Moscow Journal of Combinatorics and Number Theory

2020
vol. 9 no. 3

On defining linear orders by automata

Bruno Courcelle, Irène Durand and Michael Raskin

msp

Moscow Journal of Combinatorics and Number Theory
Vol. 9, No. 3, 2020
https://doi.org/10.2140/moscow.2020.9.253

On defining linear orders by automata

Bruno Courcelle, Irène Durand and Michael Raskin

Motivated by enumeration problems, we define linear orders $\leq_Z$ on Cartesian products $Z := X_1 \times X_2 \times \cdots \times X_n$ and on subsets of $X_1 \times X_2$ where each component set X_i is $[0, p]$ or $\mathbb{N}$, 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 z and z', at most two components differ, and they differ by at most 1.

We are interested in algorithms that determine the next tuple in Z by using local information, where "local" is meant with respect to certain graphs associated with Z. We want these algorithms to work as well for finite and infinite components X_i. We will formalise them by *deterministic graph-walking automata* and compare their enumeration powers according to the finiteness of their sets of states and the kinds of moves they can perform.

Introduction

This article is motivated by the construction of enumeration algorithms[1] [Durand 2012]. An *enumerator* E_A of a set A is an algorithm that lists its elements. For an example, we may wish to list the minimal dominating sets of a given graph. Each element is determined from the previous one and the current state of the algorithm. The set A may be countable, for example the set of prime numbers. These algorithms can allow repetitions or, on the contrary, they can be designed to eliminate them.

New enumerators can be built from existing ones. For example, an enumerator E_A for a set $A = B \cup C$ can be built from enumerators E_B and E_C for B and C respectively. For the cases where B is finite, E_A can start by enumerating B and afterwards, C. Or, it can output alternatively an element of B and one of C. This is appropriate if B is infinite or if it is extremely large and its cardinality is not known. A library of basic enumerators and operations that build enumerators by combining existing ones has been defined and implemented by I. Durand [2012].

Our initial motivation was to enrich this library by constructions for the Cartesian product $A = B \times C$. Considering $B \times C$ as a matrix, one can enumerate its elements row by row if C is finite, or column by column if B is finite, or in a diagonal way, as in Cantor's enumeration of $\mathbb{N} \times \mathbb{N}$. The latter enumeration can be formalised by the polynomial $P(x, y) = x + (x + y)(x + y + 1)/2$ that defines the rank of a pair (x, y). We do not use it for two reasons. First we want a unique algorithm that works for B and C, either finite or infinite, but P does not compute consecutive ranks for the pairs in $\{0, 1, \ldots, p\} \times \mathbb{N}$. Furthermore, we do not want to use numbers that index the sets B and C. We want to build E_A from enumerators E_B and E_C that produce the next element or report the end of the enumeration. Actually, we will use

MSC2010: 06A05, 05C38, 68R10, 68P10.
Keywords: enumeration algorithm, diagonal enumeration, graph-walking automaton, linear order.

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

enumerators E_B and E_C that can also produce the previous element, which is easily implementable with a stack.

An enumerator E_A can be seen as an automaton that produces a sequence of elements of A, hopefully exhausting it. It produces a linear order if it does not allow repetitions, and the order type is that of $\mathbb{N}$ if A is infinite.

This view is adequate for the case of $A = B \times C$ where B and C are enumerated without repetitions by E_B and E_C, and hence are linearly ordered. Then $B \times C$ can be considered as graph, shaped as a rectangular, possibly infinite, grid. An enumerator E_A can be formalised as a deterministic graph-walking automaton that traverses the grid and builds a path spanning it that represents the intended enumeration. A *graph-walking automaton* has a "head" that can move in the graph (the grid) from a vertex to a neighbouring one. The decision where to move is taken from the current state and the knowledge of the neighbouring vertices. It is convenient to use north, west, south, southeast etc. as *directions* for describing the moves. For example, the current position is on the eastern border of a finite grid if there is no east directed edge from it.

We propose different constructions of graph-walking automata, and hence of enumerators for Cartesian products $Z := X_1 \times X_2 \times \cdots \times X_n$ and for certain *affine* subsets Z of $X_1 \times X_2$ where each component set X_i is linearly ordered and has 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. Our orders are inspired by Cantor's 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 $z = (z_1, \dots, z_n)$ and $z' = (z'_1, \dots, z'_n)$, we have $|z_i - z'_i| \leq 1$ for each i. The reason is that we want each step to call the component enumerators for just one forward or backward step. (Cantors's enumeration does not satisfy this condition). Furthermore, we define their *distance* $d(z, z')$ 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 in order to minimise the number of calls to the component enumerators made at each step.

These requirements can be expressed in terms of graphs $G_1(Z)$ and $G_2(Z)$ that we describe informally for $Z = X_1 \times X_2$. The graph $G_1(Z)$ is a planar rectangular grid with horizontal and vertical edges, and $G_2(Z)$ is $G_1(Z)$ augmented with diagonal edges in each square. A d1-ordering (resp. a d2-ordering) of Z is a Hamiltonian path in $G_1(Z)$ (resp. in $G_2(Z)$) starting at $(0, 0, \dots, 0)$.

We are interested in algorithms that determine the tuple in Z following a tuple z by using local information, where *local* is meant with respect to $G_1(Z)$ or $G_2(Z)$, and that work as well for finite and infinite components X_i. We will formalise them by means of *deterministic graph-walking automata*, whose runs on a given graph define *walks* (a walk is like a path, but vertices can be visited several times). We will actually construct 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 that we call *directions*. The set of directions is finite. At a vertex reached by a walk, the automaton determines the direction of the next edge to be traversed from the *state* (it may have infinitely many states) and some knowledge of a finite neighbourhood, for example the set of directions of the incident edges, but larger neighbourhoods may be useful, as we will see. After the traversal via the next edge, the state may be changed, according to the used transition rule. The directions for $G_1(Z)$ and $G_2(Z)$ will be among north, west, south, southeast etc. We will not develop a general theory of graph-walking automata (see [Engelfriet and Hoogeboom 2007] for a study in relation with logic, or [Fraigniaud et al. 2005]), but we will define automata well-adapted to the graphs $G_1(Z)$ and $G_2(Z)$.

Some of our main theorems are informally stated as follows.

Theorem 1. *There is no finite or infinite automaton that defines a $d1$-ordering in each set $X_1 \times X_2$ where 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, which looks at distance 2.*

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. We want to build *d2-ℓ-orderings* where levels are traversed consecutively, by increasing order of height. We say that these orderings *respect levels*. No d1-ordering can respect levels, except in very particular cases.

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

The corresponding construction of E_Z from enumerators $E_{X_1}, \ldots, E_{X_2}$ has been implemented in the system TRAG [Durand 2012] (see also the Appendix).

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

Section 1 defines graph-walking automata. Section 2 describes d2-ℓ-orderings of sets $X \times Y$ where X and Y are $\mathbb{N}$ or parts of it, and the corresponding automata. Section 3 compares various d1-orderings of $X \times Y$ as in Section 2. Section 4 studies d2-ℓ-orderings of affine subsets of $\mathbb{N} \times \mathbb{N}$. Section 5 defines d2-$\ell$-orderings of Cartesian products $X_1 \times X_2 \times \cdots \times X_n$ for $n > 2$. Section 6 is a conclusion and the Appendix presents the implementation.

1. Graph-walking automata

Definition 1.1 (directions in graphs). Let $\mathcal{D}$ be a finite set called the set of *directions*. A $\mathcal{D}$-graph is a triple $G = (V, E, \mathrm{dir})$, where V and E are the vertex and edge sets of an undirected graph without loops and parallel edges, and dir is a partial mapping $V \times V \to \mathcal{D}$ such that, for all $x, y, z \in V$, $\mathrm{dir}(x, y)$ is defined if and only if x and y are adjacent, and $\mathrm{dir}(x, y) = \mathrm{dir}(x, z)$ implies $y = z$. We denote by x^d the vertex y such that $\mathrm{dir}(x, y) = d$. The *directions around* a vertex x are those d such that x^d is defined. We denote this set by $\mathcal{D}_G(x)$. It describes the *neighbourhood* of x in G.

Definition 1.2 (graph-walking automata in $\mathcal{D}$-graphs). (a) A *$\mathcal{D}$-graph-walking automaton* (or simply, a *$\mathcal{D}$-automaton*) is a tuple $\mathcal{A} = (Q, \mathcal{T}, q_{\mathrm{init}})$, where Q is the finite or countable set of *states*, $q_{\mathrm{init}} \in Q$ and $\mathcal{T}$ is the set of *transitions*: they are of the form $(q, \delta) \to (d, q')$ or $(q, \delta) \to \mathtt{End}$, where $q, q' \in Q$, $\delta \subseteq \mathcal{D}$ and $d \in \delta$. This means that 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 $(q, \delta) \to \mathtt{End}$) is computable.

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

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

defined with the help of the sequence of states

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

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 the neighbourhood $\mathcal{D}_G(b_n)$ of b_n determine in a unique way a direction d such that $(b_n)^d$ is defined and will be the next visited vertex b_{n+1}; the state q_n is updated to q_{n+1}.

We will construct deterministic $\mathcal{D}$-automata so that they define paths rather than walks.

Definition 1.3 (directions and automata in 2-dimensional grids). (a) Let $Z \subseteq \mathbb{N} \times \mathbb{N}$. We let $G_1(Z)$ be the graph with vertex set Z and edges between (x, y) and $(x, y+1)$ and between (x, y) and $(x+1, y)$. The associated directions are

$$\begin{aligned}
\operatorname{dir}((x, y), (x, y+1)) &:= \mathtt{N},\\
\operatorname{dir}((x, y+1), (x, y)) &:= \mathtt{S},\\
\operatorname{dir}((x, y), (x+1, y)) &:= \mathtt{E},\\
\operatorname{dir}((x+1, y), (x, y)) &:= \mathtt{W}.
\end{aligned}$$

The set of directions is $\mathcal{D}_1 := \{\mathtt{N},\mathtt{S},\mathtt{E},\mathtt{W}\}$.

