j)$ of a pair $(i, j) \in \mathbb{N} \times \mathbb{N}$ as the pair, also in $\mathbb{N} \times \mathbb{N}$:

$$\tau(i, j) := \text{IF } i + j \text{ is even THEN } (i + j, i) \text{ ELSE } (i + j, j).$$

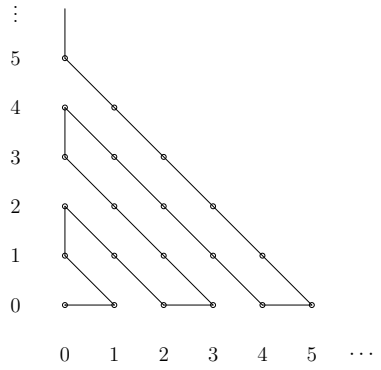

Figure 1: A d2-ℓ-ordering of $\mathbb{N} \times \mathbb{N}$.

Note that (i, j) can be recovered from $\tau(i, j)$: $\tau^{-1}(m, n) =$IF m is even THEN $(n, m - n)$ ELSE $(m - n, n)$. The pairs $(i, j) \in \mathbb{N} \times \mathbb{N}$ are ordered by increasing lexicographic order of their types $\tau(i, j)$. That is:

$$(i, j) \leq_\Delta (i', j') \text{ if and only if } \tau(i, j) \leq_{lex} \tau(i', j'),$$

We obtain a d2-ℓ-ordering of $\mathbb{N}\times\mathbb{N}$. The corresponding ordered set is denoted by $\mathbb{N}\Delta\mathbb{N}$. Its level k is the interval of pairs (i, j) such that $i + j = k$. It begins with /00/10,01/02,11,20/30,21,12,03/04,... where we separate levels by /. Odd levels are traversed in reverse lexicographic order.

The pair $\texttt{next}(i, j)$ that follows (i, j) in this order is computed as follows (we use $\wedge$ as logical "and"):

$$\begin{aligned}\texttt{next}(i, j) =& \text{IF } i + j \text{ is even} \wedge j > 0 \text{ THEN } (i + 1, j - 1) \text{ ELSE}\\ &\text{IF } i + j \text{ is even} \wedge j = 0 \text{ THEN } (i + 1, j) \text{ ELSE}\\ &\text{IF } i + j \text{ is odd} \wedge i > 0 \text{ THEN } (i - 1, j + 1) \text{ ELSE}\\ &\text{IF } i + j \text{ is odd} \wedge i = 0 \text{ THEN } (i, j + 1) \text{ END}\end{aligned}$$

Remark 1.1. In terms of automata (formally defined below, see Table 1), the property "$i + j$ is even" is handled as a *state* that we call `Down`, and similarly, "$i + j$ is odd" is a state called `Up`. (In Figure 1, the vertices 02,11,20 of height 2 are ordered "downwards"). The Boolean values of the tests "$i = 0$" and "$j = 0$" describe the four possible positions of (i, j) with respect to the borders of $G_2(\mathbb{N} \times \mathbb{N})$ represented in the plane. The condition "$i = 0$" characterizes the West border and "$j = 0$" characterizes the South border. There are no North and East borders because we consider $\mathbb{N} \times \mathbb{N}$.

The first clause can be formalized by an automaton transition of the form (that does not change the state):

(`Down`, "not on the South border") $\rightarrow$ ("move to South-East", `Down`)

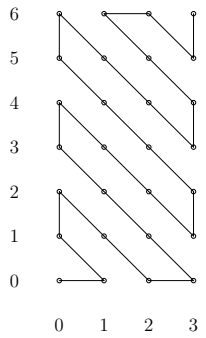

Figure 2: The diagonal enumeration of $Z^{(3,6)}$.

In Figure 1, an example is the edge (1,1)→(2,0) in the path that orders $\mathbb{N} \times \mathbb{N}$. Other examples of edges defined from this transition are (0,4)→(1,3) and (2,2)→(3,1). As $i + j = (i + 1) + (j - 1)$ in this transition, the level is the same for (i, j) and $\texttt{next}(i, j)$, whence the state, defined from the arithmetic parity of the height is not changed. This is so because we are not on the South border.

The last clause yields the edge (0,3)→(0,4) derived from the transition (that changes the state `Up` into `Down`):

(`Up`, "on the West border") → ("move to North", `Down`).

Definition 1.3. *d2-ℓ-orderings of* $X \times Y \subseteq \mathbb{N} \times \mathbb{N}$.

We first extend the algorithm of Definition 1.2 so that it defines a d2-ℓ-ordering of $X \times Y$ when X and/or Y is finite. The order is defined from types $\tau(i, j)$ as above. The corresponding Hamiltonian path in $G_2(Z^{(3,6)})$ where $Z^{(3,6)} := [0, 3] \times [0, 6]$ is illustrated on Figure 2. If X is finite, its maximum is denoted by $\max(X)$.

The information about neighbourhood also uses the Boolean tests $i = \max_X$ and $j = \max_Y$ that are always false if X or, respectively Y, is infinite.

The pair $\texttt{next}(i, j)$ is undefined if (i, j) is the last element, and then the "value" is the message "`none`". We have:

$\texttt{next}(i, j) =$

IF $i = \max_X \wedge j = \max_Y$ THEN `none` ELSE
IF $i + j$ is even $\wedge\ j \neq 0 \wedge i \neq \max_X$ THEN $(i + 1, j - 1)$ ELSE
IF $i + j$ is even $\wedge\ i = \max_X$ THEN $(i, j + 1)$ ELSE
IF $i + j$ is even $\wedge\ j = 0 \wedge i \neq \max_X$ THEN $(i + 1, j)$ ELSE
IF $i + j$ is odd $\wedge\ i \neq 0 \wedge j \neq \max_Y$ THEN $(i - 1, j + 1)$ ELSE
IF $i + j$ is odd $\wedge\ j = \max_Y$ THEN $(i + 1, j)$ ELSE
IF $i + j$ is odd $\wedge\ i = 0 \wedge j \neq \max_Y$ THEN $(i, j + 1)$ END

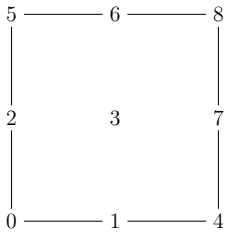

Figure 3: The different types of border used by $\mathcal{B}$.

State	Position	Action	Next state
`Down`	0,1	`E`	`Up`
	2,3,5,6	`SE`	`Down`
	4,7	`N`	`Up`
`Up`	1,3,4,7	`NW`	`Up`
	2	`N`	`Down`
	5,6	`E`	`Up`
Up or `Down`	8		`End`

Table 1: Automaton $\mathcal{B}$

Here is an alternative formalization by an automaton $\mathcal{B}$ that we will use again. There are nine possible types of position of a vertex (i, j) on the grid when X and Y are not singletons[3]:

origin (numbered 0), on the South border (numbered 1),

on the West border (2), in the middle (3), at the South-East corner (4),

at the North-West corner (5), on the North border (6),

on the East border (7) and at the North-East corner (numbered 8).

See Figure 3. Each type can be determined by a combination of Boolean conditions such as "$j = 0$" and "$i \neq \max_X$". In Table 1, to define $\mathcal{B}$ in a readable way, we use the digits 0 to 8 to indicate the types of positions instead of combinations of Boolean conditions. Position "2,3,5,6" means of type 2 or 3 or 5 or 6. The initial state is `Down`. The final state is `End`.

The complete description of $\mathcal{B}$ by a table should include the special cases where X and/or Y is singleton, but we want to keep the table readable. The transitions of the second and fifth lines are described in Remark 1.1.

Proposition 1.1. *The automaton $\mathcal{B}$ defines a d2-ℓ-ordering.*

[3]If X is singleton and Y is not, then 0 and 4 coincide, and so do 2 and 7 and 5 and 8.

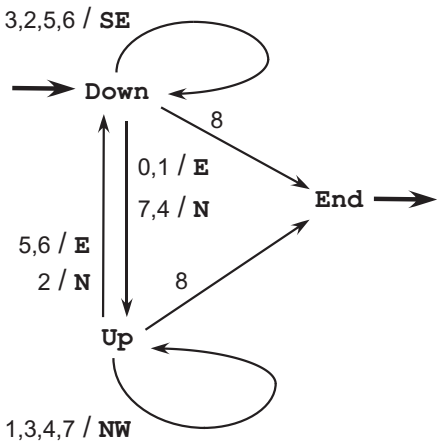

Figure 4: The automaton $\mathcal{B}$ of Proposition 1.1.

Remarks 1.1. (1) The description in Table 1 is appropriate if X and Y are not singletons. However, the definition of `next` works well in all cases.

If X or Y is singleton, there is a unique d2-ℓ-ordering. Otherwise, there are exactly two, one starting by (0,0)→(1,0) (as in Figures 1 and 2) and the other by (0,0)→(0,1). The latter one is obtained by taking `Up` as initial state and adding to Table 1 the transition from the origin (defined by $i = 0 \wedge j = 0$) to state `Down` with a move to the North. The obtained path starts with /00/01,10/20,11,02/03,... . We will denote by $\mathcal{B}^{\#}$ this modified automaton. In automataon $\mathcal{B}$, the state `Up` corresponds to the odd levels and `Down` to the even ones. For $\mathcal{B}^{\#}$, `Down` corresponds to the odd levels and `Up` to the even ones.

(2) The automaton $\mathcal{B}$ (as defined by `next`, cf. Definition 1.3) also works in the special case where $Y = \{0\}$. All positions satisfy $j = 0 \wedge j = \max_Y$. The transitions used are:

IF $i + j$ is even $\wedge\ j = 0 \wedge i \neq \max_X$ THEN $(i + 1, j)$,

and

IF $i + j$ is odd $\wedge\ i \neq \max_X \wedge j = \max_Y$ THEN $(i + 1, j)$

Its works also in the special case where $X = \{0\}$. All positions satisfy $i = 0 \wedge i = \max_X$ .The transitions used are:

IF $i + j$ is even $\wedge\ i = \max_X$ THEN $(i, j + 1)$

and

IF $i + j$ is odd $\wedge\ i = 0 \wedge j \neq \max_Y$ THEN $(i, j + 1)$. □

We want to formalize the notion of an automaton that defines paths in the graphs $G_2(X \times Y)$ or in some of their subgraphs, in particular $G_1(X \times Y)$. This formalization instantiates the general notion sketched in the introduction. In Section 3, we will define automata that define Hamiltonian paths in $G_2(X_1 \times ... \times X_n)$.

Definition 1.4. *Graph-walking automata in 2-dimensional grids.*

(a) The set of *directions* is $\mathcal{D} := \{\texttt{N,E,S,W,NE,SE,SW,NW}\}$, standing for North, East, etc. These definitions refer to a layout of $G_2(X \times Y)$ as in Figure 1. Each direction can be formally defined as a pair of integers: $\texttt{N}:= (0,1)$,$\texttt{E}:= (1,0)$,$\texttt{S}:= (0,-1)$,$\texttt{NE}:= (1,1)$,$\texttt{SE}:= (-1,1)$, etc.

(b) If $(i,j) \in \mathbb{N} \times \mathbb{N}$, and $d \in \mathcal{D}$, then $(i,j)^d \in \mathbb{N} \times \mathbb{N}$ is defined as $(i,j) + d$ (using vectorial addition). Hence, we have, for example, $(i,j)^{\texttt{E}} = (i+1,j)$ and $(i,j)^{\texttt{SE}} = (i+1,j-1)$.

(c) Let G be a subgraph of $G_2(X \times Y)$. The *directions around* a vertex (i,j) are those $d \in \mathcal{D}$ such that $(i,j)^d$ is adjacent to (i,j) in G. We denote this set by $\mathcal{D}_G(i,j)$. It describes the *neighbourhood* of (i,j) in G.

(d) A *graph-walking automaton (of dimension 2)* is a tuple $\mathcal{A} = (Q, \mathcal{T}, q_{init})$ where Q is the finite or countable set of *states*, $q_{init} \in Q$ and $\mathcal{T}$ is the set of *transitions*: they are of the form $(q,\delta) \to (d,q')$ or $(q,\delta) \to \texttt{End}$ where $q, q' \in Q$, $\delta \subseteq \mathcal{D}$ and $d \in \delta$. (The direction d is chosen by the transition in the set δ of possible ones.) From the final state `End`, no transition is possible.

An automaton is *deterministic*: each pair (q,δ) determines a single transition. If Q is infinite, we assume that it is effectively given, and that (d,q') (or `End`) such that $(q,\delta) \to (d,q')$ (or $(q,\delta) \to \texttt{End}$) is computable.

(e) The walk $\pi_{\mathcal{A}}(G,a)$ in G, defined by $\mathcal{A}$ and that starts from a is:

$$a = b_0 \to b_1 \to b_2 \to ... \to b_n \to ...$$

defined with the help of the sequence of states:

$$q_{init} = q_0 \to q_1 \to q_2 \to ... \to q_n \to ...$$

such that, for each $n \geq 0$, $(q_n, \mathcal{D}_G(b_n)) \to (d, q_{n+1})$ and $b_{n+1} := (b_n)^d$. Informally, the state q_n at b_n and its neighbourhood $\mathcal{D}_G(b_n)$ determine in a unique way a direction d such that $(b_n)^d$ is defined and adjacent to b_n, giving b_{n+1}, and the state q_n is updated into q_{n+1}.

We will use graph-walking automata, simply called *automata*, that define paths rather than walks.

We will sometimes restrict the directions to consider. For defining a d2-ℓ-ordering (cf. Definition 1.3 and Proposition 1.1), we only consider the directions `N,E,SE,NW` forming the set $\mathcal{D}_2$. We say that $\mathcal{B}$ is a $\mathcal{D}_2$-automaton. In order to define a d1-ordering, we will only consider directions `N,E,S,W` forming the set $\mathcal{D}_1$. However, an extended notion of direction, including `EE` and `NN`, which means that two steps respectively to East and North are possible, will be used in Theorem 2.2.

