Every Constructive Countable Partially Ordered Set With No Maximal Elements Can Be Constructively Divided into Disjoint Omega-Chains: Result and Possible Applications

Olga Kosheleva and Vladik Kreinovich

Abstract It was recently proven that any countable partially ordered set with no maximal elements can be divided into disjoint omega-chains, i.e., ordered subsets isomorphic to the set of natural numbers. In this paper, we prove that this division can be done algorithmically. We also discuss possible applications to the analysis of multiple personality disorder.

1 Formulation of the problem

Definitions: reminder. To formulate the problem in precise terms, let us recall a few definitions. A pair $\langle X, <\rangle$ of a set X and a binary relation $<$ is called a *partially ordered set* (or *poset*, for short) if the following two condtions are satisfied for all a, b, and c:

- if $a < b$ and $b < c$, then $a < c$ (*transitivity*);
- it is not possible to have $a < a$.

When $a < b$, we say that a is *smaller* than b and b is *larger* than a.

An element $a \in X$ of a poset if called *maximal* if no element of X is larger than a.

Finally, a poset $(X, <)$ is called an ω*-chain* if it is isomorphic to the set of natural numbers with natural order, i.e., when $X = \{a_0, a_1, a_2, \ldots\}$ so that $a_i < a_j \Leftrightarrow i < j$.

Known result. It is known – see, e.g., [10] – that every countable poset with no maximal elements can be divided into disjoint ω-chains.

Olga Kosheleva
Department of Teacher Education, University of Texas at El Paso, 500 W. University
El Paso, Texas 79968, USA, e-mail: olgak@utep.edu

Vladik Kreinovich
Department of Computer Science, University of Texas at El Paso, 500 W. University
El Paso, Texas 79968, USA, e-mail: vladik@utep.edu

Natural question. It is known that some existence results – even when the proof is not constructive – are actually algorithmic, in the sense that by using appropriate algorithms, we can construct the corresponding objects. On the other hand, it is also known that some well-known existence results are not algorithmic; see, e.g., [1, 2, 3, 4, 5, 6, 7, 8, 9, 11]. So, a natural question is whether the division into chains can be done algorithmically.

What we do in this paper. In this paper, we prove that the desired division can indeed be done algorithmically.

2 Definitions and the main result

Definition.

- *We say that a countable poset $\langle\{x_0, x_1, \ldots\}, <\rangle$ is* constructive *if there exists an algorithm that, given i and j, checks whether $x_i < x_j$.*
- *We say that a division of a poset $\langle\{x_0, x_1, \ldots\}, <\rangle$ into ω-chains is* constructive *if there exist the following two algorithms:*
  - *the first algorithm, given i and j, checks whether x_i and x_j belong to the same ω-chain; and*
  - *the second algorithm, given an integer i, returns the element next to x_i in the corresponding ω-chain, and either returns the previous element of this chain or – if there is no previous element – returns the corresponding message.*

Proposition. *Every constructive countable poset with no maximal elements can be constructively divided into disjoint ω-chains.*

3 Proof

How we will prove. We will start with one of the possible non-constructive proofs, and we will show how to make this proof constructive.

A non-constructive proof with which we start. Since X is countable, we can number its elements as $x_1, x_2, \ldots$. Let us show how to form disjoint ω-chains C_1, C_2, $\ldots$, whose union will form the whole set X, so that for each k, in the remaining set $X_k \stackrel{\text{def}}{=} X - (C_1 \cup \ldots \cup C_{k-1})$ for every element there is a larger one.

For $k = 1$, the remainder set is the original set X, for which this property is true – since X has no maximal elements.

Let us assume that we already have formed the chains $C_1, \ldots, C_{k-1}$, and let us show how to form C_k. Let s_k be the smallest index j for which x_j does not belong to any of the already formed chains. For each element x_i from the remainder set X_k, in

this same set X_k there is a larger element $x_j > x_i$. Let us denote one of these elements by $L_k(i)$. Then, as the next chain, we take

$$C_k = \{x_{s_k}, x_{L_k(L_k(s_k))}, x_{L_k(L_k(L_k(L_k(s_k))))}, \ldots\}. \tag{1}$$

Let us show that in the remaining set $X_{k+1} = X_k - C_k$, still every element x_i has a larger one. Indeed, by inductive assumption, there is an element $x_j \in X_k$ for which $x_i < x_j$, e.g., the elements x_j with $j = L_k(i)$.