We let $G_2(Z)$ be augmented with "diagonal" edges between $(x, y+1)$ and $(x+1, y)$ and between (x, y) and $(x+1, y+1)$. The associated directions are as above, together with $\operatorname{dir}((x+1, y), (x, y+1)) := \mathtt{NW}$ and similarly for the three other diagonal directions. The set of directions is $\mathcal{D}_2 := \{\mathtt{N},\mathtt{S},\mathtt{E},\mathtt{W},\mathtt{NW},\mathtt{SW},\mathtt{NE},\mathtt{SE}\}$.

(b) We will construct $\mathcal{D}_1$- and $\mathcal{D}_2$-automata that order linearly certain sets Z.

In the next section, we will extend the definitions of $G_1(Z)$ and $G_2(Z)$ to subsets Z of $X_1 \times X_2 \times \cdots \times X_n$ without extending the notion of direction.

2. Definitions and first results for Cartesian products

We will order linearly sets $Z := X_1 \times X_2 \times \cdots \times X_n$ and subsets of $X_1 \times X_2$, where each component set X_i is linearly ordered with order type ω and 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 $z = (z_1, \ldots, z_n)$ and $z' = (z'_1, \ldots, z'_n)$, we have $|z_i - z'_i| \leq 1$ for each i.

Definition 2.1 (distances, heights and levels). (a) The *distance* $d(z, z')$ of $z = (z_1, \ldots, z_n)$ and $z' = (z'_1, \ldots, z'_n)$ 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 only consider d1- and d2-orderings.

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

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

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

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

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

We now present a diagonal enumeration[2] of $\mathbb{N} \times \mathbb{N}$, its extension to certain subsets of the form $X \times Y$ and the corresponding $\mathcal{D}_2$-graph-walking automata.

Definition 2.2 (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 following pair, also in $\mathbb{N} \times \mathbb{N}$:

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

Note that (i, j) can be recovered from $\tau(i, j)$:

$$\tau^{-1}(m, n) = \text{IF } m \text{ is even THEN } (n, m - n) \text{ ELSE } (m - n, n).$$

The pairs $(i, j) \in \mathbb{N} \times \mathbb{N}$ are ordered by increasing lexicographic order of their types $\tau(i, j)$. That is,

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

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 with a slash. Odd levels are traversed in reverse lexicographic order.

The pair $\texttt{next}(i, j)$ that follows (i, j) in this order is obtained by the clauses below where we use "$\wedge$" as logical "and":

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

In terms of automata 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 "$i = 0$" characterises the western border and "$j = 0$" characterises the southern border. There are no northern and eastern borders.

Borders can also be detected by looking at the directions around the current position in the grid, that is a vertex of $G_2(\mathbb{N} \times \mathbb{N})$. For example, the southern border is characterised by the neighbourhoods {`N`, `E`, `NE`} and {`N`, `E`, `W`, `NW`, `NE`}. The second clause can be formalised by automaton transitions expressing the following:

$$(\texttt{Down}, \text{"on the southern border"}) \to (\text{"move to the east"}, \texttt{Up}),$$

[2]It is not Cantor's enumeration (see Remark 2.6)

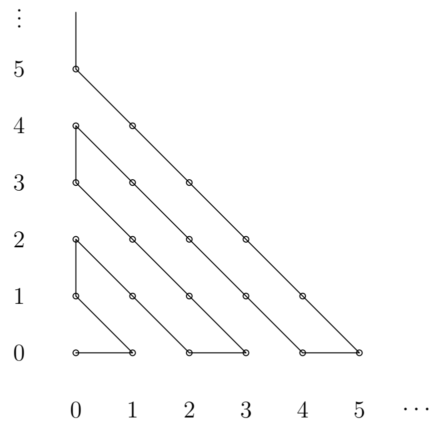

Figure 1. A d2-ℓ-ordering of $\mathbb{N} \times \mathbb{N}$.

that is, in terms of neighbourhoods (see Section 1)

$$(\texttt{Down}, \{\texttt{N}, \texttt{E}, \texttt{NE}\}) \to (\texttt{E}, \texttt{Up}), \quad (\texttt{Down}, \{\texttt{N}, \texttt{E}, \texttt{W}, \texttt{NW}, \texttt{NE}\}) \to (\texttt{E}, \texttt{Up}).$$

In Figure 1, these two transitions define respectively the edges $(0, 0) \to (1, 0)$ and $(2n, 0) \to (2n+1, 0)$ belonging to the path that orders $\mathbb{N} \times \mathbb{N}$. The edges directed to NW and SE, defined by the first and the third clauses, do not change the level $i + j$ of a pair (i, j) because they "go" from it to $(i - 1, j + 1)$ or $(i + 1, j - 1)$. The state defined from the arithmetic parity of the level is not changed.

The last clause yields the edge $(0, 3) \to (0, 4)$ derived from the transition expressing the following:

$$(\texttt{Up}, \text{“on the western border”}) \to (\text{“move to the north”}, \texttt{Down}).$$

Definition 2.3 (a d2-ℓ-ordering of $X \times Y \subseteq \mathbb{N} \times \mathbb{N}$). We modify the algorithm of Definition 2.2 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 in 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.

In the first clause, the pair $\texttt{next}(i, j)$ is undefined because (i, j) is the last element, and its "value" is the message "none" indicating the end of the enumeration. The clauses are

```
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.
```

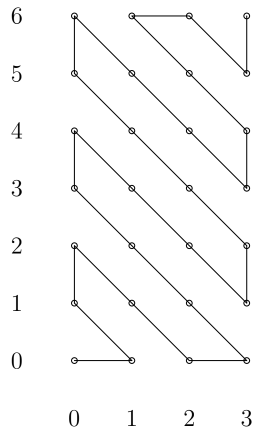

Figure 2. The diagonal enumeration of $Z^{(3,6)}$.

There are nine clauses. The conditions relative to i and j can be replaced by conditions on neighbourhoods, as in Definition 2.2, where "on the southern border" is expressed by "the neighbourhood is $\{\mathtt{N}, \mathtt{E}, \mathtt{NE}\}$ or $\{\mathtt{N}, \mathtt{E}, \mathtt{W}, \mathtt{NW}, \mathtt{NE}\}$".

Here is an alternative formalization by the automaton, which we will use again. There are nine possible types of position of a vertex (i, j) on the grid if X and Y are not singletons:[3]

- origin (numbered 0),
- on the southern border (1),
- on the western border (2),
- in the middle (3),
- at the southeastern corner (4),
- at the northwestern corner (5),
- on the northern border (6),
- on the eastern border (7) and
- at the northeastern corner (8).

See Figure 3. Each type can be determined by a combination of Boolean conditions such as "$j = 0$" and "$i \neq \max_X$" or of possible neighbourhoods. Position 3 is characterised by the maximum neighbourhood, that is, $\{\mathtt{N}, \mathtt{E}, \mathtt{W}, \mathtt{S}, \mathtt{NW}, \mathtt{NE}, \mathtt{SW}, \mathtt{SE}\} = \mathcal{D}_2$.

In Table 1, and for readability, we define an automaton $\mathcal{B}$ by using the digits 0 to 8 to indicate the types of positions instead of combinations of Boolean conditions or neighbourhoods. 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. Figure 4 shows the same information as Table 1.

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.

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

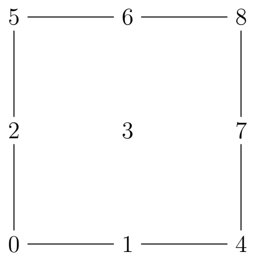

Figure 3. The different types of borders used by $\mathcal{B}$.

Proposition 2.4. *The automaton $\mathcal{B}$ defines a d2-ℓ-ordering.*

Remarks 2.5. (1) $\mathcal{B}$ is actually a $\mathcal{D}_{2\ell}$-automaton, where we define $\mathcal{D}_{2\ell} := \{$`N`, `E`, `W`, `S`, `NW`, `SE`$\}$. Position 3 is correctly determined by the neighbourhood $\{$`N`, `E`, `W`, `S`, `NW`, `SE`$\}$ and no transitions use the directions `NW` and `SE`.

(2) 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

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 $\mathcal{B}$.

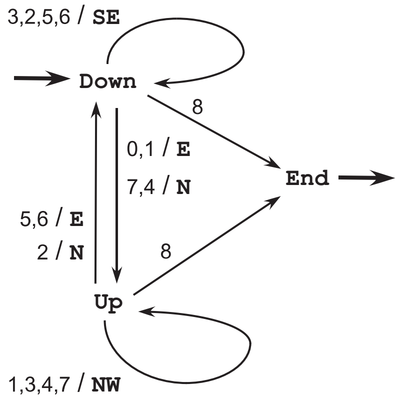

Figure 4. The automaton $\mathcal{B}$ of Proposition 2.4.

are exactly two, one starting by $(0,0) \to (1,0)$ (as in Figures 1 and 2) and the other by $(0,0) \to (0,1)$. The latter one is obtained by taking `Up` as the initial state and adding to Table 1 the transition from the origin (defined by $i = 0 \wedge j = 0$) to the state `Down` with a move north. The obtained path starts with /00/01,10/20,11,02/03, ... We will denote by $\mathcal{B}^{\#}$ this modified automaton. In automaton $\mathcal{B}$, 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.