In a concrete implementation, we assume that an oracle (a program) can determine membership in Z of any pair b and the set $\mathcal{D}_G(b)$. (A set Z can be defined by affine conditions, such as $i \leq 3j+5 \wedge j \leq -10i+30$. See Section 3.3).

2 D1-orderings on sets $X_1 \times X_2 \times ... \times X_n$.

Proposition 2.1. *From a d1-ordering of a finite set $Y \subseteq X_1 \times X_2 \times ... \times X_n$, one can define a d1-ordering of $Z := \mathbb{N} \times Y$.*

Proof. We discuss d1-orderings as Hamiltonian paths in the graphs $G_1(Y)$ and $G_1(Z)$. Let $P_{a,b}$ from $a = (0, 0, \ldots, 0)$ to some vertex b be a Hamiltonian path in $G_1(Y)$. The opposite path is $P_{b,a}$ from b to a. For each $i \in \mathbb{N}$, let $i \odot P_{a,b}$ be the path $(i, a) \to (i, c_1) \to (i, c_2) \to ... \to (i, b)$ where $P_{a,b}$ is $a \to c_1 \to c_2 \to ... \to b$. Then, one gets in Z the infinite Hamiltonian path $0 \odot P_{a,b} \to 1 \odot P_{b,a} \to 2 \odot P_{a,b} \to 3 \odot P_{b,a} \to ...$ starting from $(0, ..., 0) = (0, a) \in Z$. (The arrow represents the concatenation of paths). □

Remark 2.1. This construction is related to that of Gray codes, cf. [5]. The 3-ary Gray code with 3 digits is the sequence of 3-tuples in $\{0, 1, 2\} \times (\{0, 1, 2\} \times \{0, 1, 2\})$ that reads:

$$000, 001, 002, 012, 011, 010, 020, 021, 022,$$
$$122, 121, 120, 110, 111, 112, 102, 101, 100,$$
$$200, 201, 202, 212, 211, 210, 220, 221, 222.$$

It is thus of the form $0 \odot P \to 1 \odot P' \to 2 \odot P$ where P is:

$$00 \to 01 \to 02 \to 12 \to 11 \to 10 \to 20 \to 21 \to 22,$$

and P' is the opposite path.□

Proposition 2.1 does not apply to $Z := \mathbb{N} \times \mathbb{N}$, and an ordering "row by row" is obviously not adequate as its order type will be $\omega + \omega + ... = \omega \cdot \omega \neq \omega$. This is a motivation for using the diagonal d2-ℓ-ordering of Definition 1.2.

However, d1-orderings can also be defined.

Proposition 2.2. *(1) There is a d1-ordering on $\mathbb{N} \times \mathbb{N}$ definable by an infinite $\mathcal{D}_1$-automaton.*

(2) Each set $Z = X_1 \times X_2 \times ... \times X_p$ where each X_i is finite or infinite, has a d1-ordering.

Proof. (1) See Figures 5 and 6. Theorem 2.2. will establish a more general result for $G_1(X \times Y)$ where X and/or Y may be finite.

(2) We first consider $\mathbb{N}^p$. We use an induction on p. For $p = 2$, the result holds by Assertion (1). Assume we have a d1-ordering $\leq_p$ of $\mathbb{N}^p$. Since, $(\mathbb{N}^p, \leq_p)$ is isomorphic to $(\mathbb{N}, \leq)$, we have by (1) an ordering of $\mathbb{N}^{p+1} = \mathbb{N} \times (\mathbb{N}^p)$. In this order, a step from a vertex to the next one, either modifies the first component

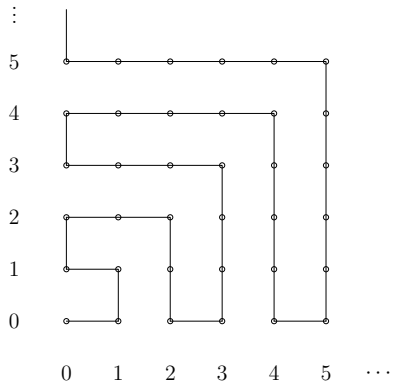

Figure 5: A d1-ordering of $\mathbb{N} \times \mathbb{N}$.

(in $\mathbb{N}$) or the second one, in $\mathbb{N}^p$. In the latter case, only one component of $\mathbb{N}^p$ is modified, as $\leq_p$ is a d1-ordering. In both cases, this step modifies a single component of $\mathbb{N}^{p+1}$. Hence, we have a d1-ordering.

If Z is finite, Proposition 2.1 gives the answer. Otherwise, one can permute the components and write $Z = \mathbb{N} \times ... \times \mathbb{N} \times X_q \times ... \times X_p$ with $X_q, ..., X_p$ finite, hence Z is isomorphic to $\mathbb{N}^q \times (X_q \times ... \times X_p)$ hence to $\mathbb{N} \times Y$ with Y finite, and Proposition 2.1 gives the answer.

The computation of the vertex following any $\boldsymbol{z}$ in Z is computable as all definitions and proofs are effective. Hence, there exists a $\mathcal{D}_1$-automaton, with infinitely many states[4]. □

We now examine whether automata can define d1-orderings. Figure 6 shows a d1-ordering of $X \times Y$ where Y is finite of odd cardinality, and X is finite or infinite, that is defined by an infinite $\mathcal{D}_1$-automaton. Whether X and/or Y is finite need not be known at the beginning, but is determined at some point of the computation. This automaton is easy to define with states including counters[5].

More generally, there are $\mathcal{D}_1$-automata that construct d1-orderings of $X \times Y$ by using some information about X and/or Y. This information can be:

(1) X is finite,

(2) Y is finite,

(3) X is either infinite or finite of odd cardinality,

(4) Y is either infinite or finite of odd cardinality,

(5) X is either infinite or finite of even cardinality,

[4] As in fly-automata, cf. [1], we allow countable sets of states but transitions must be computable.

[5] For defining the path of Figure 5, one can use a finite $\mathcal{D}_1$-automaton that can test whether the current vertex is on the South-West-North-East diagonal.

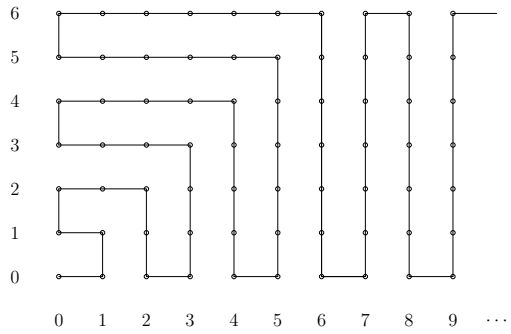

Figure 6: A d1-ordering for [0,2n]×ℕ

(6) Y is either infinite or finite of even cardinality.

The automata for Cases (1) and (2) are finite. In Cases (1), (3), (5), Y may be of any type. In the others, X may be of any type. In Cases (3) and (5), the automaton need not know whether X is infinite or not, and similarly for Y in Cases (4) and (6). Without such information, no deterministic automaton can work correctly, as we prove now.

Theorem 2.1. *There is no (finite or infinite) $\mathcal{D}_1$-automaton that constructs a d1-ordering of $X \times Y$ for arbitrary (linearly ordered) sets X and Y.*

Proof. For getting a contradiction, we assume the existence of an automaton $\mathcal{A} = (Q, \mathcal{T}, q_{init})$ that finds a Hamiltonian path starting at (0,0) in $G_1(X \times Y)$ for any linearly ordered sets X, Y, either $\mathbb{N}$ or $[0, p]$. This automaton uses only the directions `N,E,S,W`. Sets X and Y are finite or infinite, which the automaton "does not know", which means that $\mathcal{A}$ works in all cases. The set of states may be infinite, but determinism will yield a contradiction.

The neighbourhood $\mathcal{D}_G(x)$ describes the following possible positions of a vertex x, numbered 0, 1, 2, 3 in Figure 3:

x is the origin: $\mathcal{D}_G(x)$={`N,E`},

or on the South border, and not the origin: $\mathcal{D}_G(x)$={`N,E,W`},

or on the West border, and not the origin: $\mathcal{D}_G(x)$={`N,E,S`}

or in the middle: $\mathcal{D}_G(x)$={`N,E,S,W`}.

We first run the automaton on $\mathbb{N} \times \mathbb{N}$. Let P be a Hamiltonian path defined by $\mathcal{A}$ in $G_1(\mathbb{N} \times \mathbb{N})$. It has a subpath $P[a, b]$ from a to b, for some a on the South border and some b on the West border, such that all intermediate vertices x have neighbourhood $\mathcal{D}_G(x)$={`N,E,S,W`}.The initial part $P[(0, 0), a]$ of P is inside the finite portion R of $G_1(\mathbb{N} \times \mathbb{N})$ (drawn on the plane, cf. Figure 5) determined by $P[a, b]$ and the West and South borders by an obvious planarity argument. We let R contain the vertices of $P[a, b]$ and the initial parts of the borders, from (0,0) to $(a, 0)$ and from (0,0) to $(0, b)$).

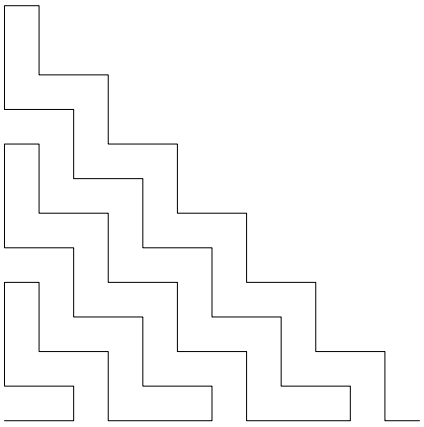

Figure 7: A d1-ordering of $\mathbb{N} \times \mathbb{N}$ that is adaptable to $X \times Y$ where X and/or Y is finite.

Let m be the maximal integer such that (m, j) belongs to $P[a, b]$ for some j. Let $c := (m + 1, j)$ for such a j. The vertex c is not in R.

We now consider $\mathcal{A}$ running in $G_1([0, m + 1] \times \mathbb{N})$. It follows the path $P[(0, 0), b]$, as it does not distinguish $G_1([0, m + 1] \times \mathbb{N})$ from $G_1(\mathbb{N} \times \mathbb{N})$ when traversing R. The path continues in $G_1([0, m+1] \times \mathbb{N})$ from b to c outside of R. But after c it must continue Southwards, and cannot reach $(m + 1, p)$ for large values of p. Hence we obtained the desired contradiction. □

We now enrich our automata by letting them foresee whether, from a vertex x, they can make two moves to East and/or two moves to North. The set of checkable directions around a vertex will be $\mathcal{E}$={`N`,`NN`,`E`,`EE`,`S`,`W`}. Clearly, if in $\mathcal{D}_G(x)$ we have `EE`, we must also have `E`. If `E` is in $\mathcal{D}_G(x)$ but `EE` is not, this means that x is at distance 1 of the East border.

We first encourage the reader to contemplate Figures 8, 9 and 10. The construction of the path in Figure 8 corresponding to the case $|X| = 7$, $|Y| = 5$ extends to $\mathbb{N} \times \mathbb{N}$, cf. Step 2 of the proof. Difficulties arise in the cases where X and/or Y have even cardinality. One typical case is shown in Figure 9 corresponding to the case $|X| = 7$, $|Y| = 6$. Another one is for $|X| = 8$, $|Y| = 6$ (Figure 10). Dotted lines indicate modifications from Figure 8.

Theorem 2.2. *There exists a finite $\mathcal{E}$-automaton that constructs a d1-ordering of $X \times Y$ for arbitrary (linearly ordered) sets X and Y.*

Proof. Step 1: The intended automaton will first handle the particular cases where X and/or Y have cardinality 1 or 2. This can be checked from $\mathcal{D}_G((0, 0))$ as $(0, 0)$ is the starting vertex. Hence, Y has cardinality 1 if and only if `E` is not

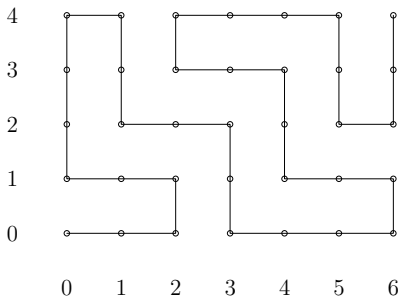

Figure 8: The basic case of $X \times Y$ with sets X, Y of odd cardinalities.

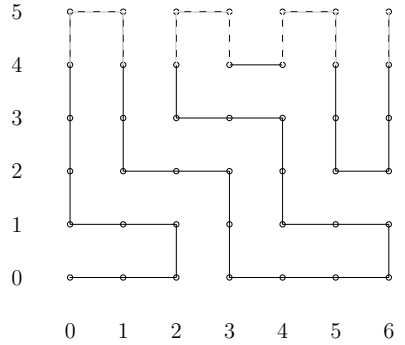

Figure 9: Extended the previous order to accomodate set Y of even cardinality.

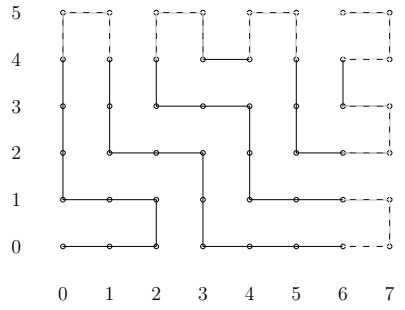

Figure 10: Extended the previous order to accommodate sets X, Y of even cardinality.

in $\mathcal{D}_G((0,0))$ and cardinality 2 if and only if `EE` is not in $\mathcal{D}_G((0,0))$ but `E` is. We omit details relative to these special cases.

Step 2: We now build an automaton $\mathcal{C}$ intended to work for all sets X,Y that are either infinite or of finite odd cardinality at least 3. It is based on the $\mathcal{D}_2$-automaton $\mathcal{B}$ of Proposition 1.1.

From $\mathcal{B}$, we first define a $\mathcal{D}_2$-automaton $\mathcal{B}'$ by duplicating actions: `N` becomes `N;N`, `SE` becomes `SE;SE`, etc. Because of the assumptions on the cardinalities of X and Y, from each vertex reached by a path defined by $\mathcal{B}'$, if one can make one step to East, one can make another one to East, and similarly for North, South and West. This automaton need not check the extended directions `EE` and `NN`.