- If this larger element x_j is not in C_k, then it is in X_{k+1}.
- If the larger element x_j is in C_k, then, as can see from the construction of C_k, we have $x_i < x_j < x_{L_k(j)}$ (so $x_i < x_{L_k(j)}$) and $x_{L_k(j)} \notin C_k$.

Let us now show that these chains cover the whole set X. Indeed, on the first step, the smallest yet-uncovered index is $s_1 = 1$. On each step k, the smallest yet-uncovered index is larger than s_{k-1} – so $s_k > s_{k-1}$. From this inequality, by induction, we conclude that $s_k \geq k$. By the construction of s_k, all smaller indices $i < s_k$ are already covered by one of the previously constructed chains. This means that all the elements x_i with $i \leq k$ are already covered by one of the chains $C_1, \ldots, C_k$. In particular, for $k = i$, this means that the element x_i is covered by one of the chains $C_1, \ldots, C_i$. Thus, the constructed ω-chains indeed form a partition of the original poset.

Preliminary discussion. To perform the desired construction algorithmically, let us recall that in a poset with no maximal elements, each element x_i has infinitely many larger elements. Indeed, since no elements are maximal, there is a larger element $x_j > x_i$. Similarly, for x_j, there is an element $x_\ell > x_j$ which is – by transitivity – also larger than x_i and different from x_j, etc.

Since for every i for which $x_i \in X_k$, there are infinitely many elements $x_j \in X_k$ for which $x_j > x_i$ and there are only finitely many indices $j < i$, there are indices $j > i$ for which $x_j \in X_k$ and $x_j > x_i$.

Let us show how to make this proof constructive. Let us now show, by induction over k:

- how to make each function $L_k(i)$ algorithmic,
- how to algorithmically select s_k, and
- how to form an algorithm A_k that, given an index i, checks whether $x_i \in C_k$.

Let us assume that this is already done for the chains $C_1, \ldots, C_{k-1}$. Let us show how to algorithmically form L_k, s_k, and A_k.

How to constructively select L_k. For each $i \in X_k$, let us take, as $L_k(i)$, the smallest index $j > i$ for which $x_j > x_i$ and $x_j \in X_k$ (i.e., $x_j \notin C_1$, and $\ldots$, and $x_j \notin C_{k-1}$). It is easy to algorithmically compute the value $L_k(i)$: indeed, we can try the values $j = i+1, i+2, \ldots$, and for each of these values we check:

- whether $x_j > x_i$ – this can be done since X is a constructive poset, and
- whether $x_j \in C_1$, $\ldots$, $x_j \in C_{k-1}$ – this can be done by using algorithms $A_1, \ldots, A_{k-1}$ that have already been constructed.

We stop when we find the smallest index j for which $x_j > x_i$ and $x_j \in X_k$. We then return this value j as $L_k(i)$.

How to constructively select s_k. As s_k, we select the smallest j for which $x_j \notin C_1 \cup \ldots \cup C_{k-1}$. This value s_k can also be found algorithmically: just take $j = 1, 2, \ldots$, for each of them check whether $x_j \in C_1$, $\ldots$, and whether $x_j \in C_{k-1}$. When we find the first index j for which x_j is not in any of the previously formed chains, we return this index as s_k.

Then, we form a set C_k by following the formula (1).

How to form an algorithm A_k. Let us construct an algorithm that, given an index i, checks whether x_i belongs to the k-th ω-chain C_k.

This algorithm is straightforward: we generate indices j of all the elements of the constructed ω-chain C_k:

$$j_1 = s_k, j_2 = L_k(L_k(s_k)), j_3 = L_k(L_k(L_k(L_k(s_k)))), \ldots \tag{2}$$

while we still have $j_\ell \le i$. If x_i is the element of the ω-chain, we will find the corresponding j_ℓ and thus, conclude that x_i belongs to this chain. If we reach $j_\ell > i$ and none of previous indices $j_1, \ldots, j_{\ell-1}$ is equal to the given i, we conclude that x_j does not belong to the chain C_k.

This completes the construction of the division of the original poset into ω-chains.

Desired algorithms. To complete the proof, let us show:

- how, based on the above construction, we can check whether two elements of X belong to the same chain, and
- how we can find the next and previous elements in the chain that contains the given element x_i.