(3) The automaton $\mathcal{B}$ (as defined by `next`, see Definition 2.3) also works in the special case where $Y = \{0\}$. All positions satisfy $j = 0 \wedge j = \max_Y$. The transitions used are defined by

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

and

$$\text{IF } i+j \text{ is odd } \wedge\ i \neq \max_X\ \wedge\ j=\max_Y \quad \text{THEN } (i+1,\, j).$$

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

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

and

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

By using $\mathcal{D}_{2\ell}$-automata, we have formalised the construction of Hamiltonian paths in the graphs $G_2(X \times Y)$, which represent d2-ℓ-orderings of $X \times Y$. We will use $\mathcal{D}_1$-automata similarly in graphs $G_1(X \times Y)$ so as to define d1-orderings. In Section 4, we will define automata that define Hamiltonian paths in $G_2(X_1 \times \cdots \times X_n)$.

In a concrete implementation, we use an oracle (a program) that determines the membership in Z of any pair $b = (i, j)$ and the set $\mathcal{D}_G(b)$, especially when Z is defined by affine conditions, such as $i \leq 3j + 5 \wedge j \leq -10i + 30$. See Section 4C.

Remark 2.6. Cantor's bijections $\mathbb{N} \times \mathbb{N} \to \mathbb{N}$ are defined by the polynomials

$$P(x,y) = x + (x+y)(x+y+1)/2 \quad \text{and} \quad P(x,y) = y + (x+y)(x+y+1)/2.$$

Fueter and Polya proved that no other quadratic polynomial defines such a bijection. References are in [Wikipedia 2015]. The enumeration by the second polynomial starts with 00/10/01/20/..., and hence does not satisfy the condition that the first components in consecutive pairs must differ by at most 1. The corresponding sequence is not a path in $G_2(\mathbb{N} \times \mathbb{N})$.

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

Proposition 3.1. *From a d1-ordering of a finite set $Y \subseteq X_1 \times X_2 \times \cdots \times X_n$, one can define a d1-ordering of $Z := \mathbb{N} \times Y$.*

Proof. D1-orderings are 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 $(i, a) \to (i, c_1) \to (i, c_2) \to \cdots \to (i, b)$, where $P_{a,b}$ is $a \to c_1 \to c_2 \to \cdots \to b$. Then, one gets in Z the infinite Hamiltonian path $0 \odot P_{a,b} \to 1 \odot P_{b,a} \to 2 \odot P_{a,b} \to 3 \odot P_{b,a} \to \cdots$ starting from $(0, \ldots, 0) = (0, a) \in Z$. (The arrow $\to$ represents the concatenation of paths). □

Remark 3.2. This construction is related to that of Gray codes; see [Wikipedia 2011]. 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 3.1 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 $\omega + \omega + \cdots = \omega \cdot \omega \neq \omega$. This is a motivation for using the diagonal d2-ℓ-ordering of Definition 2.2. However, d1-orderings can also be defined.

Proposition 3.3. (1) *There is a* d1*-ordering on* $\mathbb{N} \times \mathbb{N}$ *definable by an infinite* $\mathcal{D}_1$*-automaton.*

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

Proof. (1) See Figures 5 and 6. Theorem 3.5 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 (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 3.1 gives the answer. Otherwise, one can permute the components and write $Z = \mathbb{N} \times \cdots \times \mathbb{N} \times X_q \times \cdots \times X_p$, with $X_q, \ldots, X_p$ finite. Thus Z is isomorphic to $\mathbb{N}^q \times (X_q \times \cdots \times X_p)$ and hence to $\mathbb{N} \times Y$ with Y finite, and Proposition 3.1 gives the answer.

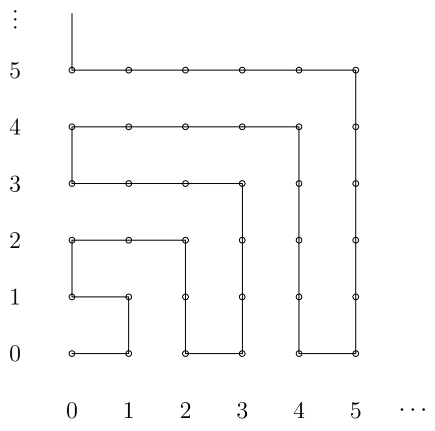

Figure 5. A d1-ordering of $\mathbb{N} \times \mathbb{N}$.

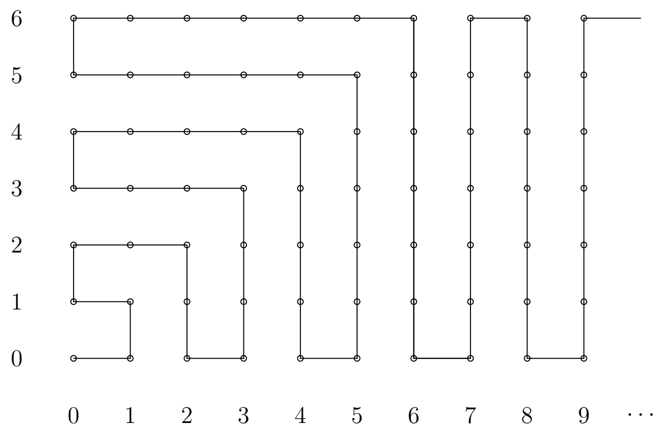

Figure 6. A d1-ordering for [0, 2n]×ℕ.

The computation of the vertex following any 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] $\square$

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.

(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 3.4. *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. To get a contradiction, we assume the existence of a $\mathcal{D}_1$-automaton $\mathcal{A} = (Q, \mathcal{T}, q_{\text{init}})$ that finds a Hamiltonian path starting at (0,0) in $G_1(X \times Y)$ for any linearly ordered sets X, Y, either ℕ or $[0, p]$.

[4] As in fly-automata, see [Courcelle and Durand 2016], 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 tests whether the current vertex is on the southwest-northeast diagonal.

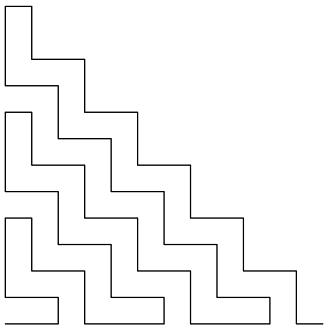

Figure 7. A d1-ordering of $\mathbb{N} \times \mathbb{N}$ that is adaptable to $X \times Y$, where X and/or Y is finite.

This automaton uses only the directions `N`, `E`, `S`, `W`. The sets X and Y are finite or infinite, which the automaton "does not know": this 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: $\mathcal{D}_G(x) = \{\mathtt{N}, \mathtt{E}\}$,
or on the southern border, and not the origin: $\mathcal{D}_G(x) = \{\mathtt{N}, \mathtt{E}, \mathtt{W}\}$,
or on the western border, and not the origin: $\mathcal{D}_G(x) = \{\mathtt{N}, \mathtt{E}, \mathtt{S}\}$,
or in the middle: $\mathcal{D}_G(x) = \{\mathtt{N}, \mathtt{E}, \mathtt{S}, \mathtt{W}\}$.

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 southern border and some b on the western border, such that all intermediate vertices x have neighbourhood $\mathcal{D}_G(x) = \{\mathtt{N}, \mathtt{E}, \mathtt{S}, \mathtt{W}\}$. 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) determined by $P[a, b]$ and the western and southern 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 $(0, b)$.

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 east and/or two moves north. That is, we enlarge neighbourhoods. We extend accordingly the definitions of Section 1 and define the set of checkable directions around a vertex as $\mathcal{E} := \{\mathtt{N}, \mathtt{NN}, \mathtt{E}, \mathtt{EE}, \mathtt{S}, \mathtt{W}\}$. If in $\mathcal{D}_G(x)$ we have `EE` (two consecutive east moves are possible), 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 from the eastern border. Similar facts hold for `NN` and `N`. The corresponding path is in Figure 7.

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 $\mathbb{N} \times \mathbb{N}$; see 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

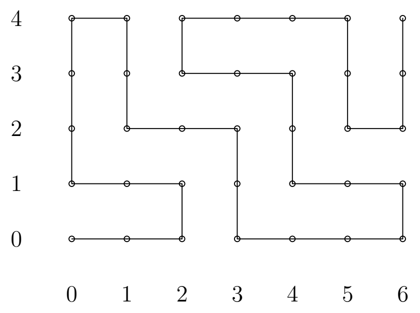

Figure 8. The basic case of $X \times Y$ with sets X, Y of odd cardinalities.

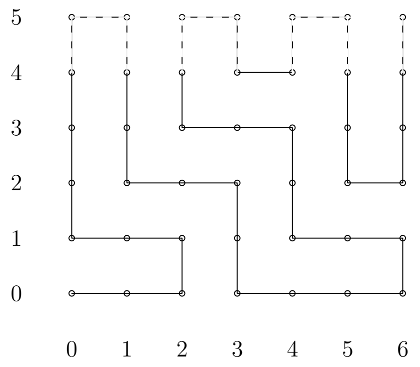

Figure 9. Extended the previous order to accommodate set Y of even cardinality.

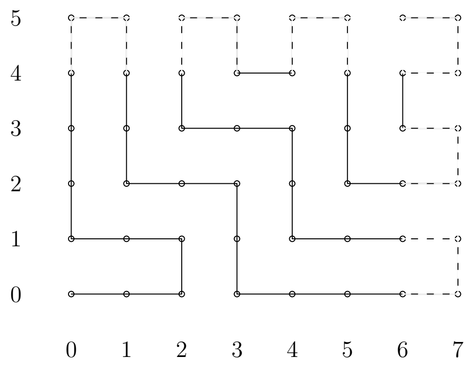

Figure 10. Extended the previous order to accommodate sets X, Y of even cardinality.

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 3.5. *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

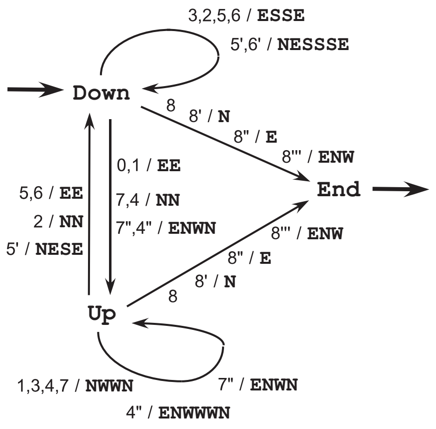

Figure 11. The $\mathcal{E}$-automaton of Theorem 3.5.