A $\mathcal{D}_1$-automaton $\mathcal{C}$ is defined by Table 2, is obtained from $\mathcal{B}'$ by replacing respectively `SE;SE` by `E;S;S;E` and `NW;NW` by `N;W;W;N`. This replacement is made explicit in Table 2. The same numbering of types of positions on borders, at corners and in the middle is used, as for describing $\mathcal{B}$ and $\mathcal{B}'$, cf. Figure 3 and 12 below. As for $\mathcal{B}'$, one need not check the extended directions `EE` and `NN`.

Claim: The $\mathcal{D}_1$-automaton $\mathcal{C}$ defines a d1-ordering of $X \times Y$ for any sets X, Y as stated.

Proof: First we prove that $\mathcal{C}$ defines a path, *i.e.*, a walk that does not visit twice a same vertex.

A *2×2 square* in the grid $G_1(X,Y)$ is a subgraph induced by $[2p, 2p+2] \times [2q, 2q+2]$, for $p, q \geq 0$. Each of them can be colored black or white, so that two adjacent squares (adjacent by a border, not just a corner) are of different color. Let $[0,2] \times [0,2]$ be white. By $\mathcal{B}'$, it is traversed by the moves `NW;NW`. The moves of the form `SE;SE` traverse black *2×2* squares.

When defining $\mathcal{C}$, we replace `SE;SE` by `E;S;S;E`, so that we go through 2 more vertices, say x and y, in the middle of the left and right borders of that *2×2* square. In the surrounding *2×2* white squares, the replacements are of `NW;NW` by `N;W;W;N`, and these replacements involve neither x nor y. A similar observation holds at the borders.

Hence, we have a path. It is easy to check, by a similar argument based on this coloring of the *2×2* squares that it goes through all vertices of $G_1(X,Y)$.

If X and Y are finite, it terminates at the corner numbered 8, i.e., at $(\max(X), \max(Y))$, an example is in Figure 8.

Step 3: We must handle the three cases where $|X|$ and/or $|Y|$ is even. See Figure 4 showing the four types of borders and corners for finite sets X and Y.

Case 1: $|X|$ is odd or infinite, $|Y|$ is even, the vertices on the row just below the topmost one are in positions of types 5', 6' and 8' (cf. Figure 12) characterized by the following conditions relative to a vertex x:

5': `N,EE,S`$\in \mathcal{D}_G(x)$, `W,NN`$\notin \mathcal{D}_G(x)$,

6': `N,EE,W,S`$\in \mathcal{D}_G(x)$, `NN`$\notin \mathcal{D}_G(x)$,

8': `N,EE,W,S`$\in \mathcal{D}_G(x)$, `E,NN`$\notin \mathcal{D}_G(x)$.

If X is infinite, positions of types 4,7,8' do not occur.

In order to reach the vertices on the top row, we make small detours defined as follows. In case of the current state is `Down`:

> Action `E;S;S;E` from vertices of type 5 or 6 is replaced by `N;E;S;S;S;E`, from vertices of type 5' or 6' (see top part of Figure 14).
>
> From vertex 8', action is `N`, terminating the path.

In case of the current state is `Up`:

> Action `E;E` from vertices of type 5 is replaced by `N;E;S;E` from vertices of type 5'.
>
> Action `E;E` from vertices of type 6 is replaced by `N;E;S;S;S;E` from vertices of type 6'.
>
> From vertex 8', action is `N`, terminating the path.

Case 2: $|X|$ is even, $|Y|$ is odd or infinite. vertices on the column just to the right of the last one are of types 4", 7" and 8", characterized by the following conditions:

> 4": `N,E,W`$\in \mathcal{D}_G(x)$, `S,EE`$\notin \mathcal{D}_G(x)$,
>
> 7": `NN,E,S,W`$\in \mathcal{D}_G(x)$, `EE`$\notin \mathcal{D}_G(x)$,
>
> 8": `E,S,W`$\in \mathcal{D}_G(x)$, `N,EE`$\notin \mathcal{D}_G(x)$,

If Y is infinite, positions of types 5,6,8" do not occur.
In case of the current state is Down:

> Action `N;N` from vertices of type 4 is replaced by: `E;N;W;W;W;N` from vertices of type 4".
>
> Action `N;W;W;N` from vertices of type 4 is replaced by: `E;N;W;W;W;N` from vertices of type 4". (See Figure 13).
>
> Action `N;N` from vertices of type 7 is replaced by: `E;N;W;N`, from vertices of type 7",
>
> From vertex 8", action is `E`, terminating the path.

Case 3: $|X|$ and $|Y|$ are even. This case combines Cases 1 and 2. The relevant types of positions replacing 4,5,6,7,8 from the basic case are 4",5',6',7" characterized as above and

> 8": `N,E,S,W`$\in \mathcal{D}_G(x)$, `NN,EE`$\notin \mathcal{D}_G(x)$,

From a vertex of type 8", the action is `E;N;W`, terminating the path.
These definitions are collected in Table 3. □

Remark 2.2. It is clear that $\mathbb{Z}$ has no d1-ordering, and that, curiously, $\mathbb{Z} \times \mathbb{Z}$ has one, that is a kind of spiral around the origin. So has $\mathbb{N} \times \mathbb{Z}$ and thus $\mathbb{Z}^p$ for $p > 2$.

State	Position	Action	New state
`Down`	0,1	`E;E`	`Up`
	2,3,5,6	`E;S;S;E` (instead of `SE;SE`)	`Down`
	4,7	`N;N`	`Up`
`Up`	1,3,4,7	`N;W;W;N` (instead of `NW;NW`)	`Up`
	2	`N;N`	`Down`
	5,6	`E;E`	`Up`
`Up` or `Down`	8		`End`

Table 2: The automaton $\mathcal{C}$

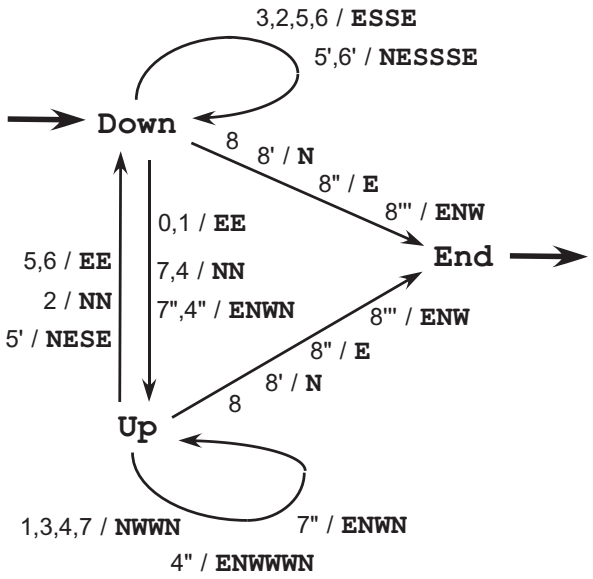

Figure 11: The $\mathcal{E}$-automaton of Theorem 2.2.

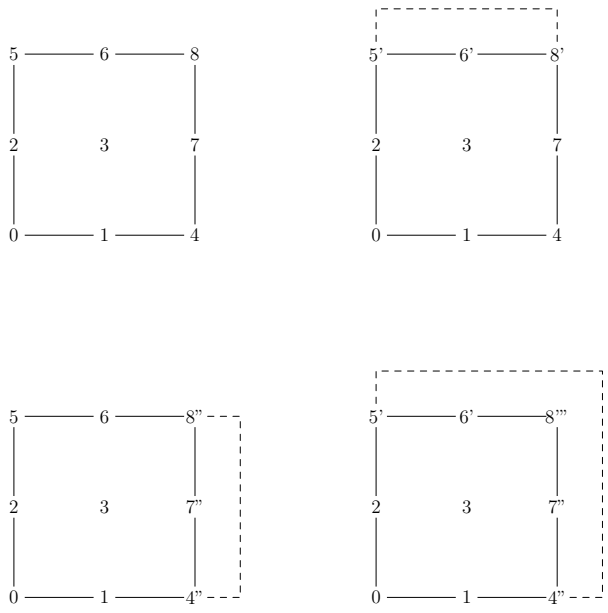

Figure 12: Numbering of types of positions relative to the borders. In particular, close to the North and East borders.

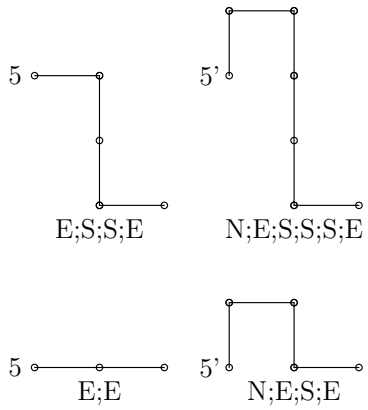

Figure 13: Some detours for vertices close to the North border.

State	Position	Action	Next state
`Down`	0,1	`E;E`	`Up`
	2,3,5,6	`E;S;S;E`	`Down`
	4,7	`N;N`	`Up`
	5',6'	`N;E;S;S;S;E`	`Down`
	4",7"	`E;N;W;N`	`Up`
	8'	`N`	`End`
	8"	`E`	
`Up`	1,3,4,7	`N;W;W;N`	`Up`
	2	`N;N`	`Down`
	5,6	`E;E`	`Down`
	4"	`E;N;W;W;W;N`	`Up`
	5'	`N;E;S;E`	`Down`
	6'	`N;E;S;S;S;E`	`Down`
	7"	`E;N;W;N`	`Up`
`Up` or `Down`	8		`End`
	8'	`N`	`End`
	8"	`E`	`End`
	8"'	`E;N;W`	`End`

Table 3: The $\mathcal{E}$-automaton of Theorem 2.2.

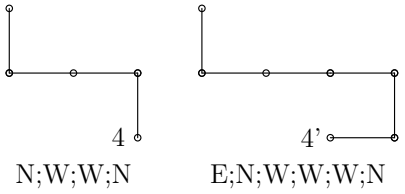

Figure 14: The detour at South-East corner.

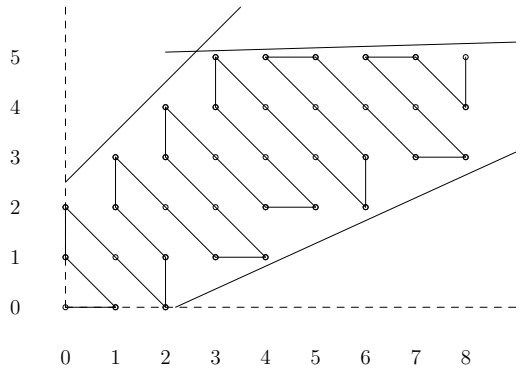

Figure 15: A proper subset of a Cartesian product ordered by Automaton $\mathcal{B}$.

3 Diagonal orderings of convex subsets of $\mathbb{N} \times \mathbb{N}$

Figure 15 shows that the automaton $\mathcal{B}$ of Section 1 can order proper subsets of $\mathbb{N} \times \mathbb{N}$ that are not Cartesian products. We develop this observation.

3.1 D2-ℓ-orderings of subsets of $\mathbb{N} \times \mathbb{N}$

We ask the following questions.

Question 3.1. Which subsets Z of $\mathbb{N} \times \mathbb{N}$ have a d2-ℓ-ordering?

We will consider $\mathcal{D}$-automata, more powerful than $\mathcal{D}_2$-automata as they can move to North-East in addition to North-West, North, East, South-West and South. As we want them to define d2-ℓ-*paths*, *i.e.*, the Hamiltonian paths corresponding to d2-ℓ-orderings, they will make no move to South-West.

Question 3.2. When is a d2-ℓ-ordering definable by a finite or infinite $\mathcal{D}$-automaton?

If Z is finite and d2-ℓ-orderable, then such an ordering is definable by a finite $\mathcal{D}$-automaton with $|Z|$ states. Hence, this question is only interesting for one infinite set Z or for a class of finite and/or infinite sets.

Definition 3.1. *Conditions on a set* $Z \subseteq \mathbb{N} \times \mathbb{N}$.

(a) We denote by Z_k the level k of Z. For each nonempty level Z_k, $\min(Z_k)$ (resp. $\max(Z_k)$) is its unique vertex of minimal (resp. maximal) second coordinate.

(b) We define for $Z \subseteq \mathbb{N} \times \mathbb{N}$ containing (0,0) the following conditions.

C1: The graph $G_2(Z)$ is connected.

C2: Each nonempty level Z_k is connected in $G_2(Z)$, hence, induces a North-West-South-East diagonal path.

C3: Each nonempty level can be labelled by `Up` or `Down`, so that if Z_k and $Z_{k'}$ are two consecutive nonempty levels with $k < k'$, then:

C3.1: if Z_k is labelled by `Down`, then $\min(Z_k)$ is adjacent in $G_2(Z)$ to the vertex x defined as follows:

$x := \min(Z_{k'})$ if $Z_{k'}$ is labelled by `Up`, or

$x := \max(Z_{k'})$ if $Z_{k'}$ is labelled by `Down`, or, otherwise,

$\{x\} = Z_{k'}$.

C3.2: if Z_k is labelled by `Up`, then $\max(Z_k)$ is adjacent in $G_2(Z)$ to x defined as in C3.1.

The label of a singleton level $Z_k = \{x\}$ can be equivalently `Up` or `Down` because $x = \min(Z_k) = \max(Z_k)$. Condition C1 implies that, if Z_k and $Z_{k'}$ are as in C3, then k' is $k+1$ or $k+2$. If $G_1(Z)$ is connected, then so is $G_2(Z)$ hence, Condition C1 holds; furthermore, if a level is not empty, all previous levels are not either, that is, we have $k'=k+1$ for k, k' as in C3.

Example 3.1. The set $Z := \{(0, 2i), (1, 2i+1) \mid i \geq 0\}$ satisfies Conditions C1,C2 and C3, with all levels labelled by `Up`. It has no level of odd height.