To check whether the elements x_i and x_j belong to the same chain, we check, for $k = 1, 2, \ldots$, whether x_i belongs to the chain C_1, C_2, etc. This can be done by applying the corresponding algorithms A_1, A_2, etc. Since the chains cover the whole set X, for some k, we will find out that $x_i \in C_k$. Then, we can use the algorithm A_k to check whether the element x_j belongs to the same chain C_k.

To find the next and the previous elements of a given element x_i, we again first find k for which $x_i \in C_k$. Then:

- we compute s_k, and
- we compute, one of one, indices (2) of the elements (1) of the chain C_k

until we find the index i.

Then we can easily find the next-to-x_i element of the chain C_k, and – if there is a one – the previous-to-x_i element of C_k.

The proposition is proven.

4 Possible applications: a speculative comment

Situations. Many of our actions we perform consciously, clearly thinking about them and having decided to do them. This is how we write papers, this is how we do many other thinks. On the other hand, many of our actions are done automatically, without us thinking too much about it. For example, usually, when a person walks from his/her office to the main department office, this person follows the path automatically without thinking. If we accidentally slightly collide with someone, we usually automatically say "Sorry" without thinking about it, etc.

In such situations, it feels like some other "ego" is in control. This is a normal feeling, but sometimes, it is exaggerated to a pathological level, when a person has split personalities that take care of different actions.

What our result can tell about such situations. Sometimes, we do not feel in control of an automatic action, but we do remember everything we did. Sometimes, e.g., when a person is in deep thought, this person may not remember how exactly he/she walked to the department's office. This can be really disjoint – like in the story of Dr. Jekyll and Mr. Hyde, where different personalities do not have memory of what the other personality did. All this can be described by an order relation $a < b$ meaning that at moment b, we remember what happened at moment a.

What our result shows that in all such cases – unless there is an event that no one remembers – we can describe each complex "remembering" relation $<$ by dividing all the events into different "personalities". Who knows, maybe this can help with split personality disorders. Of course, this is an approximate model; in reality:

- the set of events is large but still finite, and
- remembering is not always transitive: when a chain is long, we may forget,

but maybe our result will somehow help.

Acknowledgements

This work was supported by the AT&T Fellowship in Information Technology, by the Institute for Risk and Reliability, Leibniz Universitaet Hannover, Germany, by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) Focus Program SPP 100+ 2388, Grant Nr. 501624329, by the European Union under the project ROBOPROX (No. CZ.02.01.01/00/22 008/0004590), by the Center of Excellence in Econometrics, Faculty of Economics, Chiang Mai University, Thailand, by the Ho Chi Minh City University of Banking, Vietnam, and by Thang Long University, Hanoi, Vietnam.

The authors are thankful to all the participants of the 35th NMSU/UTEP Workshop on Mathematics, Computer Science, and Computational Science (Las Cruces, New Mexico, USA, April 11, 2026), especially to Ilya Shapirovsky and Remi Salinas Schmeis, for valuable discussions.

References

1. O. Aberth, *Precise Numerical Analysis Using C++*, Academic Press, New York, 1998.
2. M. J. Beeson, *Foundations of constructive mathematics*, Springer-Verlag, N.Y., 1985.
3. E. Bishop, *Foundations of Constructive Analysis*, McGraw-Hill, 1967.
4. E. Bishop and D. S. Bridges, *Constructive Analysis*, Springer, N.Y., 1985.
5. D.S. Bridges, *Constructive Functional Analysis*, Pitman, London, 1979.
6. D. S. Bridges and S. L. Vita, *Techniques of Constructive Analysis*, Springer-Verlag, New York, 2006.
7. V. Kreinovich, A. Lakeyev, J. Rohn, and P. Kahl, *Computational complexity and feasibility of data processing and interval computations*, Kluwer, Dordrecht, 1998.
8. B. A. Kushner, *Lectures on Constructive Mathematical Analysis*, Amer. Math. Soc., Providence, Rhode Island, 1984.
9. M. Pour-El and J. Richards, *Computability in Analysis and Physics*, Springer-Verlag, New York, 1989.
10. R. Salinas Schmeis, "Partitioning piostes into ω-chains", *Abstracts of the 35th NMSU/UTEP Workshop on Mathematics, Computer Science, and Computational Science*, Las Cruces, New Mexico, USA, April 11, 2026, p. 39.
11. K. Weihrauch, *Computable Analysis*, Springer Verlag, Berlin, 2000.