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 $\mathcal{C}$.

cardinality 1 if and only if `E` is not 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 $\mathcal{D}_{2\ell}$-automaton $\mathcal{B}$ of Proposition 2.4.

From $\mathcal{B}$, we first define a $\mathcal{D}_{2\ell}$-automaton $\mathcal{B}'$ by duplicating actions: `N` becomes `N;N`, that is two moves north, `SE` becomes `SE;SE`, etc. Because of the assumptions on the cardinalities of X and Y, from each vertex reached by a path with an even number of edges that is defined by $\mathcal{B}'$, if one can make one step east, one can make another one to the east, and similarly for north, southeast and northwest. This automaton need not check the "double" directions `EE` and `NN`.

A $\mathcal{D}_1$-automaton $\mathcal{C}$ is defined by Table 2, is obtained from $\mathcal{B}'$ by replacing respectively the "double" moves `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 $\mathcal{B}'$; see Figures 3 and 12. As for $\mathcal{B}'$, one need not check the "double" 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 of claim. First we prove that $\mathcal{C}$ defines a path, i.e., a walk that does not visit the same vertex twice.

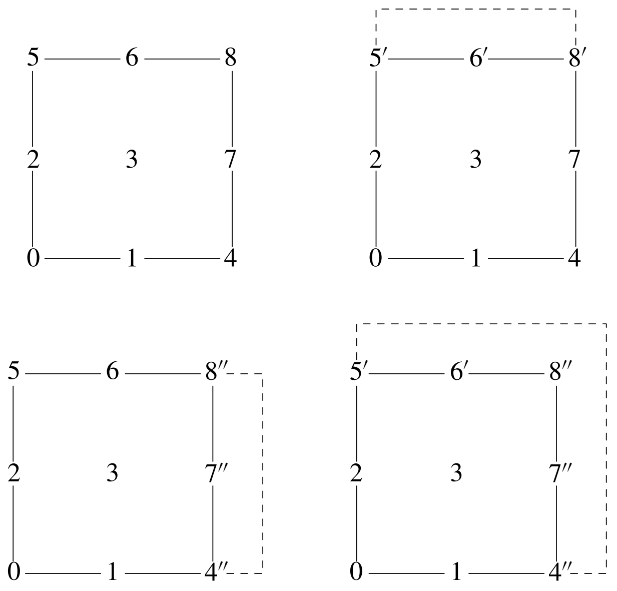

Figure 12. Numbering of types of positions relative to the borders. In particular, close to the northern and eastern borders.

A $2{\times}2$ *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 coloured black or white, so that two adjacent squares (adjacent by a border, not just a corner) are of different colour. Let $[0, 2] \times [0, 2]$ be white. By $\mathcal{B}'$, it is traversed by the "double" moves `NW`; `NW`. Those of the form `SE;SE` traverse black $2{\times}2$ squares.

When defining $\mathcal{C}$, we replace `SE;SE` by `E;S;S;E`, so that we go through two more vertices, say x and y, in the middle of the top and bottom borders of that $2{\times}2$ square. In the surrounding $2{\times}2$ 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 colouring of the $2{\times}2$ 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 $8'$ (see Figure 12) characterised by the following conditions relative to a vertex x:

$$\begin{aligned}
5' &: \texttt{N,EE,S} \in \mathcal{D}_G(x), & \texttt{W,NN} &\notin \mathcal{D}_G(x),\\
6' &: \texttt{N,EE,W,S} \in \mathcal{D}_G(x), & \texttt{NN} &\notin \mathcal{D}_G(x),\\
8' &: \texttt{N,EE,W,S} \in \mathcal{D}_G(x), & \texttt{E,NN} &\notin \mathcal{D}_G(x).
\end{aligned}$$

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

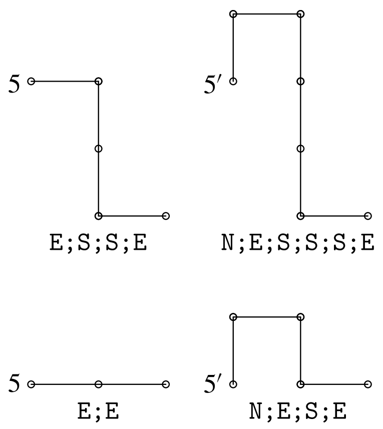

Figure 13. Some detours for vertices close to the northern border.

In order to reach the vertices on the top row, we make small detours defined as follows. When 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 the top part of Figure 13).
- From vertex $8'$, action is `N`, terminating the path.

When 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 $8''$, characterised by the following conditions:

$$\begin{array}{lll} 4'': & \mathtt{N,E,W} \in \mathcal{D}_G(x), & \mathtt{S,EE} \notin \mathcal{D}_G(x), \\ 7'': & \mathtt{NN,E,S,W} \in \mathcal{D}_G(x), & \mathtt{EE} \notin \mathcal{D}_G(x), \\ 8'': & \mathtt{E,S,W} \in \mathcal{D}_G(x), & \mathtt{N,EE} \notin \mathcal{D}_G(x). \end{array}$$

If Y is infinite, positions of types 5, 6, $8''$ do not occur.
When 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 14).
- 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.

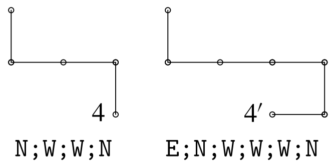

Figure 14. The detour at southeastern corner.

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 3.5.

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 $4''$, $5'$, $6'$, $7''$ characterised as above and

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

From a vertex of type $8''$, the action is E;N;W, terminating the path.

These definitions are collected in Table 3. □

Remark 3.6. 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 $p > 2$.

4. Diagonal orderings of subsets of $\mathbb{N} \times \mathbb{N}$

Figure 15 shows that the automaton $\mathcal{B}$ of Section 2 can order proper subsets of $\mathbb{N} \times \mathbb{N}$ that are not Cartesian products. In this section, we develop this observation.

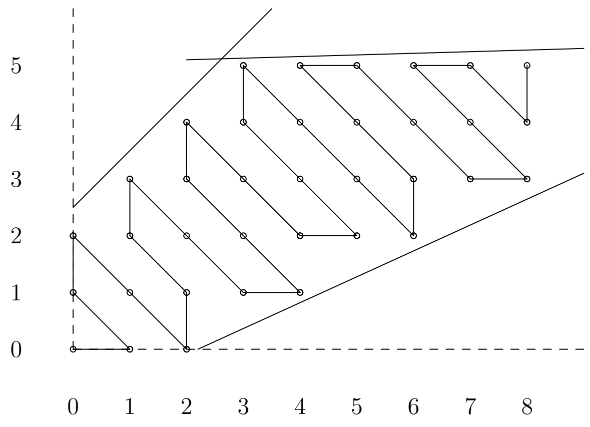

Figure 15. A proper subset of a Cartesian product ordered by automaton $\mathcal{B}$ of Section 2.

4A. *D2-ℓ-orderings of subsets of* $\mathbb{N} \times \mathbb{N}$.

We ask the following questions.

Question 4.1. Which subsets Z of $\mathbb{N} \times \mathbb{N}$ have a d2-ℓ-ordering?

We will consider $\mathcal{D}_2$-automata, more powerful than $\mathcal{D}_{2\ell}$-automata as they can move northeast in addition to northwest, north, east, southwest and south. As we want them to define d2-ℓ-*paths*, i.e., Hamiltonian paths corresponding to d2-ℓ-orderings, they will make no moves southwest.

Question 4.2. When is a d2-ℓ-ordering definable by a finite or infinite $\mathcal{D}_2$-automaton?

If Z is finite and d2-ℓ-orderable, then such an ordering is definable by a finite $\mathcal{D}_2$-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 4.3 (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 $G_2(Z)$, and hence, induces a north-west-southeast 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 $\texttt{Last}(Z_k)$ is adjacent to $\texttt{First}(Z_{k'})$ in $G_2(Z)$, where[6]

- if Z_k is labelled by `Down`, then $\texttt{First}(Z_k) := \max(Z_k)$ and $\texttt{Last}(Z_k) := \min(Z_k)$ and
- if Z_k is labelled by `Up`, then $\texttt{First}(Z_k) := \min(Z_k)$ and $\texttt{Last}(Z_k) := \max(Z_k)$.

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 $G_2(Z)$ and 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 4.4. 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.

[6]These definitions are the same for $Z_{k'}$.

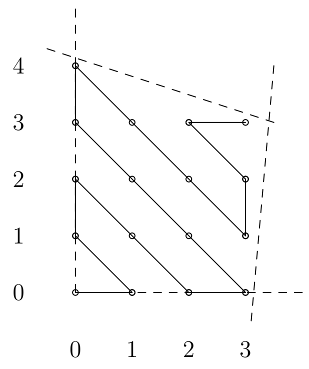

Figure 16. The set W of Example 4.6(1) used in Proposition 4.7.

Proposition 4.5. *A subset of* $\mathbb{N} \times \mathbb{N}$ *containing* $(0, 0)$ *has a d2-ℓ-ordering if and only if it satisfies conditions* (C1), (C2) *and* (C3).

Proof. Let Z be a subset of $\mathbb{N} \times \mathbb{N}$ that has a d2-ℓ-ordering with associated path P in $G_2(Z)$. This set satisfies conditions (C1) and (C2). We label by $\mathtt{Up}$ a nonempty level that is traversed upwards by P, that is from southeast to northwest, and by $\mathtt{Down}$ in the other case. We label arbitrarily a singleton level $Z_k = \{x\}$ and in either case we have $\mathtt{Last}(Z_k) = \mathtt{First}(Z_k) = x$. Then condition (C3) holds with this labelling.