Proposition 3.1. *Let $Z \subseteq \mathbb{N} \times \mathbb{N}$ that contains (0,0).*

(1) It has a d2-ℓ-ordering if and only if Conditions C1, C2 and C3 hold.

(2) If all levels have at least two vertices except level 0 and, possibly, the last level (when Z is finite), then Z has at most two d2-ℓ-orderings.

(3) If there are p indices k such that $|Z_k| = 1$ and the following nonempty level has at least two elements, then Z has at most 2^p d2-ℓ-orderings.

Proof. (1) Clear from definitions.

(2) and (3) Assume that Z has a d2-ℓ -path P. It yields a labelling of levels satisfying Condition C3.

Consider $P[(0,0), x]$, the beginning of this path until vertex x, not the last one. Assume there is another d2-ℓ-path P' such that $P'[(0,0), x] = P[(0,0), x]$. Let Z_k be the level containing x.

These paths can differ immediately after x only if `N` and `E` are in $\mathcal{D}_{G_2(Z)}(x)$.

We consider five cases.

Case 1: $Z_k = \{x\}$. An example is in Figure 2 with $x = (0,0)$, see Remark 1.1 (1).

Case 2: Z_k is labelled `Down` and $x \neq \min(Z_k)$. The move after x must be to `SE` in P and in P'.

Case 3: Z_k is labelled `Up` and $x \neq \max(Z_k)$. The move after x must be to `NW` in P and in P'.

Case 4: Z_k is labelled `Down` and $x = \min(Z_k)$. The move after x must be to `E`, or if not possible, to `N`, or if not possible, to `NE`, in P and in P'.

Case 4: Z_k is labelled `Up` and $x = \max(Z_k)$. The move after x must be to `N`, or if not possible to `E`, or if not possible to `NE`, in P and P'.

The only case where P and P' might differ just after x is when $Z_k = \{x\}$ and $\mathcal{D}_{G_2(Z)}(x)$ contains `N` and `E`.

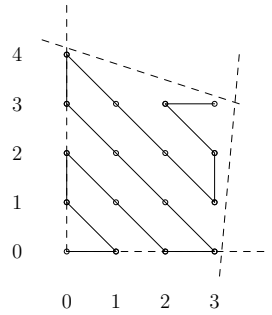

Figure 16: The set W of Example 3.2 (1) used in Proposition 3.2.

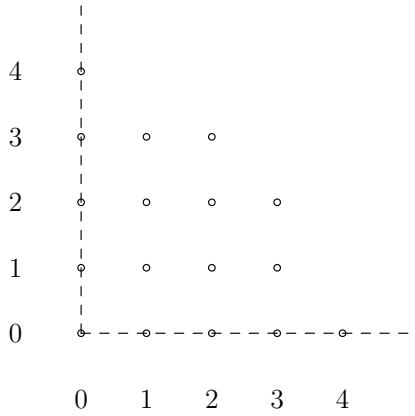

Figure 17: Set X of Example 3.2 (2)

Assertion (2) is an immediate consequence of this observation, because there is at most one vertex x satisfying this condition. With the hypothesis of Assertion (3), there are at most p such vertices, hence at most 2^p d2-ℓ-paths. □

Example 3.2 (3) below illustrates Assertion (3).

Example 3.2. (1) Figure 16 shows an example of a set $W \subseteq \mathbb{N}\times\mathbb{N}$ that satisfies Conditions C1 to C3. It has a d2-ℓ-path starting with (0,0)→(1,0), shown in this figure. An initial step (0,0)→(0,1) can be extended into a d2-ℓ-path until (0,4) but not after because $\max(W_4)$ and $\max(W_5)$ are not adjacent. Anticipating the sequel, we observe that W is defined by the conditions $i \leq 3$ and $j \leq -i/3 + 4$.

(2) The related set X of Figure 17 has no d2-ℓ-ordering, for a similar reason. It is defined by the conditions $j \leq -i/2 + 4$ and $j \leq -2i + 8$. It satisfies C1 and C2.

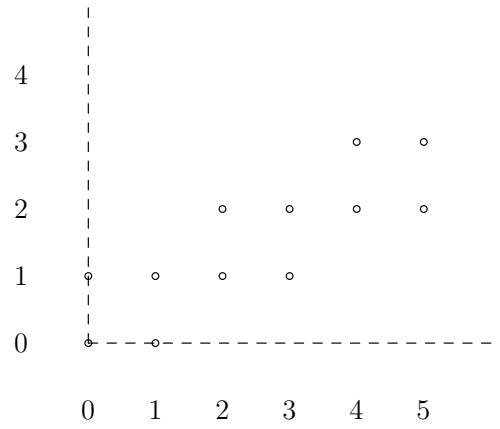

Figure 18: Set Y of Example 3.2 (3)

(3) The finite set Y shown on Figure 18 has 8 d2-ℓ-orderings. Three of them are:

00/10,01/11/21/31,22/32/42/52,43/53, defined by $\mathcal{B}$,

00/01,10/11/21/22,31/32/42/43,52/53, defined by $\mathcal{B}^{\#}$,and

00/10,01/11/21/22,31/32/42/52,43/53, defined neither by $\mathcal{B}$ nor by $\mathcal{B}^{\#}$.

It infinite extension Y' defined by the conditions $j \leq i/2+1$ and $j \geq i/2-1$ has infinitely many d2-ℓ-orderings.$\square$

We continue the study of sets $Z \subseteq \mathbb{N} \times \mathbb{N}$. If $G_1(Z)$ is connected and has a d2-ℓ-ordering that is definable by a $\mathcal{D}$-automaton, then this ordering is definable by a $\mathcal{D}_2$-automaton, actually the same, because no move to North-East can be used.

We recall that automata are deterministic and must have computable transitions, cf. Definition 1.4 (d).

Proposition 3.2. *(1) There exists an infinite set of finite sets $Z \subseteq \mathbb{N} \times \mathbb{N}$ that have unique d2-ℓ-orderings, but these orderings are not definable by any finite or infinite $\mathcal{D}$-automaton.*

(2) There exists an infinite set $Z \subseteq \mathbb{N} \times \mathbb{N}$ that has a unique a d2-ℓ-ordering that is not definable by any finite or infinite $\mathcal{D}$-automaton.

Proof. We let $W \subseteq [0,3] \times [0,4]$ shown in Figure 16. It has a unique d2-ℓ-path (defined by $\mathcal{B}$) from $s := (0,0)$ to $t := (3,3)$. Let $\overline{W} \subseteq [0,4] \times [0,3]$ be obtained from W a symmetry with respect to the South-West-North-East diagonal. It has a unique d2-ℓ-path (defined by $\mathcal{B}^{\#}$) also from $s := (0,0)$ to $t := (3,3)$.

(1) Let w_n be the word 0^n1. We define $X^{(n)} \subseteq \mathbb{N} \times \mathbb{N}$ by concatenating copies U_i of W or $\overline{W}$, such that U_i is a copy of W if $w_i = 0$ and of $\overline{W}$ otherwise. Two consecutive copies U_i and U_{i+1} are linked by an horizontal edge between

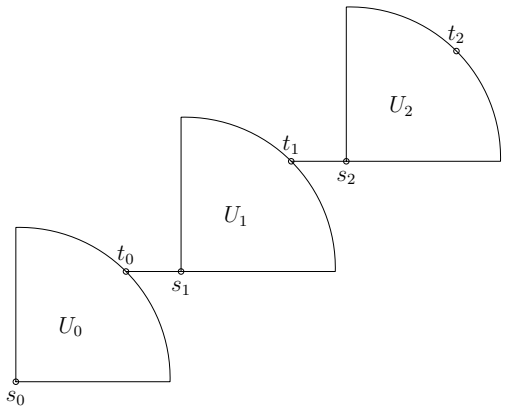

Figure 19: Set $X^{(2)}$ of Proposition 3.2.

t_i and s_{i+1}. See Figure 19 for $X^{(2)}$. Each set $X^{(n)}$ has a unique d2-ℓ-ordering. Assume that a $\mathcal{D}$-automaton (equivalently, a $\mathcal{D}_2$-automaton) can d2-ℓ-order all the sets $X^{(n)}$. When it reaches a vertex s_i, it cannot "know" whether the next move must be to North or to East, because it cannot know whether U_i is of type W or $\overline{W}$. Infinitely many states would not help.

(2) We now construct X similarly from an infinite word w in $\{0,1\}^\omega$. It has a unique d2-ℓ-ordering. If w is not ultimately periodic, this ordering cannot be defined by a finite $\mathcal{D}$-automaton, by an argument similar to that used in (1). It is definable by an infinite one (whose transitions must be computable, cf. Definition 4(d)) if there exists a computable function f_w: $\{0,1\}^* \to \{0,1\}$ such that $f_w(u)$ defines the letter 0 or 1 that follows u in w in the case where u is a prefix of w (otherwise, it yields 0). As there are uncountably many infinite words and countably many computable functions, there exists uncountably many words w in $\{0,1\}^\omega$ such that f_w is not computable, hence uncountably many sets X of the above form with unique d2-ℓ-orderings that are not definable by any $\mathcal{D}$-automaton. □

Remark 3.1. One might consider more powerful automata whose transitions from a vertex x are determined from the state and the *distance-p neighbourhood* of x for some p. A similar proof can be done with sets similar to W, of diameter larger than p. See also the conclusion.

3.2 A $\mathcal{D}$-automaton extending $\mathcal{B}$

We define a $\mathcal{D}$-automaton $\mathcal{F}$ that extends $\mathcal{B}$, intended to d2-ℓ-order sets Z such that $G_2(Z)$ is connected but $G_1(Z)$ is not. In Table 4, in the "Possible directions" column, "..." means "any". (The list of cases read from top to bottom can be implemented by IF THEN ELSE expressions). The initial state is `Down` and the final one is `End`.

State	Possible directions	Action	Next state
Down	SE,...	SE	Down
	¬SE, E,...	E	Up
	¬SE,¬E, N,...	N	Up
	¬SE,¬E,¬N, NE,...	NE	Up
	¬SE,¬E,¬N, ¬NE,...		End
Up	NW,...	NW	Up
	¬NW, N,...	N	Down
	¬NW, ¬N, E,...	E	Down
	¬NW,¬N,¬E,NE..	NE	Down
	¬NW,¬N,¬E,¬NE..		End

Table 4: The $\mathcal{D}$-automaton $\mathcal{F}$.

Example 3.3. (1) Let Z be defined by $2i/3 \leq j \leq 3i/2$. Its first levels are {(0,0)},∅, {(1,1)}, ∅,{(2,2)}, {(2,3),(3,2)}. A d2-ℓ-ordering can be defined by $\mathcal{F}$ that makes North-East moves (0,0)→(1,1)→(2,2) and then continues with the transition rules of $\mathcal{B}$.

(2) Let Z be defined by $i/2 \leq j \leq (i+1)/2$. Its first levels are {(0,0)},∅, {(1,1)},{(2,1)} ∅,{(3,2)}. All levels are singleton. It satisfies Conditions C1, C2 and C3.

Our next aim is to characterize the sets $Z \subseteq \mathbb{N} \times \mathbb{N}$ that are d2-ℓ-ordered by the $\mathcal{D}$-automaton $\mathcal{F}$ and the $\mathcal{D}_2$-automaton $\mathcal{B}$.

Definition 3.2. *More conditions on sets* $Z \subseteq \mathbb{N} \times \mathbb{N}$.
We consider the following variant of Condition C3

> C4: Each nonempty level is labelled by Down or by Up, in such a way that:
>
> C4.0: Z_0 is labelled by Down,
>
> C4.1: if $Z_{k'}$ follows Z_k labelled by Down, then it is labelled by Up and $\min(Z_k)$ is adjacent in $G_2(Z)$ to $\min(Z_{k'})$.
>
> C4.2: if $Z_{k'}$ follows Z_k labelled by Up, then it is labelled by Down and $\max(Z_k)$ is adjacent in $G_2(Z)$ to $\max(Z_{k'})$.

Note that if $G_2(Z)$ is connected, we have $k' = k+1$ in Conditions C4.1 and C4.2.

Example 3.1 satisfies Conditions C1-C3 but not Condition C4. By the next theorem, Conditions C1, C2 and C4 imply Condition C3. This fact can be checked directly without considering automata.

Theorem 3.1. *Let* $Z \subseteq \mathbb{N} \times \mathbb{N}$. *It is d2-$\ell$-ordered by* $\mathcal{F}$ *if and only if it satisfies Condition C1,C2 and C4. It is d2-ℓ-ordered by* $\mathcal{B}$ *if and only if* $G_1(Z)$ *is connected (which implies C1) and* Z *satisfies Condition C2 and C4.*

Proof. Let $Z \subseteq \mathbb{N} \times \mathbb{N}$ satisfy Condition C2 and C4.

If $G_1(Z)$ is connected, then, the automata $\mathcal{F}$ and $\mathcal{B}$ order Z by traversing the levels in order $Z_0, Z_1, ...$ They are in state `Down` on even levels and in state `Up` on the others.

If $G_2(Z)$ is connected but $G_1(Z)$ is not (some levels may be empty), the automaton $\mathcal{F}$ traverses levels in increasing order.

Conversely, consider a Hamiltonian d2-ℓ-path defined by $\mathcal{B}$. As its moves that increase the height of a vertex are to North and to East only, $G_1(Z)$ is connected. This path is a sequence of intervals, all elements of which have same height. This proves Condition C2. The transitions between two levels are by moves to North or to East. These transitions prove Condition C4.

The proof is similar for a path defined by $\mathcal{F}$. □

3.3 Sets that satisfy Conditions C1-C4?

We consider sets defined by conjunctions of arithmetical conditions, that are intersections of finitely many half-planes.

Definition 3.3. *Affine subsets* of $\mathbb{N} \times \mathbb{N}$.

