The Lattice of Topological Domains :
An Example of Mizar Development

Zbigniew Karno

Institute of Mathematics

Warsaw University in Białystok

15-267 Białystok, Akademicka 2, Poland

Abstract

We present the complete development of the theory of topological domains (regular sets) in the Mizar formalism. It enveloped the lattice-theoretic foundations of the theory of domains that was given in [23] and in [12]. The aim is to present the main result in [11] concerning certain characterizations of an extremally disconnected topological space by means of properties of the lattice of domains and the main result in [12] concerning completeness of the lattice of domains of a topological space. Beforehand in comparison with a mathematical vernacular we present the mizared definitions of all useful notions and the all theorems which are necessary to the explanations.

The development presented here uses extracts of articles in the Main Mizar Library (MML), which have been fully checked by the PC Mizar System (Version 3.37). Therefore, we formulate only all definitions and all theorems without the proof justifications. The full articles with proofs are available in the MML distributed together with the PC Mizar System.

Contents

1 Introduction

The purpose of this paper is to give the complete development of the theory of topological domains in the Mizar formalism. It should be noted that our approach has lattice–theoretic foundations. Therefore, we first present necessary parts of the theory of topological spaces ([6], [7]) and the theory of lattices ([24], [1]) which are available in the Main Mizar Library (MML). We next present the main result in [11] on certain characterizations of an extremally disconnected topological space by means of properties of the lattice of domains and the main result in [12] on completeness of the lattice of domains of a topological space. Because recently the library evolves quite fast, it should be mentioned, that it is the stage of the MML that fits the 3.37 version of the PC Mizar System.

The theory of closed and of open domains has been introduced by Kuratowski [15] in connection with the closure and the interior operations in topological spaces and by Stone [19] in connections with the theory of Boolean algebras. Subsequently, it has been extensively investigated by several authors (see e.g. Birkhoff [2], Tarski [21], MacNeille [17]).

Let us recall the concepts of closed and of open domains. Let X be a topological space and let A be a subset of X. A is said to be a *closed domain* (or *regularly closed set*) if $A = \overline{\operatorname{Int} A}$, and A is said to be an *open domain* (or *regularly open set*) if $A = \operatorname{Int} \overline{A}$ (see e.g. [15], [13]). In the family of all closed domains in X let us define two binary operations $\vee$ and $\wedge$ in the following way : $A \vee B = A \cup B$ and $A \wedge B = \overline{\operatorname{Int}(A \cap B)}$, where A and B are arbitrary closed domains. Similarly, in the family of all open domains in X let us define two binary operations $\vee$ and $\wedge$ in the following way : $A \vee B = \operatorname{Int} \overline{A \cup B}$ and $A \wedge B = A \cap B$, where A and B are arbitrary open domains. It is well known (see e.g. [2], [16]) that for a given topological space all its closed domains with operations $\vee$ and $\wedge$ defined above form a complete Boolean lattice, and similarly all its open domains with analogous operations $\vee$ and $\wedge$ form a complete Boolean lattice, too.

A simple generalization of the notions of closed and of open domains is the following one : A is said to be a *domain* (or a *regular set*) in X provided $\operatorname{Int} \overline{A} \subseteq A \subseteq \overline{\operatorname{Int} A}$ (see [7] and compare [10] where a domain is called a *condensed set*). Of course, any closed domain is a domain and any open domain is a domain, too.

In [23] and [12] certain connections between these concepts of domains have been introduced and studied. In particular, in the family of all domains of a given topological space X two binary operations $\vee$ and $\wedge$ have been defined. They are defined as follows : $A \vee B = \operatorname{Int} \overline{A \cup B} \cup (A \cup B)$ and $A \wedge B = \overline{\operatorname{Int}(A \cap B)} \cap (A \cap B)$, where A and B are arbitrary domains in X. In [23] it has been proved that all domains of a given topological space with the operations $\vee$ and $\wedge$ defined above form a complemented lattice. In the same paper it has been shown that both the lattice of open domains and the lattice of closed domains are sublattices of the lattice of all domains. In [12] it has been proved an extension of an analogous theorem for the lattices of closed and of open domains; namely that the lattice of all domains of a given topological space is complete.