Conversely, let Z satisfy (C1) and (C2). From any labelling satisfying (C3), we obtain a path P from $(0, 0)$ with appropriate transitions between levels, which describes a d2-ℓ-ordering of Z. $\square$

Example 4.6. (1) Figure 16 shows an example of a set $W \subseteq \mathbb{N} \times \mathbb{N}$ that satisfies conditions (C1)–(C3). It has a d2-ℓ-path starting with $(0, 0)\rightarrow(1, 0)$, shown in this figure. An initial step $(0, 0)\rightarrow(0, 1)$ 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).

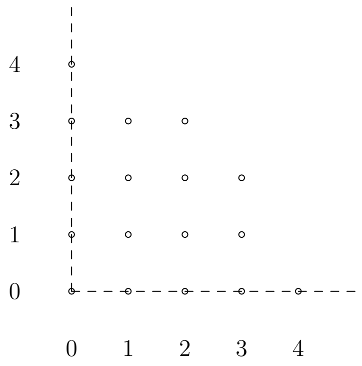

Figure 17. Set X of Example 4.6(2).

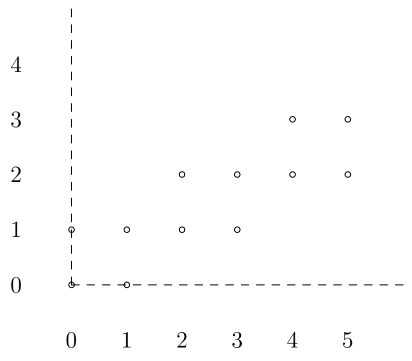

Figure 18. Set Y of Example 4.6(3).

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

$$\begin{array}{ll} 00/10,\, 01/11/21/31,\, 22/32/42/52,\, 43/53, & \text{defined by } \mathcal{B}, \\ 00/01,\, 10/11/21/22,\, 31/32/42/43,\, 52/53, & \text{defined by } \mathcal{B}^{\#}, \text{ and} \\ 00/10,\, 01/11/21/22,\, 31/32/42/52,\, 43/53, & \text{defined neither by } \mathcal{B} \text{ nor by } \mathcal{B}^{\#}. \end{array}$$

Its infinite extension Y' defined by $i/2-1/2 \leq j \leq i/2+1$ has infinitely many d2-ℓ-orderings. □

We continue the study of sets $Z \subseteq \mathbb{N} \times \mathbb{N}$. If $G_1(Z)$ is connected and Z has a d2-ℓ-ordering that is definable by a $\mathcal{D}_2$-automaton, then this ordering is definable by a $\mathcal{D}_{2\ell}$-automaton, actually the same, because no move northeast can be used.

We recall that automata are deterministic and must have computable transitions; see Definition 1.2(a).

Proposition 4.7. (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 $\mathcal{D}_2$-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 $\mathcal{D}_2$-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 southwest-northeast 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 a horizontal edge between 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 $\mathcal{D}_2$-automaton (equivalently, a $\mathcal{D}_{2\ell}$-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 north or 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 $\mathcal{D}_2$-automaton, by an argument similar to that used in (1). It is definable by an infinite one (whose transitions must be computable, see Definition 1.2(a)) if there exists a computable function $f_w \colon \{0,1\}^* \to \{0,1\}$ such that $f_w(u)$ defines the

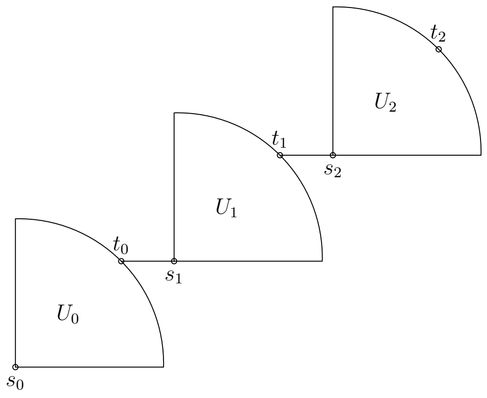

Figure 19. Set $X^{(2)}$ of Proposition 4.7.

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 exist uncountably many words w in $\{0, 1\}^\omega$ such that f_w is not computable, and hence uncountably many sets X of the above form with unique d2-ℓ-orderings that are not definable by any $\mathcal{D}_2$–automaton. □

One might consider more powerful automata whose transitions from a vertex x are determined from the state and the neighbourhood of x consisting of vertices at distance at most some p. A similar proof can be done with sets similar to W, of diameter larger than p.

4B. ***A $\mathcal{D}_2$-automaton extending $\mathcal{B}$.*** We define a $\mathcal{D}_2$-automaton $\mathcal{F}$ that extends $\mathcal{B}$ and is 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.

Example 4.8. (1) Let Z be defined by $2i/3 \le j \le 3i/2$. Its first levels are $\{(0, 0)\}$, $\varnothing$, $\{(1, 1)\}$, $\varnothing$, $\{(2, 2)\}$, $\{(2, 3), (3, 2)\}$. A d2-ℓ-ordering can be defined by $\mathcal{F}$ that makes northeast moves $(0, 0) \to (1, 1) \to (2, 2)$ and then continues with the transition rules of $\mathcal{B}$.

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 $\mathcal{D}_2$-automaton $\mathcal{F}$.

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

Our next aim is to characterise the sets $Z \subseteq \mathbb{N} \times \mathbb{N}$ that are d2-ℓ-ordered by the $\mathcal{D}_2$-automaton $\mathcal{F}$ and the $\mathcal{D}_{2\ell}$-automaton $\mathcal{B}$ of Section 4B.

In the following definition, we use the notation of Definition 4.3.

Definition 4.9 (other 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 `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` (resp. `Up`) then, it is labelled by `Up` (resp. `Down`).

(C4.2) If $Z_{k'}$ follows Z_k, then $\texttt{Last}(Z_k)$ is adjacent to $\texttt{First}(Z_{k'})$ in $G_2(Z)$.

If $G_1(Z)$ is connected, we have $k' = k+1$ in conditions (C4.1) and (C4.2).

Example 4.4 satisfies conditions (C1)–(C3) but not condition (C4). By condition (C4), the labelling of nonempty levels is defined in a unique way, because we want to characterise the existence of deterministic automata in the next theorem. Conditions (C1), (C2) and (C4) imply condition (C3).

Theorem 4.10. *Let $Z \subseteq \mathbb{N} \times \mathbb{N}$. It is d2-$\ell$-ordered by automaton $\mathcal{F}$ if and only if it satisfies conditions (C1), (C2) and (C4). It is d2-ℓ-ordered by automaton $\mathcal{B}$ if and only if $G_1(Z)$ is connected (which implies (C1)) and Z satisfies conditions (C2) and (C4).*

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

If $G_1(Z)$ is connected, then, the automata $\mathcal{F}$ and $\mathcal{B}$ order Z by traversing the levels in the order $Z_0, Z_1, \ldots$. 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 the nonempty 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 north and 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 north or east. These transitions prove condition (C4).

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

4C. *Sets that satisfy conditions* (C1)–(C4).

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

Definition 4.11 (affine subsets of $\mathbb{N} \times \mathbb{N}$). We call *affine*[7] a subset Z of $\mathbb{N} \times \mathbb{N}$ defined by the conjunction of finitely many conditions of the following forms, intended to specify that $(i, j) \in Z$:

(i) $i \leq a$,

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

(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 design algorithms for deciding properties of a given affine set Z that are listed below. Each level Z_k can be enumerated in a straightforward brute force manner.

[7] A more general definition of an affine set could be obtained by omitting the nonnegativity conditions on a, c, d and e.

Questions 4.12. (1) Is a given affine set Z finite?

(2) Is $G_2(Z)$ connected? Is $G_1(Z)$ connected?

(3) Are conditions (C1)–(C3) satisfied?

(4) If they are, does there exist an automaton[8] that defines a d2-ℓ-ordering?

The following examples show a variety of cases.

Example 4.13. We let Z be defined by the following conditions:

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

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

(3) $i \le j \le i$. Then $G_2(Z)$ is an infinite diagonal southwest-northeast path and $G_1(Z)$ has no edge.

(4) $(4i-1)/10 \le j \le i/2$. 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 4.14 (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, and 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 $0 < k' < k$, which is equivalent to condition (C2).

We will also use the following two notions:

(*knight convexity*)[9] (H) If $(i, j+1)$ and $(i+2, j) \in Z$, then $(i+1, j) \in Z$.
(V) If $(i, j+2)$ and $(i+1, j) \in Z$, then $(i, j+1) \in Z$.

An affine set Z defined by a inequality of type (i) or (iii) satisfies condition (H); if it is defined by an inequality of type (i) or (ii) with $b \ge 0$, it satisfies condition (V). Diagonal and knight convexities are also preserved by intersection.

Theorem 4.15. *One can decide if an affine subset of $\mathbb{N} \times \mathbb{N}$ is finite and if it satisfies conditions (C1), (C3) and (C4).*

Lemma 4.16. (1) *An affine subset is finite if its description contains, among other inequalities, one of the form $j \le bi + c$ with $b < 0$ or two of the following forms: $i \le a$ and $j \le bi + c$ with $b \ge 0$, or $j \le bi + c$ and $j \ge di - e$ with $d > b > 0$. If so, one can enumerate it and check conditions (C1), (C3) and (C4).*

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

[9]Named by reference to the move of knights in chess.

(2) *An affine subset defined Z by $bi - e \leq j \leq bi + c$ with $b, c, e \geq 0$ is empty or infinite. This can be decided.*

Proof. (1) The listed inequalities imply finiteness. In each case, the set Z can be enumerated level by level. One can then check whether it satisfies conditions (C1), (C3) and (C4).