We call *affine* a subset Z of $\mathbb{N} \times \mathbb{N}$ defined by the conjunction of finitely many conditions of the following forms, for specifying $(i, j) \in Z$:

(i) $i \leq a$,

(ii) $j \leq bi + c$ or

(iii) $j \geq di - e$

where $a, b, c, d, e \in \mathbb{Q}, a, c, d, e \geq 0$.

That $a, c, e \geq 0$ ensures that (0,0) is in Z. We restrict coefficients to rational numbers in order to be able to get algorithms for deciding certain properties of a given affine set Z listed below. Each level Z_k can be enumerated in a straightforward (brute force manner).

Question 3.3. (1) Is it finite?

(2) Is $G_2(Z)$ connected?

(3) Are Conditions C1-C3 satisfied?

(4) If they are, does there exist a $\mathcal{D}$-automaton[6] that defines a d2-ℓ-ordering?

The following examples show a variety of cases.

Example 3.4. We let Z be defined by the following conditions:

(1) $i/2 - 1/3 \leq j \leq i/2$. Then $Z = \{(2n, n) \mid n \in \mathbb{N}\}$ and $G_2(Z)$ is infinite without edges.

(2) $(i-1)/2 \leq j \leq i/2$. Then $Z = \{(2n, n), (2n+1, n) \mid n \in \mathbb{N}\}$, $G_2(Z)$ is connected but $G_1(Z)$ is not.

[6] One might also wish to order Z by a d2-ordering, that does not necessarily respect levels. We leave this study for future research.

(3) $i \leq j \leq i$. Then $G_2(Z)$ is an infinite diagonal South-West-North-East path and $G_1(Z)$ has no edge.

(4) $(4i-1)/10 \leq j \leq i/2$. Then $G_2(Z - \{(0,0)\})$ is connected but $G_2(Z)$ is not as Z_1, Z_2 and Z_3 are empty.

The sets Z of Cases 1 and 4 are not d2-ℓ-ordered by any automaton. Those of Cases 2 and 3 are by $\mathcal{F}$ but not by $\mathcal{B}$.

Definition 3.4. *Convexity properties*

We define for a subset Z of $\mathbb{N} \times \mathbb{N}$ the following convexity properties:

> *horizontal convexity*: if (i,j) and $(i+k,j) \in Z$, then $(i+k',j) \in Z$ for $0 < k' < k$,
>
> *vertical convexity*: if (i,j) and $(i,j+k) \in Z$, then $(i,j+k') \in Z$ for $0 < k' < k$.

These properties are preserved by intersection. They are satisfied by each set defined by a single inequality, hence by every affine set. Furthermore, a similar argument shows than an affine set satisfies the following

> *diagonal convexity*: if $(i,j+k)$ and $(i+k,j) \in Z$, then $(i+k', j+k-k') \in Z$ for $0 < k' < k$.

Theorem 3.2. *One can decide if an affine subset of* $\mathbb{N} \times \mathbb{N}$ *satisfies Conditions C1, C3 and C4.*

We recall that an affine subset of $\mathbb{N} \times \mathbb{N}$ satisfies Condition C2.

Lemma 3.1. *One can decide if an affine subset of* $\mathbb{N} \times \mathbb{N}$ *is finite. If so, one can enumerate it and check Conditions C1, C3 and C4.*

Proof. Let Z be an affine subset of $\mathbb{N} \times \mathbb{N}$. It is finite if and only if its description contains (among others) one inequality of the form:

(a) $j \leq bi + c$ with $b < 0$,

or two inequalities of the following forms:

(b) $i \leq a$ and $j \leq bi + c$, or

(c) $j \leq bi + c$ and $j \geq di - e$ with $d > b \geq 0$.

In each case, the set Z can be enumerated level by level. One can then check whether it satisfies Conditions C1, C3 and C4. □

We now assume that Z is infinite. In its definition, we can eliminate an inequality $i \leq a'$ if we already have the inequality $i \leq a$ with $a < a'$, and similarly for the equalities of types (ii) and (iii) (of Definition 3.3). After these eliminations, we obtain a *non redundant description.*

Example 3.5. The following affine sets Z are infinite.

(1) Set defined by conditions of type (i) and (iii): $j \geq i/2$ and $j \geq i-2$, $i \leq 5$.

(2) Set defined by conditions of type (ii) and (iii): $j \geq i/2$ and $j \geq i-10$, $i \leq j$.

(3) Set defined by $i = j$, hence, by conditions of type (ii) and (iii): $j \leq i \leq j$.

If $Z \subseteq \mathbb{N} \times \mathbb{N}$, and $p < q$, with $p, q \in \mathbb{N} \cup \{\infty\}$, we denote by $Z_{[p,q[}$ the union of the levels Z_k for $p < k < q$.

Lemma 3.2. *Let Z be an infinite subset of $\mathbb{N} \times \mathbb{N}$. Let Z_p be a nonempty level.The following equivalences hold:*

(1) $G_1(Z)$ is connected if and only if $G_1(Z_{[0,p+1[})$ and $G_1(Z_{[p,\infty[})$ are so.

(2) Z satisfies C1 and C3 if and only if $Z_{[0,p+1[}$ and $Z_{[p,\infty[}$ do so and the label `Up` *or* `Down` *of Z_p is the same in Condition C3 for $Z_{[0,p+1[}$ and $Z_{[p,\infty[}$.*

(3) If Z contains (0,0), then it satisfies C1 and C4 if and only if $Z_{[0,p+1[}$ satisfies C1 and C4, and $Z_{[p,\infty[}$ satisfies C1, and its nonempty levels have a labelling that satisfies C4.1 and C4.2, such that the label of Z_p is the same for $Z_{[0,p+1[}$ and $Z_{[p,\infty[}$.

Proof. (1) Clear because $Z_{[0,p+1[} \cap Z_{[p,\infty[} = Z_p \neq \emptyset$. The same holds for $G_1(Z)$, i.e, for Property C1.

(2) Clear from (1) and the definition of C3.

(3) Clear from (1) and the definitions. In the labelling of the levels of $Z_{[p,\infty[}$, instead of Condition C4.0, we require that, Z_p, the first nonempty level of $Z_{[p,\infty[}$ has the same label for $Z_{[0,p+1[}$ and $Z_{[p,\infty[}$. □

We now show how such an integer p can be determined in order to complete the proof of Theorem 3.2. We will use two variants of Conditions C4.1 and C4.2, relative to a set $Z \subseteq \mathbb{N} \times \mathbb{N}$:

C4.3.1: if $Z_{k'}$ follows Z_k, then, in $G_1(Z)$, the vertex $\min(Z_k)$ is adjacent to $\min(Z_{k'})$ and $\max(Z_k)$ is adjacent to $\max(Z_{k'})$,

C4.3.2: same condition with $G_2(Z)$ instead of $G_2(Z)$.

Proposition 3.3. *Let Z be an infinite affine subset of $\mathbb{N} \times \mathbb{N}$. One can determine if there exists an integer p such that the conditions of Lemma 3.2 are satisfied.*

Proof. Let Z be given by a non redundant description. We let $I(Z)$ be the set of pairs of coordinates $(x, y) \in \mathbb{Q} \times \mathbb{Q}$ of the intersection points in the plane of the lines associated with the defining inequalities, including the inequalities $i \geq 0$ and $j \geq 0$. We let $i(Z)$ be the smallest integer i such that $i \geq x + y$ for all (x, y) in $I(Z)$.

As Z is assumed infinite, it follows from the proof of Lemma 3.1 that we have three cases.

Case 1: If Z is defined by $i \leq a$ and inequalities of type (iii) ($j \geq di - e$). We can take $p = i(Z)$. Then Z_p is not empty, $G_1(Z_{[p,\infty[})$ is connected and satisfies C4.3.1. It follows that the nonempty levels of $Z_{[p,\infty[}$ have a labelling that satisfies C4.1 and C4.2. Whether Z satisfies C1, C3 or C4 depends only on the finite set $Z_{[0,p+1[}$ and can be checked.

Case 2 : Z is defined by inequalities of type (ii), of the form $j \leq bi + c$, and of type (iii), of the form $j \geq di - e$ with $b > d \geq 0$ for any such two inequalities. Let $\overline{b}$ be the minimal such b and $\overline{d}$ be the maximal such d.

We first consider W defined by the corresponding inequalities $j \leq \overline{b}i + \overline{c}$ and $j \geq \overline{d}i - \overline{e}$ and we let $r, q \in \mathbb{N}$ be such that $\overline{b} > r/q > \overline{d}$. Let $i \geq (\overline{d}q - \overline{e} - \overline{c})/(\overline{b} - \overline{d})$ and j such that $\overline{d}i + \overline{d}q - \overline{e} \leq j \leq \overline{b}i + \overline{c}$. We have in W the following vertices: $(i, j), (i + k, j)$ and $(i + r, j + k')$ for all $k = 1, ..., q$ and $k' = 1, ..., r$.

Hence, we get a path in $G_1(W)$ from (i, j) to $(i + q, j + r)$. This path can translated into a path from $(i + nq, j + nr)$ to $(i + (n + 1)q, j + (n + 1)r)$ for each $n > 0$. These translated paths concatenate into an infinite path in $G_1(W)$ starting from (i, j). By horizontal and vertical convexity, we obtain that $G_1(W)$ is connected.

We let $p := \max\{i + j, i(Z)\}$. It is clear that $Z \subseteq W$, Z_p is not empty and $G_1(Z_{[p,\infty[})$ is connected.

We now check Condition C4.3.1.

Let (i', j') be minimal in its level. This means that $\overline{d}i - \overline{e} + 1 \leq j \leq \overline{d}i + \overline{d} - \overline{e} + 1$. Then, since $G_1(Z_{[p,\infty[})$ is connected, (i', j') is adjacent to some vertex of height $h = i' + j' + 1$. The minimal vertex in Z_{h+1} is adjacent to (i', j') otherwise, $(i' + 2, j' - 1)$ would be in W. But we cannot have $j \geq \overline{d}i + 2\overline{d} - \overline{e} + 1$. A similar proof works for maximal elements. We are thus in the same situation as in Case 1.

Case 3: Z is defined by inequalities among which are $j \leq bi + c$ and $j \geq bi - e$ with $b = r/q \geq 0$. Let W be the set defined by these two inequalities (we may have $b = e = 0$ and, necessarily, $c \geq 1$). The other possible defining inequalities are of the forms $j \leq b'i + c$ and/or $j \geq di - e$ with $b' > b > d$.

Hence Z contains the vertex (nq, nr) for each integer n such that $nr + nq \geq i(Z)$.We let $p = m(r + q)$ be the minimal integer larger or equal to $i(Z)$ (with $m \in \mathbb{N}$). It is clear that $Z \subseteq W$, Z_p is not empty and $Z_{[p,\infty[} = W_{[p,\infty[}$. However, $G_2(Z_{[p,\infty[})$ need not be connected (see Example 3.4 (1)).

The finite set $Z_{[p,p+r+q+1[}$ is isomorphic to $Z_{[p+r+q,p+2(r+q)+1[}$ by a translation of vector (r, q), their intersection is Z_{p+r+q}. The set $Z_{[p,\infty[}$ is the union of the pairwise isomorphic sets $Z_{[p+n(r+q),p+(n+1)(r+q)+1[}$.

Then $G_1(Z_{[p,\infty[})$ (resp. $G_2(Z_{[p,\infty[})$) is connected if and only if $G_1(Z_{[p,p+r+q+1[})$ (resp. $G_2(Z_{[p,p+r+q+1[})$ is. Also $Z_{[p,\infty[}$ satisfies C3 (resp. C4) if and only if $Z_{[p,p+r+q+1[}$ satisfies C3 (resp. C4.3.1 or C4.3.2). These conditions are thus decidable. □

Proof. **of Theorem 3.2**
Let Z be given by inequalities. One can decide if it is finite, and decide Conditions C1, C3 and C4 by Lemma 3.1.

If Z is infinite, one eliminates the redundant equations and one uses Proposition 3.3. By Lemma 3.2, one can decide Conditions C1, C3 and C4. □

Open questions 3.1. (1) For handling quickly Case 3 of Proposition 3.3, can one find algebraic conditions relating r, q, c and e insuring that $G_1(W)$ or only $G_2(W)$ is connected?

(2) Is there an efficient algorithm for the decision problem of Theorem 3.2?

4 Dimension > 2

We consider next the definition of d2-ℓ -orderings of sets $X_1 \times X_2 \times ... \times X_p$ where $X_1, ..., X_p$ are finite or infinite linearly ordered sets, equivalently, $[0, m]$ or $\mathbb{N}$. We will prove that a unique automaton with 2^{p-1} states can define a d2-ℓ -ordering of any such a set, without knowing whether the components X_i are finite or infinite. We will use an induction on p for which we require more facts about orderings of $X_q \times ... \times X_p$.

4.1 Levels

We generalize the notion of level from Definition 1.1. We define it abstractly in a linearly ordered set. The notion of height will arise from that of level.

Definition 4.1. *Levelled linear order*

(a) A *levelled linear order* (*llo*) is a linear order Z defined as a finite or infinite concatenation of finite nonempty intervals $Z_0, Z_1, ..., Z_n, ...$ such that $Z_0 < Z_1 < ... < Z_n < ...$ Each interval is called a *level*. If $m \in Z_j$, then $ht(m) := j$ is the *height* of m.

We define $Lev(Z)$ as the linearly ordered set $\mathbb{N}$ if Z is infinite (all levels are nonempty), and $[0, p]$ if Z_p is the maximal nonempty level.

(b) The *product* of a linear order $X \subseteq \mathbb{N}$ and a llo Z is the llo on the set $U := X \times Z$ defined as in Definition 1.2, with a notion of type, that depends here on the levels of Z. The *Z-type* of a pair $(i, m) \in X \times Z$ is the triple of integers

$$\sigma(i,m) := \text{IF } i + ht(m) \text{ is even THEN } (i + ht(m), i, m) \\ \text{ELSE } (i + ht(m), ht(m), m).$$

A pair (i, m) can be determined from its type $\sigma(i, m)$.

The order $\leq_U$ on pairs (i, m) is increasing lexicographically on the Z-types $\sigma(i, m)$.The level U_k is the interval consisting of the pairs (i, m) such that $i + ht(m) = k$. It is important that each level of Z be finite in the case where Z is infinite.