In [11] a condition whether the lattice of all domains of a given topological space is Boolean has been formulated. To present that one we first recall a definition of a class of topological spaces which is important here. A topological space X is called *extremally disconnected* if for every open subset A of X the closure $\overline{A}$ is open in X [20] (comp. [14], [8]). In [11] the following characterization has been obtained. The lattice of all domains of a topological space X is modular if and only if X is extremally disconnected. Moreover, for every extremally disconnected space the lattice of all its domains coincide with both the lattice of all its closed domains and the lattice of all its open domains. Using these facts the following characterization of extremally disconnected spaces generalizing the previous one has been obtained. Namely, the lattice of all domains of a topological space X is Boolean if and only if X is extremally disconnected. The aim here is to present of these results using the Mizar formalism.

The paper is written for people who are a little more familiar with the Mizar language; hence, a casual reader may have to refer first to some of the cited paper e.g. [22], [3].

2 Topological spaces in Mizar

In this section we present all necessary facts on topological spaces formulated in the Mizar formalism. To the explanation we use extracts of the articles [6] and [7], which are available in the MML.

2.1 Mizared definition of topological spaces

We mean by a *topological space* (see [13] and comp. [8]) a pair $(X, \mathcal{T})$ consisting of a set X and a family $\mathcal{T}$ of subsets of X satisfying the following conditions :

    (1) $X \in \mathcal{T}$;

    (2) if $U_1 \in \mathcal{T}$ and $U_2 \in \mathcal{T}$, then $U_1 \cap U_2 \in \mathcal{T}$;

    (3) if $\mathcal{U}$ is a subfamily of $\mathcal{T}$, then $\bigcup \mathcal{U} \in \mathcal{T}$.

The set X is called a *space* and the family $\mathcal{T}$ is called the *topology* of X. The elements of X are called *points* of the space, and the subsets of X are called *subsets* of the space. A subset of the space X belonging to $\mathcal{T}$ is called *open* and its complement is called *closed*.

In Mizar the definition of a topological space is somewhat different than this one given above and has been done in three succeeding steps. It has been defined first a two fields structure mode called `TopStruct`, which has as a prefix the structure mode `1-sorted`. It looks as follows (see [6]).

```
struct(1-sorted) TopStruct (# carrier -> non-empty set,

                            topology -> Subset-Family of the carrier #);
```

In the structure definition there occur two symbols of selectors : `carrier` and `topology`. The symbol `carrier` corresponds to the first field of this structure and the symbol `topology` corresponds to the second one. Thus `the carrier` and `the topology` are field terms and the first one has the type `non-empty set` and the second one has the type `Subset-Family of the carrier`. Observe that the field term `the carrier` there occurs in the pattern of that structure mode. Now if `X` has a type which widens to the type `TopStruct`, then `the carrier of X` and `the topology of X` are selector terms and the first one has the type `non-empty set` and the second one has the type `Subset-Family of the carrier of X`.

In the next step, we can define for every structure of a type which widens to the type `TopStruct` the attribute `TopSpace-like`, using the *Clustered Attribute Definition*. Every such attributed structure must satisfied all of the conditions (1), (2) and (3) given above. It has been done in the following way (comp. [6]).

```
definition mode TopSpace-like -> TopStruct means :: PRE_TOPC:def 1
  the carrier of it ∈ the topology of it &
   (for F being Subset-Family of the carrier of it st
    F ⊆ the topology of it holds
     union F ∈ the topology of it) &
   (for A,B being Subset of the carrier of it st
    A ∈ the topology of it & B ∈ the topology of it holds
     A ∩ B ∈ the topology of it);
end;
```

In a correctness condition to this definition the existence of a structure of the type `TopSpace-like TopStruct` has been justified. In effect this definition produces an existential cluster, which ensures that the class of all structures of the type `TopSpace-like TopStruct` is nonempty. Now, every structure of a type which widens to the type `TopSpace-like TopStruct` is a structure satisfying all of the conditions given in the definition of a topological space. So, in the last step we can define the notion of a topological space as the

type `TopSpace` coinciding with the type `TopSpace-like TopStruct` using the *Expandable Mode Definition*. It looks as follows (comp. [6]).

```
definition mode TopSpace is TopSpace-like TopStruct;

end;
```

The existential cluster has been constructed next, which ensures that the class of all structures of the type `TopSpace` with the attribute `strict`, hence the class of all structures of the type `TopSpace`, is nonempty.

```
definition

 cluster strict TopSpace;

end;
```

Note that the meaning of the attribute `strict` in Mizar is very difficult to explain (see [22]). It requires a separate presentation, and in this paper it will be omitted.