(2) Let $b = r/q$. A pair (i, j) of nonnegative integers is in Z if and only if $(i+q, j+r)$ is. The result follows by elementary arithmetic. □

In the description of an affine set, we can eliminate an inequality $i \leq a'$ if we already have $i \leq a$ with $a < a'$ in the description, and similarly for the inequalities of types (ii) and (iii) (of Definition 4.11). After these eliminations, we obtain a *nonredundant description* (although possibly not minimal).

Example 4.17. The following affine sets are infinite:

(1) The set Z defined by $j \geq i/2$, $j \geq i-2$ and $i \leq 7$ (conditions of type (i) and (iii)).

(2) The set Z' defined by $i/2 \leq j \leq i$ (conditions of type (ii) and (iii)). The level Z'_1 is empty but $G_1(Z' - Z'_0)$ is connected.

(3) The set Z'' defined by $i = j$ (hence by conditions of type (ii) and (iii)). The graph $G_1(Z'')$ has no edge but $G_2(Z'')$ is connected. □

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 $p \leq k < q$.

Lemma 4.18. *Let Z be an affine 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 for $Z_{[p,\infty[}$.*

(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) This is clear since $Z_{[0,p+1[} \cap Z_{[p,\infty[} = Z_p \neq \emptyset$. The same holds for $G_2(Z)$, i.e, for condition (C1).

(2) This is clear from (1) and the definition of (C3).

(3) This is 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 $Z_{[p,\infty[}$, have same label for $Z_{[0,p+1[}$ and $Z_{[p,\infty[}$. □

Proof of Theorem 4.15. Let Z be an affine subset of $\mathbb{N} \times \mathbb{N}$ defined by a nonredundant description. We have the following five cases, where each one excludes the previous ones. They cover all descriptions of affine sets.

Case 1: We are in the cases covered by Lemma 4.16(1); hence Z is finite and this lemma yields the result.

Case 2: The description of Z consists of inequalities of types (i) and (iii). Then Z is infinite, $G_1(Z)$ is connected and Z satisfies knight convexity of type (H). All levels are nonempty, and one can label them according to (C4.0) and (C4.1). We check condition (C4.2). Let Z_k be labelled by `Down`. Then

$\mathtt{Last}(Z_k) = \min(Z_k) = (i, j)$ is adjacent to $(i, j+1)$ of height $k+1$. By condition (H) we cannot have $(i+2, j-1) \in Z_{k+1}$. Hence $\mathtt{First}(Z_{k+1}) = \min(Z_{k+1})$ is $(i, j+1)$ or $(i+1, j)$ and thus is adjacent to $\mathtt{Last}(Z_k)$. If Z_k is labelled by $\mathtt{Up}$, then $\mathtt{Last}(Z_k) = \max(Z_k) = (0, j)$ and $\mathtt{First}(Z_{k+1}) = \max(Z_{k+1}) = (0, j+1)$ is adjacent to it. The set Z of Example 4.17(1) is of this type.

Case 3: The description of Z consists of inequalities of types (ii) with $b \geq 0$. Then Z is infinite, $G_1(Z)$ is connected and Z satisfies knight convexity of type (V). The proof is similar to that of Case 2.

For the next cases, we define $I(Z)$ as the set 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 $I(Z)$.

Case 4: 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 $\bar{b}$ be the minimal such b and $\bar{d}$ be the maximal such d, Let $\bar{c}$ and $\bar{e}$ be the corresponding constant coefficients.

We first consider W defined by the inequalities $j \leq \bar{b}i + \bar{c}$ and $j \geq \bar{d}i - \bar{e}$ and we let $r, q \in \mathbb{N}$ be such that $\bar{b} > r/q > \bar{d}$. Let i, j be such that $i \geq (\bar{d}q - \bar{e} - \bar{c})/(\bar{b} - \bar{d})$ and $\bar{d}i + \bar{d}q - \bar{e} \leq j \leq \bar{b}i + \bar{c}$. We have in W the vertices (i, j), $(i+q, j)$ and $(i+q, j+r)$, as one checks easily. Hence, by horizontal and vertical convexities, we have 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 again, 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. It satisfies conditions (C4.1) and (C4.2), where Z_p can be labelled $\mathtt{Up}$ or $\mathtt{Down}$. For proving (C4.2), we use knight convexities, as in Cases 2 and 3. As $Z_{[0,p+1[}$ is finite, we can enumerate it and use Lemma 4.18 to decide if Z satisfies conditions (C1), (C3) and (C4).

Case 5: The set 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 defined by these two inequalities (we may have $b = e = 0$ and, necessarily, $c \geq 1$). By Lemma 4.16(2), we can identify the case where W, whence Z, is empty. Assume now W is infinite. 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 than 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 4.13(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). The intersection of these two sets is Z_{p+r+q}. Hence, $Z_{[p,\infty[}$ is the union of the pairwise isomorphic sets $Z_{[p+n(r+q),p+(n+1)(r+q)+1[}$.

Then we have that $G_1(Z_{[p,\infty[})$ (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.

Thus $Z_{[p,\infty[}$ satisfies (C3) (resp. (C4)) if and only if $Z_{[p,p+r+q+1[}$ does, which is decidable. As $Z_{[0,p+1[}$ is finite, we can decide it satisfies conditions (C1), (C3) and (C4), and we obtain the final result by means of Lemma 4.18. ☐

Open question: Is there an efficient algorithm for the decision problem of Theorem 4.15?

5. Dimension > 2

We consider next the definition of d2-ℓ-orderings of sets $X_1 \times X_2 \times \cdots \times X_p$, where $X_1, \dots, X_p$ are finite or infinite linearly ordered sets, equivalently, $[0, m]$ or $\mathbb{N}$. We will prove that a unique automaton[10] 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 $X_q \times \cdots \times X_p$.

5A. ***Levels.*** We generalise the notion of level from Definition 2.1. We define it abstractly in a linearly ordered set. The notion of height will arise from that of level.

Definition 5.1 (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 $Z_0, Z_1, \dots, Z_n, \dots$ such that $Z_0 < Z_1 < \cdots < Z_n < \cdots$ Each interval is called a *level*. If $m \in Z_j$, then $ht(m) := j$ is the *height* of m.

We define $\mathrm{Lev}(Z)$ 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 an llo Z is the llo on the set $U := X \times Z$ defined as in Definition 2.2, with a notion of type, which 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 $X \times \mathrm{Lev}(Z)$, ordered as in Definition 2.3, the interval $i \odot Z_j$ to (i, j), where $i \odot (s_0, \dots, s_q)$ denotes the linear order $((x, s_0), (x, s_1), (x, s_2), \dots, (x, s_q))$.

Example 5.2. (1) An example of an llo is $Z^{(3,6)} := [0, 3] \times [0, 6]$ of Definition 2.3, which we can describe as

$$/00/10, 01/02, 11, 20/30, 21, 12, 03/ \dots /26, 35/36/$$

by separating levels. See Figure 2. Every linear order from Definition 2.3 is an llo, as a notion of level is defined.

(2) From a linear order X and an integer p, we can define an llo where each level has p elements, except the last one, which may have less. For $X := \mathbb{N}$ and $p = 3$, we get the llo $/0, 1, 2/3, 4, 5/6, 7, 8/\dots$.

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

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

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

[10]It is a graph-walking automaton traversing the graph $G_2(Z)$ but its description will not use directions.

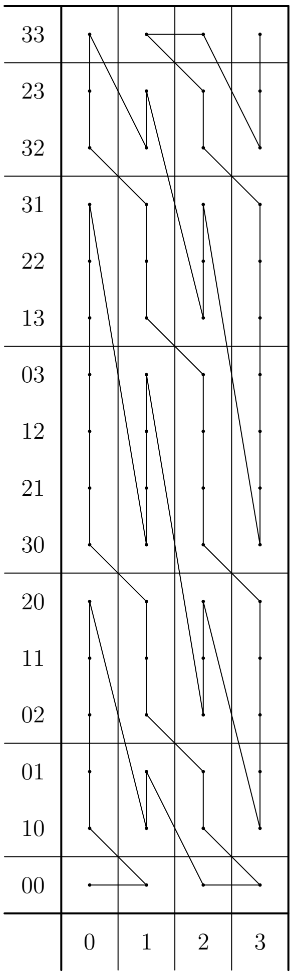

Figure 20. The d2-ℓ-ordering of Example 5.2(5).

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

$$/000/100, 010, 001/002, 011, 020, 110, 101, 200/300, 210, 201,$$
$$102, 111, 120, 030, 021, 012, 003/013, 022, 031, 130, \dots,$$
$$301/302, 311, 320, 230, \dots, 023/\dots/233, 332, 323/333/.$$

For conciseness, we write, e.g., 121 instead of (1, 21), a notation that we use below.

Observation 5.3 (concrete descriptions of the levels of $X \times Z$). (1) If X and Z are infinite, the levels U_i of the llo on $U := X \times Z$ of Definition 5.1 are as follows (where $\bullet$ denotes concatenation of sequences):

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

We can determine as follows 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 with subcases:

(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. There are two subcases:

(b0) 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} (see the definition of $\mathtt{back}_Z$ below).

(b1) 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}.

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

(c0) 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).

(c1) 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 visualise case (c) as follows:

If i is even, then

$$U_i \bullet U_{i+1} = (0 \odot Z_i) \bullet \cdots \bullet (i \odot Z_0) \bullet ((i + 1) \odot Z_0) \bullet \cdots \bullet (0 \odot Z_{i+1}),$$

and if i is odd, then

$$U_i \bullet U_{i+1} = (i \odot Z_0) \bullet \cdots \bullet (0 \odot Z_i) \bullet (0 \odot Z_{i+1}) \bullet \cdots \bullet ((i + 1) \odot Z_0).$$

In case (b0) 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 (c0) 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 $Z = X_q \times \cdots \times X_p$.

(2) 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 $\mathrm{Lev}(Z)$. We let $k \leq m_X + M_Z$ ($U_{m_X+M_Z}$ 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

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

If k is odd, we have

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

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.