Intuitively, U is obtained by substituting in $X \times Lev(Z)$ ordered as in Definition 1.4, the interval $i \odot Z_j$ to (i, j) where $i \odot (s_0, ..., s_q)$ denotes the linear order $(x, s_0), (x, s_1), (x, s_2), ..., (x, s_q)$.

Example 4.1. (1) An example of a llo is $Z^{(3,6)} := [0, 3] \times [0, 6]$ of Definition 1.4, that we can describe as /00/10,01/02,11,20/30,21,12,03/.../26/35/36/ by separating levels. Cf. Figure 2. Every linear order from Definition 1.4 is a llo, as a notion of level is defined.

(2) From a linear order X and an integer p, we can define a llo where each level has p elements, except the last one that may have less. For $X := \mathbb{N}$ and $p = 3$, we get the llo:

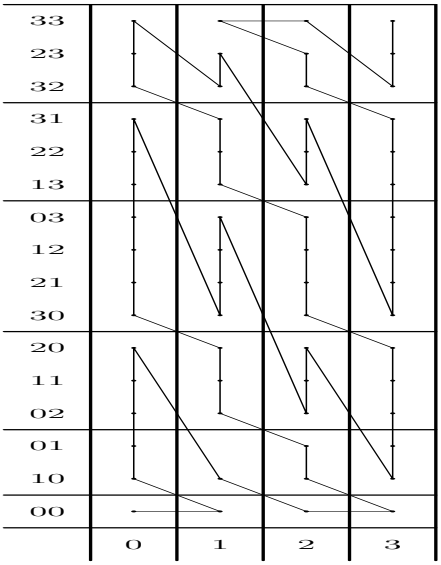

Figure 20: The d2-ℓ -ordering of Example 4.1 (5).

/012/345/678/....

(3) From a levelled linear order X and an integer q, we can define a llo by restricting X to its first q levels.

(4) The llo on $[0,3] \times [0,3]$ is:

/00/ 10, 01/ 02, 11, 20/ 30, 21, 12/13, 22, 31/32, 23/33/.

(5) The llo on $[0,3] \times ([0,3] \times [0,3])$ is (see Figure 20):

/000/ 100, 010, 001/ 002, 011, 020, 110, 101, 200/

300, 210, 201, 102, 111, 120, 030/013, 022, 031, 130, 121, 112,103,
202, 211, 220, 310, 301/.../233, 332, 323/333/.

Observation 4.1. *Concrete description of the levels of* $X \times Z$.

If X and Z are infinite, the levels U_i of the llo on $U := X \times Z$ of Definition 4.1 are as follows (where $\bullet$ denotes concatenation of sequences).

If i is even, then $U_i = (0 \odot Z_i) \bullet (1 \odot Z_{i-1}) \bullet ... \bullet (i \odot Z_0)$.

If i is odd, then $U_i = (i \odot Z_0) \bullet (i-1 \odot Z_1) \bullet ... \bullet (0 \odot Z_i)$.

We can determine the pair (y, z') that follows (x, z) in U, where $z \in Z_j$ and thus $(x, z) \in U_{x+j}$. There are three cases.

(a) z is not last in Z_j (so that (x,z) is not last in U_{x+j}). Then $y = x$ and z' follows z in Z_j.

(b) (x,z) is not last in U_{x+j} but z is last in Z_j.

If $x + j$ is even, we must have $j > 0$, otherwise z is last in Z_0 and (x,z) is last in $U_{x+j} = U_x$. Hence, we have $y = x+1$ and z' is the first element in Z_{j-1}.

If $x + j$ is odd, we must have $x > 0$, otherwise $(x,z) = (0,z)$ is last in $U_{x+j} = U_j$. Hence, we have $y = x - 1$ and z' is the first element in Z_{j+1} (cf. the definition of $\mathtt{back}_Z$ below).

(c) If (x,z) is last in U_{x+j} (hence, z is last in Z_j), we have two cases:

> if $x + j$ is even, then $j = 0$, z is last in Z_0, $y = x + 1$ and z' is the first element in Z_0; (possibly equal to z):
>
> if $x + j$ is odd, then $x = 0$, z is last in Z_j, $y = 0$ and z' is the first element in Z_{j+1}.

Here, the height of (y,z') is one more than that of (x,z). In the previous two cases, it is the same.

We can visualize Case (c) as follows:

> If i is even, then
>
> $U_i \bullet U_{i+1} = (0 \odot Z_i) \bullet ... \bullet (i \odot Z_0) \bullet ((i+1) \odot Z_0) \bullet ... \bullet (0 \odot Z_{i+1})$,
>
> if i is odd, then
>
> $U_i \bullet U_{i+1} = (i \odot Z_0) \bullet ... \bullet (0 \odot Z_i) \bullet (0 \odot Z_{i+1}) \bullet ... \bullet ((i+1) \odot Z_0)$.

In Case (b) the transition from the last element of Z_j to the first one in Z_{j-1} is called a *back step* in Z. In Case (c) the transition from z, last in Z_0, to z', first in Z_0, is also a back step inside the level Z_0 of Z, in the case where Z_0 is not singleton. However, Z_0 is singleton if $Z = X_q \times ... \times X_p$.

If X and/or Z is finite, this description must be modified. We define m_X in $\mathbb{N} \cup \{\infty\}$ as the least upper bound of X, and, similarly, M_Z as the least upper-bound of $Lev(Z)$. We fix $k \leq m_X + M_Z$ ($U_{m_X+M_Z}$ is the last nonempty level). To describe U_k, we define

> $\alpha := \max\{0, k - M_Z\} = k - \min\{k, M_Z\}$,
>
> $\beta := \max\{0, k - m_X\} = k - \min\{k, m_X\}$.

We have $\alpha \leq k - \beta \leq k$ because $k - M_Z \leq m_X$.

> If k is even, we have:
>
> $U_k = (\alpha \odot Z_{k-\alpha}) \bullet ((\alpha+1) \odot Z_{k-\alpha-1}) \bullet ... \bullet ((k-\beta) \odot Z_\beta)$.
>
> If k is odd, we have:
>
> $U_k = ((k-\beta) \odot Z_\beta) \bullet ((k-\beta+1) \odot Z_{\beta-1}) \bullet ... \bullet (\alpha \odot Z_{k-\alpha})$.

Regarding the determination of the next pair, Cases (a) and (b) described above are applicable. For Case (c) there are several subcases depending on whether x is maximal and Z_j is the maximal nonempty level of Z.

If k is even, then $U_k \bullet U_{k+1} =$

$(\alpha \odot Z_{k-\alpha}) \bullet ... \bullet ((k-\beta) \odot Z_\beta) \bullet ((k+1-\beta') \odot Z_{\beta'}) \bullet ... \bullet (\alpha' \odot Z_{k+1-\alpha})$

where $\beta' := k+1-\min\{k+1, m_X\}$ and $\alpha' = k+1-\min\{k+1, M_Z\}$.

We let (x,z) be the last element in $(k-\beta) \odot Z_\beta$, followed by (x',z') in $(k+1-\beta') \odot Z_{\beta'}$.

There are three cases to consider. Examples are from Figure 20.

(i) If $k < m_X$, then $\beta = \beta' = 0$, $x = k$, $x' = k+1$, and z' is the first element in $Z_0 = Z_\beta$. Example: $z = (2,00)$, $z' = (3,00)$.

(ii) If $k = m_X$, then $\beta = 0, \beta' = 1$, $x' = x = m_X$, and z' is the first element in Z_1, hence it follows z in Z.

(iii) If $k > m_X$, then $\beta = k - m_X, \beta' = \beta + 1$, $x' = x = m_X$, and z' is the first element in Z_β, hence it follows z in Z. Example: $z = (3,01)$, $z' = (3,02)$.

Similarly, if k is odd, then $U_k \bullet U_{k+1} =$

$(k - \beta \odot Z_\beta) \bullet ... \bullet (\alpha \odot Z_{k-\alpha}) \bullet (\alpha' \odot Z_{k+1-\alpha'}) \bullet ... \bullet (k - \beta' \odot Z_{\beta'})$.

We let (x,z) be the last element in $\alpha \odot Z_{k-\alpha}$, followed by (x',z') in $\alpha' \odot Z_{k+1-\alpha'}$.

(iv) If $k < M_Z$, then $\alpha = \alpha' = 0$, $x = x' = 0$, and z' is the first element in Z_{k+1}. Hence it follows z in Z. Example: $z = (0,23)$, $z' = (0,33)$.

(v) If $k = M_Z$, then $\alpha = 0 = x, \alpha' = 1 = x'$, and z' is the first element in $Z_k = Z_{M_Z}$.

(vi) If $k > M_Z$, then $\alpha = x = k - M_Z, \alpha' = 1 = x'$, and z' is the first element in Z_{M_Z}.

In case (i) we have a transition $z \longrightarrow z'$ inside Z_0. In cases (v), (vi), it is inside Z_{M_Z}.

Definition 4.2. *Back steps from last elements in their levels.*
(a) If Z is a llo, we define $\texttt{back}_Z$ as the set of pairs $(\max(Z_k), \min(Z_{k-1}))$ such that $k > 0$ and Z_k is not empty. If Z_k and Z_{k-1} are singleton, then $\min(Z_{k-1})$ is just the element preceding $\max(Z_k)$ in Z.

(b) The *Last in level* test $\texttt{Lil}_Z$ applied to $z \in Z$ means that z is the last element in its level.

Example 4.2. *Back steps in Cartesian products.*

Let $Z := [0,3] \times [0,3]$, $U := [0,3] \times Z$ and consider on Figure 20 the level $U_4 = ((0,31),(1,30),(1,21),(1,12),...,(2,20),(2,10),(2,10))$. We write ij a pair (i,j) of Z and (k,ij) a pair in U corresponding to a triple (k,i,j) in $[0,3] \times [0,3] \times [0,3]$.

Level U_4 has a transition (0,31)→(1,30) based on a back step in Z from 31 to 30 that decreases height in Z. In the llo on $U := [0,3] \times Z$, some other $\texttt{Back}_Z$ transitions are 20→10, 33→ 32, 36→26,16→06 and 01→00.

Construction 4.1. We define an automaton $\mathcal{A}_U$ intended to define the llo on $U := X \times Z$ (cf. Definition 4.1) where X and Z are llo's that satisfy the following conditions:

(1) All levels of X are singleton, and so are the minimal level Z_0 and the maximal one (if Z is finite).

(2) We are given an automaton $\mathcal{A}_Z$ for Z that defines back steps, and more precisely, such that, based on it, we have routines for the following tests and actions:

$\texttt{first}_Z$, $\texttt{last}_Z$, $\texttt{Lil}_Z$, $\texttt{next}_Z$, $\texttt{back}_Z$.

For X, $\texttt{Lil}_X(x)$ is always true, and $\texttt{back}_X$ is nothing but $\texttt{prev}_X$. We will use for it the routines $\texttt{first}_X$, $\texttt{last}_X$, $\texttt{next}_X$ and $\texttt{prev}_X$.

Describing automata with directions `N,E,SE` etc. is no more convenient. If X is an interval of integers, then

$\texttt{first}_X(x)$ is implemented by the test $(x = 0)?$,

$\texttt{last}_X$(x) is implemented by the test $(x = m_X)?$,

$\texttt{next}_X$ by $x := x + 1$, and

$\texttt{prev}_X$ by $x := x - 1$.

We use this notation for uniformity with those for Z that cannot be easily expressed from integers. (However, see Algorithm 4.9 below).

The minimal level U_0 and the maximal one (if U is finite) are singleton. This will allow us to use recursively this construction. For the same reason, we will build an automata $\mathcal{A}_U$ that defines the same five tests and actions as for Z.

Examples will be given concerning $U := [0, 3] \times Z$, where $Z := [0, 3] \times [0, 3]$, see Figure 20.

Construction:

The states of $\mathcal{A}_U$ are pairs (`Up`,s) and (`Down`,s) where s is a state of $\mathcal{A}_Z$.

The initial state is (`Down`,$\texttt{Init}_Z$) where $\texttt{Init}_Z$ is the initial state of $\mathcal{A}_Z$.

We define:

$\texttt{first}_U$`:=` $\texttt{first}_X$`∧`$\texttt{first}_Z$`,`

$\texttt{last}_U$`:=` $\texttt{last}_X$`∧`$\texttt{last}_Z$`.`

$\texttt{Lil}_U$ is defined by Table5.

In Tables 5, 6, 7 and 8, "State" indicates the first component of the state. The second component is used in the computations of $\texttt{first}_Z$, $\texttt{last}_Z$, $\texttt{Lil}_Z$, $\texttt{next}_Z$ and $\texttt{back}_Z$.The actions $\texttt{next}_U$ and $\texttt{back}_U$ are described in Tables 6, 7 and 8.

The number of states of the automaton for U is twice that of the automaton for Z. If all levels of Z are singleton, then $\mathcal{A}_Z$ has only one state.□

State	Subcondition	Examples
Up	$\texttt{first}_X \wedge \texttt{Lil}_Z$	(0,03)
Up	$\texttt{last}_Z$	(1,33)
Down	$\texttt{first}_Z$	(0,00),(2,00)
Down	$\texttt{last}_X \wedge \texttt{Lil}_Z$	(3,03),(3,01)
Up or Down	$\texttt{last}_X \wedge \texttt{last}_Z$	(3,33)

Table 5: The test $\texttt{Lil}_U$.

State = Down				New	
Conditions	Subcondition	Property	Action	state	Examples; cases
$\neg\texttt{Lil}_Z$			$\texttt{next}_Z$	Down	(1,21)→(1,12) (a)
$\texttt{Lil}_Z$	$\neg\texttt{last}_X \wedge \neg\texttt{first}_Z$		$\texttt{next}_X$; $\texttt{back}_Z$	Down	(1,03)→(2,02) (b)
$\texttt{Lil}_Z$	$\texttt{last}_X \wedge \neg\texttt{last}_Z$	$\texttt{Lil}_U$	$\texttt{next}_Z$	Up	(3,03)→(3,13) (ii)-(iii)
$\texttt{Lil}_Z$	$\texttt{first}_Z \wedge \neg\texttt{last}_X$	$\texttt{Lil}_U$	$\texttt{next}_X$	Up	(2,00)→(3,00) (i)
$\texttt{last}_X \wedge \texttt{last}_Z$		$\texttt{Lil}_U$	end		

Table 6: $\texttt{next}_U$, first part.