To complete this subsection let us draw the following tree of the Mizar types. Here and subsequently the arrow indicates the direction of widening of types provided automatically by the Mizar System.
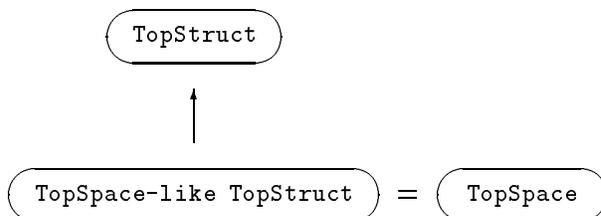
2.2   Subsets and subspaces in topological spaces

In [6], using the *Expandable Mode Definition*, the types `Point of X` and `Subset of X` have been introduced. But earlier the identifier `X` of the type `TopSpace` has been reserved. The reservation fixes `X` for a structure of the type `TopSpace` either in a whole Mizar article (i.e. from the reservation to the end of an article) or in its part till to a next reservation of the same identifier.

```
reserve X for TopSpace;



definition let X;

 mode Point of X is Element of the carrier of X;

end;
```

```
definition let X;

 mode Subset of X is set of Point of X;

end;
```

Next, using the *Functor Definition*, two functors $\emptyset$(X) and Ω(X) have been defined. They have the type Subset of X. From the theorems presented below it follows that $\emptyset$(X) equals $\emptyset$ and Ω(X) equals the carrier of X. But neither $\emptyset$ nor the carrier of X do not have the type Subset of X.

```
definition let X;
  func ∅(X) -> Subset of X means :: PRE_TOPC:def 2
   it = ∅ the carrier of X;
  func Ω(X) -> Subset of X means :: PRE_TOPC:def 3
   it = Ω the carrier of X;
end;

theorem :: PRE_TOPC:11
  ∅(T) = ∅;

theorem :: PRE_TOPC:12
  Ω(X) = the carrier of X;
```

Using the *Predicate Definition*, two predicates A is_open and A is_closed have been defined. But first the identifier A of the type Subset of X has been reserved globally.

```
reserve A for Subset of X;


definition let X,A;
  pred A is_open means :: PRE_TOPC:def 4
   A ∈ the topology of X;
end;

definition let X,A;
  pred A is_closed means :: PRE_TOPC:def 5
   Ω(X) \ A is_open;
end;
```

Next, using the *Functor Definition*, two functors Cl A and Int A of the type Subset of X have been introduced, the first one in [6] and the second one in [7]. These functors yield the closure and the interior in an arbitrary topological space (comp. [15], [13] and [8]).

```
definition let X be TopSpace, A be Subset of X;
  func Cl A -> Subset of X means :: PRE_TOPC:def 11
    for p being Point of X holds
     p ∈ it iff for G being Subset of X st G is_open holds
                                        p ∈ G implies A ∩ G <> ∅(X);
end;


definition let X be TopSpace, A be Subset of X;
```

```
 func Int A -> Subset of X means :: TOPS_1:def 1

    it = (Cl(A'))';

end;
```

It should be noted that in the above definition `A'` means the complement of `A` in `X`.

The operations `Cl` and `Int` have a lot of remarkable properties; they may be found in the MML in [6], [7], [23] and [12].

In [6], using the *Mode Definition*, the type `SubSpace of X` has been defined. Afterwards the existential cluster for this type has been done.

```
definition let X;
  mode SubSpace of X -> TopSpace means :: PRE_TOPC:def 8
    Ω(it) ⊆ Ω(X) &
      for P being Subset of it holds P ∈ the topology of it iff
        ex Q being Subset of X st Q ∈ the topology of X & P = Q ∩ Ω(it);
end;


definition let X;

 cluster strict SubSpace of X;

end;
```