(c′0) If k is even, then

$$U_k \bullet U_{k+1} = (\alpha \odot Z_{k-\alpha}) \bullet \cdots \bullet ((k-\beta) \odot Z_\beta) \bullet ((k+1-\beta') \odot Z_{\beta'}) \bullet \cdots \bullet (\alpha' \odot Z_{k+1-\alpha}),$$

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 subcases 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 $Z_0 = Z_\beta$. For example, $(x, z) = (2, 00)$, $(x', z') = (3, 00)$.

(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 that z is in Z. There is no example because $m_X = 3$.

(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 that z is in Z. For example, $(x, z) = (3, 01)$, $(x', z') = (3, 02)$.

In case (i) we have a transition $z \to z'$ inside Z_0.

(c′1) Similarly, if k is odd, then

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

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

(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 that z is in Z. For example, $(x, z) = (0, 23)$, $(x', z') = (0, 33)$.

(v) If $k = M_Z$, then $\alpha = 0 = x$, $\alpha' = 1 = x'$, and z' is the first element in $Z_k = Z_{M_Z}$. There is no example because $M_Z = 6$.

(vi) If $k > M_Z$, then $\alpha = x = k - M_Z$, $\alpha' = 1 = x'$, and z' is the first element in Z_{M_Z}. For example, $(x', z') = (1, 23)$, $(x', z') = (0, 33)$.

In cases (v), (vi), we have a transition $z \to z'$ inside Z_{M_Z}.

Definition 5.4 (back steps from last elements in their levels). (a) If Z is an 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 5.5 (back steps in Cartesian products). Let $Z := [0, 3] \times [0, 3]$, $U := [0, 3] \times Z$ and consider in Figure 20 the level

$$U_4 = \big((0, 13), (0, 22), (0, 31), (1, 30), (1, 21), (1, 12), (1, 03), (2, 02), (2, 11), (2, 20), (3, 10), (3, 01)\big).$$

We write ij for a pair (i, j) of Z and (k, ij) for 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 $(0, 31) \to (1, 30)$ based on a back step in Z from 31 to 30 that decreases height in Z. In this llo, some other $\texttt{back}_Z$ transitions are used for ordering U: $20 \to 10$ (for $(0, 20) \to (1, 10)$), $33 \to 32$ (for $(2, 33) \to (3, 32)$), $23 \to 13$ (for $(1, 23) \to (2, 13)$) and $33 \to 32$ (for $(2, 33) \to (3, 32)$).

Construction 5.6. We define an automaton $\mathcal{A}_U$ intended to define the llo on $U := X \times Z$ (see Definition 5.1), where X and Z are llo's that satisfy the following conditions:

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

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

For X, $\texttt{Lil}_X(x)$ is always true, and $\texttt{back}_X$ is nothing but $\texttt{prev}_X$. We will use 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. We will use Boolean conditions on X and Z instead. 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 these notations for uniformity with those for Z that cannot be easily expressed from integers. (However, see Proposition 5.8 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.

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

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

We define

$$\texttt{first}_U := \texttt{first}_X \wedge \texttt{first}_Z \quad\text{and}\quad \texttt{last}_U := \texttt{last}_X \wedge \texttt{last}_Z.$$

and $\texttt{Lil}_U$ is defined by Table 5. In Tables 5–8, "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 $\texttt{back}_Z$. It replaces the directions and border conditions used in the automata of Sections 2, 3 and 4. The examples concern $U := [0, 3] \times Z$, where $Z := [0, 3] \times [0, 3]$, and the d2-$\ell$-ordering shown in Figure 20. The actions $\texttt{next}_U$ and $\texttt{back}_U$ are described in Tables 6–8. In Tables 6 and 7, "property" indicates if $\texttt{Lil}_U$ holds before the transition is done. The indicated cases (a), (b), (i), (ii) etc. refer to Observation 5.3.

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

5B. *Application to Cartesian products.*

Theorem 5.7. *There is an automaton with 2^{n-1}states that defines a d2-ℓ-ordering of $U = X_1 \times X_2 \times \cdots \times X_n$, where $X_1, X_2, \ldots, X_n$ are finite or infinite linearly ordered sets.*

Proof. We use Construction 5.6 recursively by writing $U = X_1 \times Z = X_1 \times (X_2 \times (\cdots \times X_n))$. To have a d2-ordering, we must check that the distance between consecutive elements is at most 2.

We let $P(Z)$ for an llo Z (intended to be $X_i \times (\cdots \times X_n)$ for some i) be the conjunction of the following properties:

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. Cases where $\texttt{Lil}_U$ is true.

conditions	subcondition	property	action	new state	examples; cases
$\neg\texttt{Lil}_Z$		$\neg\texttt{Lil}_U$	$\texttt{next}_Z$	Down	(1, 21)→(1, 12)(a)
$\texttt{Lil}_Z$	$\neg\texttt{last}_X\wedge\neg\texttt{first}_Z$	$\neg\texttt{Lil}_U$	$\texttt{next}_X$; $\texttt{back}_Z$	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. $\texttt{next}_U$ for state=Down.

condition	subconditions	property	action	new state	examples; cases
$\neg\texttt{Lil}_Z$		$\neg\texttt{Lil}_U$	$\texttt{next}_Z$	Up	(0, 21)→(0, 12)(a)
$\texttt{Lil}_Z$	$\neg\texttt{first}_X\wedge\neg\texttt{last}_Z$	$\neg\texttt{Lil}_U$	$\texttt{prev}_X$; $\texttt{next}_Z$	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. $\texttt{next}_U$ for state=Up.

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	$\texttt{first}_X\wedge\texttt{Lil}_Z$	$\texttt{back}_Z$	Down	(0, 03)→(0, 02)
Up	$\neg\texttt{first}_X\wedge\texttt{last}_Z$	$\texttt{prev}_X$	Down	(1, 33)→(0, 33)

Table 8. $\texttt{back}_U$.

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

(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, in particular for $Z = X_n \subseteq \mathbb{N}$. Then $\texttt{Lil}_Z$ always holds and $\texttt{back}_Z$ is $\texttt{prev}_Z$.

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

(a) We have this by the definitions.

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

(c) Assume that $\mathtt{Lil}_U$ holds. From Table 5, we have $\mathtt{Lil}_Z$ in all cases, because $\mathtt{last}_Z$ implies $\mathtt{Lil}_Z$ and so does $\mathtt{first}_Z$ because Z_0 is singleton (by (a)).

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

(d) This is clear from Tables 6 and 7. ☐

Condition (a) is necessary for Construction 5.6 to work.

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

Direct computation of 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 $(3, 0, 2, 4, 0, 0)$ to take an example, without having to enumerate U until one reaches the given tuple and the one following it.

Proposition 5.8. *Let $U := X_1 \times X_2 \times \cdots \times X_n$ such that the values m_i are known. There exists an algorithm that, for input $\boldsymbol{x} = (x_1, \ldots, x_n)$ 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 $\mathtt{next}_U((x_1, \ldots, x_n))$ by means of at most n auxiliary computations of

$$\mathtt{Lil}_{X_i \times \cdots \times X_n}((y_i, \ldots, y_n)),$$
$$\mathtt{next}_{X_i \times \cdots \times X_n}((y_i, \ldots, y_n)),$$
$$\mathtt{back}_{X_i \times \cdots \times X_n}((y_i, \ldots, y_n)))$$

for $1 \leq i \leq n$ and appropriate tuples $(y_i, \ldots, y_n)$.

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

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

$$\Lambda_i(y_i, \ldots, y_n) := \big(\mathtt{Lil}_i(y_i, \ldots, y_n), \mathtt{next}_i(y_i, \ldots, y_n), \mathtt{back}_i(y_i, \ldots, y_n)\big),$$

where $\mathtt{Lil}_i(y_i, \ldots, y_n) \in \{\text{true}, \text{false}\}$, $\mathtt{next}_i(y_i, \ldots, y_n)$ is $\perp$ (undefined) if $(y_i, \ldots, y_n) = (m_i, \ldots, m_n)$ and is in $X_i \times \cdots \times X_n$ otherwise, $\mathtt{back}_i(y_i, \ldots, y_n)$ is $\perp$ if $\mathtt{Lil}_i(y_i, \ldots, y_n) = \text{false}$ or $(y_i, \ldots, y_n) = (0, \ldots, 0)$, and it is in $X_i \times \cdots \times X_n$ otherwise.

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

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

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

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

For computing $\mathtt{Lil}_i(y_i, \dots, y_n)$ (see Table 5), we use

$$\begin{array}{ll} (y_i = 0)? & \text{for } \mathtt{first}_X, \\ (y_i = m_i)? & \text{for } \mathtt{last} \ \ \mathtt{Lil}_{i+1}(y_{i+1}, \dots, y_n) \quad \text{for } \mathtt{Lil}_Z \text{ and} \\ ((y_{i+1}, \dots, y_n) = (m_{i+1}, \dots, m_n))? & \text{for } \mathtt{last}_Z. \end{array}$$

For computing $\mathtt{next}_i(y_i, \dots, y_n)$ and $\mathtt{back}_i(y_i, \dots, y_n)$ (see Tables 6, 7 and 8), we use

$$\begin{array}{ll} ((y_{i+1}, \dots, y_n) = (0, \dots, 0))? & \text{for } \mathtt{first}_Z, \\ \mathtt{next}_{i+1}(y_{i+1}, \dots, y_n) & \text{for } \mathtt{next}_Z, \\ \mathtt{back}_{i+1}(y_{i+1}, \dots, y_n) & \text{for } \mathtt{back}_Z \end{array}$$

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

Example 5.9. Here are some particular cases and examples:

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

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

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

For $\boldsymbol{x} = 302400$, the state is $\mathtt{Up}$, $\mathtt{Lil}_6(\boldsymbol{x}) = \text{false}$ (see Table 5: 3 is not first in X_1 and 02400 is not last in $X_2 \times \cdots \times X_6$) and so $\mathtt{next}_6(\boldsymbol{x}) = 3 \bullet \mathtt{next}_5(02400)$.