State=Up				New	
Condition	Subconditions	Property	Action	state	Examples; cases
$\neg\texttt{Lil}_Z$			$\texttt{next}_Z$	Up	(0,21)→(0,12) (a)
$\texttt{Lil}_Z$	$\neg\texttt{first}_X \wedge \neg\texttt{last}_Z$		$\texttt{prev}_X$; $\texttt{next}_Z$	Up	(2,03)→(1,13)
$\texttt{Lil}_Z$	$\texttt{first}_X \wedge \neg\texttt{last}_Z$	$\texttt{Lil}_U$	$\texttt{next}_Z$	Down	(0,23)→(0,33)
$\texttt{Lil}_Z$	$\neg\texttt{last}_X \wedge \texttt{last}_Z$	$\texttt{Lil}_U$	$\texttt{next}_X$	Down	(1,33)→(2,33)
$\texttt{last}_X \wedge \texttt{last}_Z$		$\texttt{Lil}_U$	end		(3,33)

Table 7: $\texttt{next}_U$, second part.

Conditions	Subcondition	Action	New state	Examples
Down	$\neg\texttt{first}_X \wedge \texttt{first}_Z$	$\texttt{prev}_X$	Up	(2,00)→(1,00)
Down	$\texttt{last}_X \wedge \texttt{Lil}_Z$	$\texttt{back}_Z$	Up	(3,03)→(3,02)
Up	$\texttt{firt}_X \wedge \texttt{Lil}_Z$	$\texttt{back}_Z$	Down	(0,03)→(0,02)
Up	$\neg\texttt{first}_X \wedge \texttt{last}_Z$	$\texttt{prev}_X$	Down	(1,33)→(0,33)

Table 8: $\texttt{back}_U$

4.2 Application to Cartesian products $U = X_1 \times X_2 \times ... \times X_n$.

Theorem 4.1. *There is a d2-ℓ -ordering of* $U = X_1 \times X_2 \times ... \times X_n$ *defined by an automaton with* 2^{n-1} *states.*

Proof. We use Construction 4.1 recursively by writing $U = X_1 \times Z = X_1 \times (X_2 \times (... \times X_n))$. To have a d2-ordering, we must check that the distance between consecutive elements is at most 2.

We consider the following property of a llo Z (intended to be $X_i \times (... \times X_n)$ for some i).

$P(Z)$: (a) Z_0 is singleton and so is the maximal level if Z is finite. (Think of $X_i \times (... \times X_n)$).

(b) The action $\texttt{back}_Z$ changes a single component.

(c) If $\texttt{Lil}_Z$ holds, then $\texttt{next}_Z$ changes a single component.

(d) If $\texttt{Lil}_Z$ does not hold, then $\texttt{next}_Z$ changes at most two components.

Property $P(Z)$ holds if all levels of Z are singleton. This so for $X_n \subseteq \mathbb{N}$. Note that $\texttt{Lil}_Z$ always holds and $\texttt{back}_Z$ is $\texttt{prev}_Z$.

Next we consider $P(U)$ where $U := X \times Z$ and $P(Z)$ holds.

We have (a).

(b) holds by the definition of $\texttt{back}_U$ in Table 8 and assertion (b) for Z.

(c) Assume that $\texttt{Lil}_U$ holds. From Table5, we have $\texttt{Lil}_U$ in all cases, because $\texttt{last}_Z$ implies $\texttt{Lil}_Z$ and so does $\texttt{first}_Z$ because Z_0 is singleton (by (a)).

Consider Table 6. The transition whose action is $\texttt{next}_X$; $\texttt{back}_Z$ and precondition is $\neg\texttt{last}_X \wedge \neg\texttt{first}_Z$ is not compatible with the condition $\texttt{Lil}_U$ which needs, in state `Down`, $\texttt{first}_Z$ or $\texttt{last}_X$. By $P(Z)$, all transitions change a single component. Similarly, consider Table 7. The transition whose action is $\texttt{prev}_X$; $\texttt{next}_Z$ and precondition is $\neg\texttt{first}_X \wedge \neg\texttt{last}_Z$ is not compatible with the condition $\texttt{Lil}_U$ which needs, in state `Up`, $\texttt{first}_X$ or $\texttt{last}_Z$. This proves (c).

(d) Clear from the tables. □

Condition (a) is necessary for Construction 4.1 to work.

The automaton is the same for sets X_i either infinite or finite with maximal value m_i.

Avoiding enumeration for computing the next element.

We take each X_i to be $[0, m_i]$ or $\mathbb{N}$, with known least upper-bound m_i. We wish to compute the n-tuple following a given one, say (3,0,0,2,4,0,0) to take an example, by a "direct" algorithm, without having to enumerate U until one reaches the given tuple and the one following it.

Proposition 4.1. *Let* $U := X_1 \times X_2 \times ... \times X_n$ *such that the values* m_i *are known. There exists an algorithm that, for input* $\boldsymbol{x} = (x_1, ..., x_n)$ *such that* $x_i \leq m_i$ *for all* i*, determines in time* $O(n)$ *the* n*-tuple that follows* $\boldsymbol{x}$ *in* $\leq_U$ *without enumerating* U*.*

Proof. We will compute $\texttt{next}_U((x_1,...,x_n))$ by means of at most n auxiliary computations of $\texttt{Lil}_{X_i\times...\times X_n}((y_i,...,y_n))$, $\texttt{next}_{X_i\times...\times X_n}((y_i,...,y_n))$ and $\texttt{back}_{X_i\times...\times X_n}((y_i,...,y_n)))$ for $1\leq i\leq n$ and tuples $(y_i,...,y_n)$.

To simplify notation, we will use $\texttt{Lil}_i(y_i,...,y_n)$, for $\texttt{Lil}_{X_i\times...\times X_n}((y_i,...,y_n))$, and similarly for next and back. We fix $U := X_1 \times X_2 \times ... \times X_n$ such that the values m_i are known.

If $(y_i,...,y_n) \in X_i \times ... \times X_n$, $1 \leq i \leq n$, we define:

$$\Lambda_i(y_i,...,y_n) := (\texttt{Lil}_i(y_i,...,y_n),\texttt{next}_i(y_i,...,y_n),\texttt{back}_i(y_i,...,y_n)),$$

where $\texttt{Lil}_i(y_i,...,y_n) \in \{True, False\}$, $\texttt{next}_i(y_i,...,y_n)$ is $\perp$ (undefined) if $(y_i,...,y_n) = (m_i,...,m_n)$ and is in $X_i \times ... \times X_n$ otherwise, $\texttt{back}_i(y_i,...,y_n)$ is $\perp$ if $\texttt{Lil}_i(y_i,...,y_n) = False$ or $(y_i,...,y_n) = (0,...,0)$ and is in $X_i \times ... \times X_n$ otherwise.

We compute $\Lambda_1(x_1,...,x_n)$ by recursion, by means of $\Lambda_2(...)$,...,$\Lambda_n(...)$ for appropriate arguments.

For computing $\Lambda_i(y_i,...,y_n)$, we use Tables 5, 6, 7 and 8. The state is Up if $y_i + ... + y_n$ is odd and Down it is even.

If $i = n$, then $\Lambda_n(y_n) := (True,\ y_n + 1, \perp)$ (or $(True,\ \perp, \perp)$ if $y_n = m_n$).

We now examine how to compute $\Lambda_i(y_i,...,y_n)$ if $i < n$.

For computing $\texttt{Lil}_i(y_i,...,y_n)$ (cf. Table5), we use:

$(y_i = 0)$? for $\texttt{first}_X$, $(y_i = m_i)$? for $\texttt{last}_X$,

$\texttt{Lil}_{i+1}(y_{i+1},...,y_n)$ for $\texttt{Lil}_Z$ and

$((y_{i+1},...,y_n) = (m_{i+1},...,m_n))$? for $\texttt{last}_Z$.

For computing $\texttt{next}_i(y_i,...,y_n)$ and $\texttt{back}_i(y_i,...,y_n)$(cf. Tables 6, 7 and 8), we use:

$((y_{i+1},...,y_n) = (0,...,0))$? for $\texttt{first}_Z$,

$\texttt{next}_{i+1}(y_{i+1},...,y_n)$ for $\texttt{next}_Z$,

$\texttt{back}_{i+1}(y_{i+1},...,y_n)$ for $\texttt{back}_Z$

and, the same definitions as above for $\texttt{first}_X$, $\texttt{last}_X$ and $\texttt{last}_Z$. □

Example 4.3. Here are some particular cases and examples:

(1) If $\boldsymbol{x} = (m_1,...,m_n) \in \mathbb{N}^n$, there is no next element because $\boldsymbol{x}$ is last in U.

(2) If $\boldsymbol{x} = (2p, 0, ..., 0)$ and $0 \leq 2p < m_1$, or $\boldsymbol{x} = (0, 0, ..., 0, 2p + 1)$ and $0 < 2p + 1 < m_n$, then, $\boldsymbol{x}$ is last in its level and the following element $\boldsymbol{x'}$ is, respectively, $(2p + 1, 0, ...,)$ or $(0, 0, ..., 0, 2p + 2)$.???

(3) *Example*: $\boldsymbol{x} = 302400 \in \mathbb{N}^6$, $m_1 > 3$, $m_4 = 2, m_5 = 4, m_2, m_3, m_6, m_7 > 0$,

For $\boldsymbol{x} = 302400$, the state is Up.

$\texttt{Lil}_6(\boldsymbol{x}) = False$,

$\texttt{next}_6(\boldsymbol{x}) = 3\bullet\texttt{next}_5(02400)$.
State is now Down.
$\texttt{Lil}_5(02400) = False$,
$\texttt{next}_6(\boldsymbol{x}) = 3\bullet 0\bullet\texttt{next}_4(2400)$
$\texttt{Lil}_4(2400) = \texttt{Lil}_3(400) = True$
$\texttt{next}_4(2400) = 2\bullet\texttt{next}_3(400) = ... = 2401$.
Hence $\texttt{next}_6(\boldsymbol{x}) = 302401$.
Note that we did not need to compute back in this case.
(4) *Example*: $\boldsymbol{x} = 4323 \in \mathbb{N}^4$, $m_4 = 5$, $m_1 = m_2 = m_3 = 1$.
$\texttt{Lil}_4(4323)=False$, $\texttt{Lil}_3(323)=True$,
$\texttt{next}_4(\boldsymbol{x}) = (4{+}1)\bullet\texttt{back}_3(323) = 5313$.

Remark 4.1. Computation is accelerated if we note the following facts, state as in Proposition 4.1.

Claim 1: $\texttt{Lil}_i(y_i, ..., y_n)$ implies $\texttt{Lil}_{i+1}(y_{i+1}, ..., y_n)$.
and
Claim 2: $\neg\texttt{Lil}_{i+1}(y_{i+1}, ..., y_n)$ implies $\texttt{next}_i(y_i, ..., y_n) = y_i\bullet\texttt{next}_{i+1}(y_{i+1}, ..., y_n)$.

Open questions 4.1. (1) With the hypothesis of Proposition 4.1, give an algorithm to determine the rank $rk(\boldsymbol{x})$ of tuple $\boldsymbol{x}$ in the ordering $\leq_U$.

(2) Conversely, determine the tuple of given rank i.

This is easy in the case of Definition 1.2, *i.e.* when $n = 2$ and $m_1 = m_2 = \infty$. Then

$rk(i, j)=$ IF $i + j$ is even THEN $(i + j + 1)(i + j + 2)/2 - j$ ELSE
$(i + j + 1)(i + j + 2)/2 - i$.

Conversely, given $rk(i, j) = n$, one determines $i + j$ as the least integer m such that $(m + 1)(m + 2)/2 \geq n$ from which one obtains i and j depending on its parity.

5 Conclusion

We presented open questions in the previous sections. Here are some more.

(1) Which "simple" automata can define d1-orderings of $\mathbb{N} \times \mathbb{N} \times \mathbb{N}$, and more generally of every product $\mathbb{N} \times \mathbb{N} \times ... \times \mathbb{N}$?

(2) Which automata can define d2-orderings (not respecting levels)?

(3) What about sets defined by Boolean combinations of affine inequalities? They may not be convex.

(4) Consider an affine subset $Z \subseteq \mathbb{N} \times \mathbb{N} \times ... \times \mathbb{N}$ defined by conditions of the form

$a_1 i_1 + ... + a_n i_n \leq b$ with $b \geq 0$ so that it contains (0,0,..0). Does there exist a finite automaton that d2- ℓ-orders it? We need to generalize Conditions C1-C4 to larger dimensions. See Example 3.2 (2) Figure 17.

References

[1] B. Courcelle and I. Durand: Computations by fly-automata beyond monadic second-order logic. *Theor. Comput. Sci.* **619** (2016) 32-67.

[2] I. Durand, Object enumeration, *European LISP Conference*, 2012, Zadar, Croatia. https://european-lisp-symposium.org/static/proceedings/2012.pdf

[3] J. Engelfriet and H.J. Hoogeboom, Automata with Nested Pebbles Capture First-Order Logic with Transitive Closure. *Logical Methods in Computer Science* **3** (2007), issue 2.

[4] P. Fraigniaud, D. Ilcinkas, G. Peer, A. Pelc and D. Peleg: Graph exploration by a finite automaton. *Theor. Comput. Sci.* **345** (2005) 331-344.