2.3   Domains in topological spaces

In [7], using the *Predicate Definition*, three predicates `A is_domain`, `A is_closed_domain` and `A is_open_domain` have been introduced. Of course, the mizared definitions of these predicates are just the same as those given in the introduction.

```
definition let X,A;
  pred A is_domain means :: TOPS_1:def 6
    Int(Cl A) ⊆ A & A ⊆ Cl(Int A);
  pred A is_closed_domain means :: TOPS_1:def 7
    A = Cl(Int A);
  pred A is_open_domain means :: TOPS_1:def 8
    A = Int(Cl A);
end;
```

Based on [7] and [23], we present below selected properties of domains, closed domains and open domains (comp. [15], [13], [16]).

```
theorem :: TOPS_1:106

 A is_closed & A is_domain iff A is_closed_domain;



theorem :: TOPS_1:107
```

```
 A is_open & A is_domain iff A is_open_domain;


theorem :: TOPS_1:108
  A is_closed_domain & B is_closed_domain implies A ∪ B is_closed_domain;

theorem :: TOPS_1:109
  A is_open_domain & B is_open_domain implies A ∩ B is_open_domain;


theorem :: TOPS_1:111

 A is_domain implies Int A is_domain & Cl A is_domain;



theorem :: TDLAT_1:16

 for A being Subset of X st A is_domain holds A' is_domain;


theorem :: TDLAT_1:17
  for A,B being Subset of X st A is_domain & B is_domain holds
   Int(Cl(A ∪ B)) ∪ (A ∪ B) is_domain & Cl(Int(A ∩ B)) ∩ (A ∩ B) is_domain;


theorem :: TDLAT_1:22

 for A being Subset of X holds Cl(Int A) is_closed_domain;



theorem :: TDLAT_1:23

 for A being Subset of X holds Int(Cl A) is_open_domain;



theorem :: TDLAT_1:24

 for A being Subset of X st A is_domain holds Cl A is_closed_domain;



theorem :: TDLAT_1:25

 for A being Subset of X st A is_domain holds Int A is_open_domain;



theorem :: TDLAT_1:26

 for A being Subset of X st A is_domain holds Cl A' is_closed_domain;
```

```
theorem :: TDLAT_1:27

 for A being Subset of X st A is_domain holds Int A' is_open_domain;
```

Note that in [12] the closure and the interior operations for families of subsets of topological spaces have been introduced and their important properties have been presented (comp. [13], [15]). Using these notions, certain properties of domains, closed domains and open domains have been studied (comp. [15], [13], [16], [8]).

3   Special topological spaces

In this section, based on [11], we present some of the standard facts on discrete, anti-discrete, almost discrete, extremally disconnected and hereditarily extremally disconnected topological spaces formulated in the Mizar formalism (comp. [14], [8]).

3.1   Discrete spaces

We start with presentation of discrete topological structures in Mizar. In [11], using the *Attribute Definition*, for every structure of a type which widens to the type `TopStruct` the attributes `discrete` and `anti-discrete` have been introduced. It has been done as follows.

```
definition
  attr discrete -> TopStruct means :: TDLAT_3:def 1
   the topology of it = bool the carrier of it;
  attr anti-discrete -> TopStruct means :: TDLAT_3:def 2
   the topology of it = { ∅ , the carrier of it};
end;
```

Note that `bool A` means in Mizar the power set of `A`.

The existential cluster has been done next, which guarantees that the class of all structures of the type `TopStruct` with the attributes `discrete anti-discrete strict`, hence the class of all structures of the type `discrete anti-discrete TopStruct`, is nonempty. It looks as follows.

```
definition

 cluster discrete anti-discrete strict TopStruct;

end;
```

Afterwards two conditional clusters have been defined and proved. The first one ensures that every type which widens to the type `discrete TopStruct` widens also to the type `TopSpace-like TopStruct` and hence to the type `TopSpace`. Similarly, the second one ensures that every type which widens to the type `anti-discrete TopStruct` widens also to the type `TopSpace-like TopStruct` and hence to the type `TopSpace`.

```
definition

 cluster discrete -> TopSpace-like TopStruct;
```

```
 cluster anti-discrete -> TopSpace-like TopStruct;

end;
```

Next, for every structure of a type which widens to the type `TopStruct` the attribute `almost_discrete` has been introduced.

```
definition
 attr almost_discrete -> TopStruct means :: TDLAT_3:def 3
  for A being Subset of the carrier of it st
   A ∈ the topology of it holds
     (the carrier of it) \ A ∈ the topology of it;
end;
```