The state is now $\mathtt{Down}$; then $\mathtt{Lil}_5(02400) = \text{false}$ (0 is not last in X_2, 2400 is not first in $X_3 \times \cdots \times X_6$) and so $\mathtt{next}_6(\boldsymbol{x}) = 3 \bullet 0 \bullet \mathtt{next}_4(2400)$.

The state is $\mathtt{Down}$; then $\mathtt{Lil}_4(2400) = \mathtt{Lil}_3(400) = \text{true}$ and

$$\mathtt{next}_4(2400) = 2 \bullet \mathtt{next}_3(400) = \cdots = 2401.$$

Hence $\mathtt{next}_6(\boldsymbol{x}) = 302401$. Note that we did not need to compute $\mathtt{back}$ in this case.

(4) Let now $\boldsymbol{x} = 4323 \in \mathbb{N}^4,\ m_4 = 5,\ m_1 = m_2 = m_3 = 1$.

Then $\mathtt{Lil}_4(4323) = \text{false}$, $\mathtt{Lil}_3(323) = \text{true}$, and $\mathtt{next}_4(\boldsymbol{x}) = (4+1) \bullet \mathtt{back}_3(323) = 5313$. □

Remark 5.10. Computation is accelerated if we note the following facts, stated as in Proposition 5.8.

Claim 1: $\mathtt{Lil}_i(y_i, \dots, y_n)$ implies $\mathtt{Lil}_{i+1}(y_{i+1}, \dots, y_n)$.

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

Open questions 5.11. With the hypothesis of Proposition 5.8, design an algorithm to determine the rank $\operatorname{rk}(\boldsymbol{x})$ of tuple $\boldsymbol{x}$ in the ordering $\le_U$, and conversely, to determine the tuple of given rank i.

These tasks are easy in the case of Definition 2.2, i.e., when $n = 2$ and $m_1 = m_2 = \infty$, because then

$$\begin{aligned} \operatorname{rk}(i, j) = \text{IF } i + j \text{ is even } \text{THEN } & (i + j + 1)(i + j + 2)/2 - j \\ \text{ELSE } & (i + j + 1)(i + j + 2)/2 - i. \end{aligned}$$

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

6. Conclusion

We presented a few 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 Cartesian products $\mathbb{N} \times \mathbb{N} \times \cdots \times \mathbb{N}$?

(2) Which automata can define d2-orderings, which may not respect levels?

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

(4) Does there exist a finite automaton that can d2-ℓ-order an affine subset $Z \subseteq \mathbb{N} \times \mathbb{N} \times \cdots \times \mathbb{N}$ defined by conditions of the form $a_1 i_1 + \cdots + a_n i_n \leq b$, with $b \geq 0$ (ensuring that Z contains $(0, 0, \ldots, 0)$). Already for $n = 2$, we have a counterexample in Figure 17. We would need to generalise conditions (C1)–(C4) to larger dimensions.

Appendix

The `Enum` package is part of the `TRAG`[11] 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 [Durand 2012]. It offered the possibility of creating basic enumerators (inductive, from a list, ...) and combining existing ones by using operations like taking a product, sequencing or filtering. The general product built on the binary product does not give d2-orderings.

Here we give some hints about how we programmed a bidirectional levelled enumerator which enumerates a (possibly infinite) Cartesian product by producing d2-ℓ-orderings.

A.1. *Enumerators and bidirectional enumerators.*

A.1.1. *General enumerators.* In the following, an enumerator E is 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`.

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
```

[11] https://trag.labri.fr

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
```

A.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)`: latest 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 one) and the second, the ones that are after, and a slot `latest-object` containing the last enumerated object. If the enumerator is moving backwards, the top element of `past-object` will be produced and moved to `latest-object`; otherwise the top element of `future-object` will be produced and moved to `latest-object`.

Creation and initialization of a bidirectional enumerator. Given a nonempty 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
```

A.2. ***Enumeration of Cartesian products.*** Let $E_1, \dots, E_p$ be nonempty enumerators (finite or not) such that $E_i = (e^i_0, e^i_1, \dots)$ if it is infinite, $E_i = (e^i_0, e^i_1, \dots, e^i_{c_i-1})$ where $c_i = |E_i|$ otherwise. There are no repetitions. Let $T_p = E_1 \times E_2 \times \cdots \times E_p$. The necessity of diagonal enumeration of T_p arises in particular when one of the components is infinite.

A.2.1. *Levelled enumerators of Cartesian products.* The *height* of a tuple $t = (e^1_{j_1}, e^2_{j_2}, \dots, e^p_{j_p}) \in T_p$ is the sum $\Sigma^p_{i=0} j_i$ of the indices of the elements in the sets E_i. We note here by L^i the i-th *level* of T_p which is the finite set of tuples of height i. Then T_p is the partition of its levels. If T_p is infinite, its number of levels is infinite. The definitions of height and level in Definition 2.1 of Section 2 concern the particular case where the enumerated sequences are $\mathbb{N}$ or intervals $[0, p] \subset \mathbb{N}$.

A *levelled enumerator* enumerates $L^0, L^1, \dots$ in increasing order. We call *major step* a step that moves to the next level and *minor step* a step inside a level. Levelled enumerators have in addition the predicate

`minor-step-p (E),`

which is true if the next step (`next-element`) does not change the level. It is false when we are done with the enumeration of the current level.

A.2.2. *Bidirectionals levelled enumerators.* A *levelled bidirectional enumerator* is a levelled enumerator which in addition, is bidirectional (it has a `way` and an `initial-way`). When going forward (`way = +1`), it enumerates levels in increasing order: $L^0, L^1, \dots$ When going backwards (`way = -1`), it enumerates them in decreasing order: $L^i, L^{i-1}, \dots$ while keeping the forward order inside the levels.

A.2.3. *Diagonal product of a bidirectional enumerator with a bidirectional levelled enumerator.* We implement Definition 5.1(b). Let `X` be a bidirectional enumerator and `Y` be a bidirectional levelled enumerator which when going forward enumerates the levels $Y^0, Y^1, \dots$. We define below `DP(X, Y)`, the *diagonal product* of `X` and `Y`. When `DP(X, Y)` is created, 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 the call `(sliding-step X Y -1)` corresponds to a back step (move to lower level, see Definition 5.4). If neither `X` nor `Y` can move in their current ways 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 case of a corner step, we reverse 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 its 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))
```

A.3. ***Diagonal enumeration of a Cartesian product.*** Let `Nil` be the bidirectional levelled 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 recursive use of `DP` yields the levelled-ℓ-ordering defined in Definition 5.1 and described in Observation 5.3.

Proposition A.1. *Let $E_1, E_2, \ldots, E_p$ be bidirectional enumerators. The enumerator*

$$\mathtt{DP}(E_1, \mathtt{DP}(E_2, \mathtt{DP}(\ldots, \mathtt{DP}(E_p, \mathtt{Nil}))))$$

is a bidirectional levelled enumerator and defines a d2-ℓ-ordering of $E_1 \times E_2 \times \cdots \times E_p$.

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))
```

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). We thank the referee for drawing our attention to the Fueter–Pólya theorem.

References

[Courcelle and Durand 2016] B. Courcelle and I. Durand, "Computations by fly-automata beyond monadic second-order logic", *Theoret. Comput. Sci.* **619** (2016), 32–67. MR Zbl

[Durand 2012] I. Durand, "Object enumeration", pp. 43–57 in *Proceedings of the 5th European Lisp Symposium* (Zadar, Croatia, 2012), edited by F. Pehar, 2012.

[Engelfriet and Hoogeboom 2007] J. Engelfriet and H. J. Hoogeboom, "Automata with nested pebbles capture first-order logic with transitive closure", *Log. Methods Comput. Sci.* **3**:2 (2007), art. id. 3. MR Zbl

[Fraigniaud et al. 2005] P. Fraigniaud, D. Ilcinkas, G. Peer, A. Pelc, and D. Peleg, "Graph exploration by a finite automaton", *Theoret. Comput. Sci.* **345**:2-3 (2005), 331–344. MR Zbl

[Wikipedia 2011] "Gray code", Wikipedia entry, 2011, available at https://en.wikipedia.org/wiki/Gray_code.

[Wikipedia 2015] "Fueter–Pólya theorem", Wikipedia entry, 2015, available at https://tinyurl.com/fuetpol.

Received 27 Nov 2019. Revised 2 Apr 2020.

BRUNO COURCELLE:

bruno.courcelle@u-bordeaux.fr
LaBRI, University of Bordeaux and CNRS, Talence, France

IRÈNE DURAND:

irene.durand@u-bordeaux.fr
LaBRI, University of Bordeaux and CNRS, Talence, France

MICHAEL RASKIN:

raskin@mccme.ru
Technische Universität München, Garching bei München, Germany

Moscow Journal of Combinatorics and Number Theory

See inside back cover or msp.org/moscow for submission instructions.

The subscription price for 2020 is US $310/year for the electronic version, and $365/year (+$20, if shipping outside the US) for print and electronic. Subscriptions, requests for back issues and changes of subscriber address should be sent to MSP.

Moscow Journal of Combinatorics and Number Theory (ISSN 2640-7361 electronic, 2220-5438 printed) at Mathematical Sciences Publishers, 798 Evans Hall #3840, c/o University of California, Berkeley, CA 94720-3840 is published continuously online. Periodical rate postage paid at Berkeley, CA 94704, and additional mailing offices.

MJCNT peer review and production are managed by EditFlow® from MSP.

PUBLISHED BY
mathematical sciences publishers
nonprofit scientific publishing
http://msp.org/

Moscow Journal of Combinatorics and Number Theory

vol. 9 no. 3 2020

Thematic Issue on Complexity