[5] Wikipedia, Gray code, https://en.wikipedia.org/wiki/Gray_code

6 Appendix

The `Enum` package is part of the `TRAG`[7] system which is written in `Common Lisp`. The code can be found at `https://idurand@bitbucket.org/idurand/trag.git`. The first version of this package was presented in [2]. It offered the possibility of creating basic enumerators (inductive, from a list, ...) and combining then using operations like products, sequences, filters... The general product built on the binary product does not give a $d2$-ordering. Here we give some hints about how we programmed a bidirectional leveled enumerator which enumerates a (possibly infinite) cartesian product with a d2-ℓ-ordering.

1 Enumerators and bidirectional enumerators

1.1 General enumerators

In the following, an enumerator E may be identified with the enumerated sequence. In the `Enum` package, each enumerator `E` has at least the following operations:

- `next-element-p (E)`: does there exist a next element?
- `next-element (E)`: move to the next element.

For the implementation, we also need

- `init-enumerator (E)`: put `E` in its initial state
- `copy-enumerator (E)`: independent copy of `E`

[7] `trag.labri.fr`

Examples with a finite enumerator

```
ENUM> (defparameter *ABC* (make-list-enumerator '(A B C))) => *ABC*
ENUM> (next-element *ABC*) => A
ENUM> (next-element *ABC*) => B
ENUM> (next-element-p *ABC*) => T
ENUM> (next-element *ABC*) => C
ENUM> (next-element-p *ABC*) => NIL
ENUM> (collect-enum *ABC*) => (A B C) ;; only if finite
```

Examples with an infinite enumerator

```
ENUM> (defparameter *naturals*
        (make-inductive-enumerator 0 (lambda (n) (1+ n)))) => *NATURALS*
ENUM> (next-element *naturals*) => 0
ENUM> (next-element *naturals*) => 1
ENUM> (next-element *naturals*) => 2
ENUM> (init-enumerator *naturals*) => #<INDUCTIVE-ENUMERATOR {100B58E013}>
ENUM> (next-element *naturals*) => 0
ENUM> (next-element *naturals*) => 1
ENUM> (next-element-p *naturals*) => T ;; always true
ENUM> (collect-n-enum *naturals* 9) => (0 1 2 3 4 5 6 7 8) ;; the first 9 values
```

1.2 Bidirectionals enumerators

A *bidirectional enumerator* `B` has in addition a `way` (`+1` to move forward, `-1` to move backwards), an `initial-way` and the following operations to handle them:

- `initial-way (B)`: initial way
- `change-initial-way (way, B)`: change `initial-way` to `way`
- `way (B)`: current way
- `inverse-way (B)`: inverse `way` of `B`

together with the following operations:

- `way-next-element-p (way B)`: does there exist a next element in this `way`?
- `way-next-element (way B)`: move to the next element in this `way`.
- `latest-element (B)`: last object enumerated.

The operations `next-element-p` and `next-element` can be written with `way-next-element-p` and `way-next-element`:

```
(defun next-element-p (B) (way-next-element-p (way B) B))
(defun next-element (B) (way-next-element (way B) B))
```

The implementation of a bidirectional enumerator uses two stacks `past-objects` and `future-objects`, the first one containing the already enumerated objects that are before the latest enumerated object and the second the ones that are after, and a slot `latest-object` containing the latest enumerated object. If the enumerator is moving backwards, the top element of `past-object` will be enumerated and moved to `latest-object`; otherwise the top element of `future-object` will be enumerated and moved to `latest-object`.

Creation and initialization of a bidirectional enumerator

Given a non empty enumerator `E`, enumerating $e_0, e_1, \ldots$, one can obtain its bidirectional version `B-E` with `(make-bidirectional-enumerator E initial-way)`. In `B-E`, one has access to `E`, the *underlying enumerator*, by `(enum B-E)`. At initialization, if `initial-way` is `-1`, we move forward `(enum B-E)` towards the first element in the positive way, so towards the first element of `E`, e_0, in order to go back to this element at the next call of `next-element`. Consequently, the first call `(next-element-p B-E)` will return `T`, the first call `(next-element B-E)` will return the first element of `E` that is e_0; then `(next-element-p B-E)` will return `Nil` as long as its `way` remains `-1`.

Example of creation and use of a bidirectional enumerator

```
ENUM> (defparameter *B-NATURALS*
        (make-bidirectional-enumerator *naturals*)) => *B-NATURALS*
ENUM> (next-element *B-NATURALS*) => 0
ENUM> (next-element *B-NATURALS*) => 1
ENUM> (next-element *B-NATURALS*) => 2
ENUM> (way *B-NATURALS*) => 1
ENUM> (inverse-way *B-NATURALS*) => -1
ENUM> (way *B-NATURALS*) => -1
ENUM> (next-element *B-NATURALS*) => 1
ENUM> (next-element *B-NATURALS*) => 0
ENUM> (next-element-p *B-NATURALS*) => NIL
ENUM> (inverse-way *B-NATURALS*) => 1
ENUM> (next-element-p *B-NATURALS*) => T
ENUM> (next-element *B-NATURALS*) => 1
ENUM> (next-element *B-NATURALS*) => 2
ENUM> (latest-element *B-NATURALS*) => 2
ENUM> (way-next-element -1 *B-NATURALS*) => 1
```

2 Enumeration of cartesian products

Let $E_1, \ldots, E_p$ be non empty enumerators (finite or not) such that E_i enumerates $e_0^i, e_1^i, \ldots$ if E_i is infinite, $e_0^i, e_1^i, \ldots, e_{c_i-1}^i$ where $c_i = card(E_i)$ otherwise. Let $T_p = E_1 \times E_2 \ldots \times E_p$ the cartesian product of the sets associated with the E_i. If all the E_i are finite, $card(T_p) = \Pi_{i=1}^{p} c_i$ otherwise T_p is infinite. The

necessity of diagonal enumeration of a cartesian product arises in particular when at least one of the components is infinite. For instance in the example above, when enumerating `*ABC*` × `*naturals*`, we would no want to enumerate: `(A 0)` `(A 1)` `(A 2)` `(A 3)` `...` and never switch to the `B` value of the `*ABC*` enumerator. We would rather want something like

```
ENUM> (defparameter *p* (make-product-enumerator (list *ABC* *naturals*)))
*P*
ENUM> (collect-n-enum *p* 10)
((A 0) (A 1) (B 0) (C 0) (B 1) (A 2) (A 3) (B 2) (C 1) (C 2))
```

where we move forward regularly on all enumerators.

2.1 Leveled enumerators (of cartesian products)

For diagonal enumeration we need the notion of height of a tuple in the cartesian product. The *height of a tuple* $t = (e^1_{j_1}, e^2_{j_2}, \ldots, e^p_{j_p}) \in T_p$, is the sum of the indices of the elements in the E_i: $l(t) = \Sigma_{i=0}^{p} j_i$. Let the cartesian product $T_p = \Pi_{i=1}^{p} E_i$. We note L^i, the i-th level (finite) of T_p which the set of tuples of height i. If T_p is finite, it has a finite number of levels and can be written as the partition of its levels. If T_p is infinite, its number of levels is infinite.

The definition of height and level in Definition 1.1 of Section 1 are a particular case the two above definitions when the enumerated sequences are $\mathbb{N}$ or intervals $[0, p] \subset \mathbb{N}$.

A *leveled enumerator* enumerates the levels $L^0, L^1, \ldots$ in increasing order. We call *major step*, a step giving a change of level and *minor step* a step inside a level. The leveled enumerators have in addition the predicate:

`minor-step-p (E)` which is true if the next step (`next-element`) does not change the level. In other words, it is false when we are done with the enumeration of the current level.

2.2 Bidirectionals leveled enumerators

A *leveled bidirectional enumerator* is a leveled enumerator which in addition, is bidirectional (it has a `way` and an `initial-way`). When going forward (`way = +1`), it enumerates the levels in increasing order: $L^0, L^1, \ldots$ When going backwards (`way = -1`), it enumerates the levels in decreasing order: $L^i, L^{i-1}, \ldots$ while keeping the forward order inside the levels.

2.3 Diagonal product of a bidirectional enumerator with a bidirectional leveled enumerator

Let `X` be a bidirectional enumerator and `Y` be a bidirectional leveled enumerator which when going forward enumerates the levels $Y^0, Y^1, \ldots$

We define below `DP(X, Y)`, the *diagonal product* of `X` and `Y`. When created `DP(X, Y)`, the initial way of `X` is set to `+1` and the initial way of `Y` is set to `-1`. A *minor step on level* is a step that changes the level of `X` in a way and the level of `Y` in the opposite way but not the level of `D`. In addition to the usual operations

we have the accessors `enum-x (D)` and `enum-y (D)` to access respectively to `X` and to `Y`. The other operations are written:

```
(defun latest-element (D)
  (cons (latest-element(enum-x D) (latest-element (enum-y D)))))

(defun minor-step-p (D) ;; precondition (next-element-p D)
  (and (next-element-p (enum-y D))
       (or (next-element-p (enum-x D)) (minor-step-p (enum-y D)))))

(defun way-next-element-p (way D)
  (or (way-next-element-p (way D) (enum-x D))
      (way-next-element-p (way(D) (enum-y D)))))

(defun way-next-element (way D)
  (let* ((enum-x (enum-x enum))
         (enum-y (enum-y enum))
         (next-x (next-element-p enum-x))
         (next-y (next-element-p enum-y)))
    (cond
      ((and next-y (minor-step-p enum-y)) ;; lower-level minor step
       (next-element enum-y))
      ((and next-y next-x) ;; minor-step on level
       ;; each one makes a major in its way
       (next-element enum-x) (next-element enum-y))
      ;; major step
      ((not (or next-x next-y))
       (corner-step enum-x enum-y way))
      (t (sliding-step enum-x enum-y way))))
  (latest-element enum))

(defun sliding-step (X Y way)
  ;; precondition: X or Y can move in its way
  (if (next-element-p Y)
      (way-next-element way Y)
      (way-next-element way X))
  (inverse-way X)
  (inverse-way Y))
```

The call `(sliding-step X Y 1)` corresponds to a *jump-up* (move to upper level) and `(sliding-step X Y -1)` corresponds to a *jump-back* (move to lower level). In the case where neither `X` nor `Y` can move in their current way and the enumeration is not over, we are in a case called *corner step* which may happen only when at least one of the enumerators is finite (otherwise there is always a possible *sliding step*). In the corner step case, we change inverse the way of the enumerator which goes in the opposite direction of `way` (of the product enumerator) and move it to the next level according to `way`. If `way = 1`, we move to the upper level. If `way = -1`, we move to the lower level. The other enumerator changes way (it could not contribute to the level change because it is blocked in the direction `way`).

```
(defun corner-step (X Y way)
  ;; change the way of the enumerator which goes in opposite direction
  ;; to way and move it; the other enumerator changes way
  (when (plusp (* way (way enum-x)))
    ;; put in enum-x the one that goes in direction -way
    (psetf enum-x enum-y enum-y enum-x))
  (inverse-way enum-x)  ;; enum-x will move in direction way
  (next-element enum-x) ;; enum-y will move in direction -way
  (inverse-way enum-y))
```

3 Diagonal enumeration of a cartesian product

Let `Nil` be the bidirectional leveled enumerator corresponding to the empty product enumerating the singleton set containing a single tuple of length 0: `Nil` $= \{()\}$; it has only one level $L^0 = \{()\}$.

We may show that recusrive use of `DP` yields the leveled-ℓ-ordering defined in Definition 4.1 and described in Observation 4.1.

Proposition 3.1. *Let $E_1, E_2, \ldots, E_p$ be bidirectional enumerators. The enumerator* DP$(E_1,$ DP$(E_2,$ DP$(\ldots,$ DP$(E_p,$ Nil$))))$ *is a bidirectional leveled enumerator and a leveled-ℓ-ordering of $T_p = \Pi_{i=1}^{p} E_i$.*

In the examples, we will use only integers so that the level of a tuple is the sum of its elements.

```
ENUM> (defparameter *e2* (make-list-enumerator '(0 1))) => *E2*
ENUM> (defparameter *e3* (make-list-enumerator '(0 1 2))) => *E3*
ENUM> (collect-enum *e2*) => (0 1)
ENUM> (collect-enum *e3*) => (0 1 2)
ENUM> (collect-enum (make-product-enumerator (list *e3* *e3*)))
((0 0) (1 0) (0 1) (0 2) (1 1) (2 0) (2 1) (1 2) (2 2))
ENUM> (collect-enum (make-product-enumerator (list *e3* *e3* *e3*)))
((0 0 0) (1 0 0) (0 1 0) (0 0 1) (0 0 2) (0 1 1) (0 2 0) (1 1 0) (1 0 1)
(2 0 0) (2 1 0) (2 0 1) (1 0 2) (1 1 1) (1 2 0) (0 2 1) (0 1 2) (0 2 2)
(1 2 1) (1 1 2) (2 0 2) (2 1 1) (2 2 0) (2 2 1) (2 1 2) (1 2 2) (2 2 2))
ENUM> (collect-n-enum (make-product-enumerator (list *naturals* *e3*)) 30)
((0 0) (1 0) (0 1) (0 2) (1 1) (2 0) (3 0) (2 1) (1 2) (2 2) (3 1) (4 0) (5 0)
(4 1) (3 2) (4 2) (5 1) (6 0) (7 0) (6 1) (5 2) (6 2) (7 1) (8 0) (9 0) (8 1)
(7 2) (8 2) (9 1) (10 0))
ENUM> (collect-n-enum (make-product-enumerator (list *naturals* *e3*)) 20)
((0 0) (1 0) (0 1) (0 2) (1 1) (2 0) (3 0) (2 1) (1 2) (2 2) (3 1) (4 0) (5 0)
(4 1) (3 2) (4 2) (5 1) (6 0) (7 0) (6 1))
ENUM> (collect-n-enum (make-product-enumerator (list *naturals* *e3* *e3*)) 20)
((0 0 0) (1 0 0) (0 1 0) (0 0 1) (0 0 2) (0 1 1) (0 2 0) (1 1 0) (1 0 1)
(2 0 0) (3 0 0) (2 1 0) (2 0 1) (1 0 2) (1 1 1) (1 2 0) (0 2 1) (0 1 2)
(0 2 2) (1 2 1))
```