Then two conditional clusters have been defined and proved. These ones ensure that every type which widens either to the type `discrete TopStruct` or to the type `anti-discrete TopStruct` widens also to the type `almost_discrete TopStruct`.

```
definition

 cluster discrete -> almost_discrete TopStruct;

 cluster anti-discrete -> almost_discrete TopStruct;

end;
```

Afterwards the existential cluster has been constructed, which guarantees that the class of all structures of the type `almost_discrete strict TopStruct`, hence the class of all structures of the types `almost_discrete TopStruct`, is nonempty.
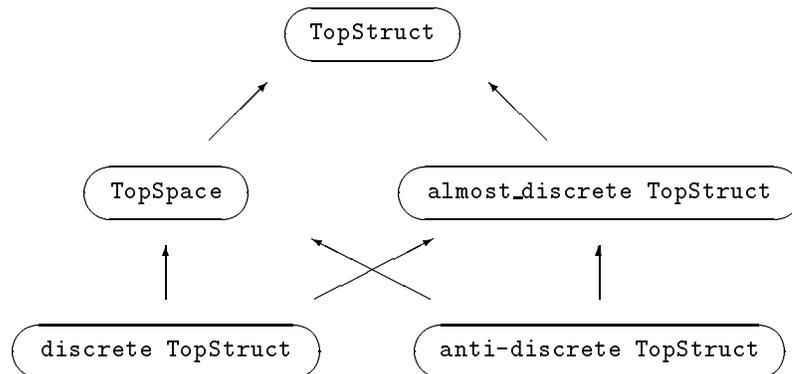
```
definition

 cluster almost_discrete strict TopStruct;

end;
```

The net of the types defined above looks like this.

We can now present discrete topological spaces in Mizar. In [11], the following two existential clusters have been constructed. The first one guarantees that the class of all structures of the type `discrete anti-discrete TopSpace` is nonempty and the second one guarantees that the class of all structures of the type `almost_discrete TopSpace` is nonempty.

```
definition

 cluster discrete anti-discrete strict TopSpace;

 cluster almost_discrete strict TopSpace;

end;
```

Then two conditional clusters have been defined and proved. These ones ensure that every type which widens either to the type `discrete TopSpace` or to the type `anti-discrete TopSpace` widens also to the type `almost_discrete TopSpace`.

```
definition

 cluster discrete -> almost_discrete TopSpace;

 cluster anti-discrete -> almost_discrete TopSpace;

end;
```

Now the tree of these types looks like this.
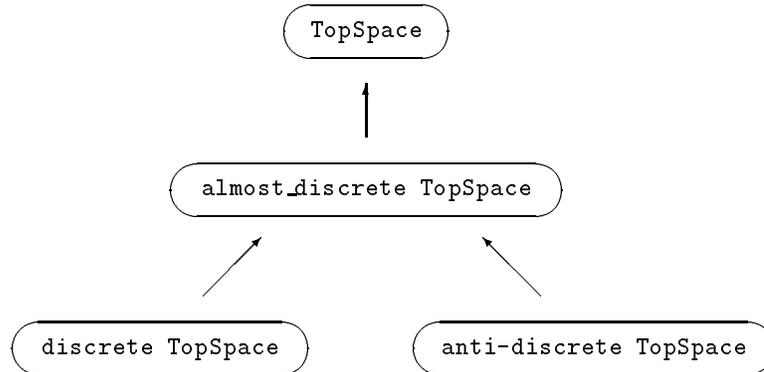
3.2 Extremally disconnected spaces

The mizared definition of extremally disconnected topological spaces has been done in [11]. Using the *Attribute Definition*, for every structure of a type which widens to the type `TopSpace` the attribute `extremally_disconnected` has been introduced as in the definition presented below. The definiens of this definition is the same as in the well known definition of extremally disconnected spaces [19] (comp. [14] or [8]).

```
definition
 attr extremally_disconnected -> TopSpace means :: TDLAT_3:def 4
```

```
    for A being Subset of it st A is_open holds Cl A is_open;
end;
```

Then the following existential cluster has been constructed. It guarantees, in particular, that the class of all structures of the type `extremally_disconnected TopSpace` is nonempty.

```
definition

 cluster extremally_disconnected strict TopSpace;

end;
```

Any extremally disconnected topological space can be characterized by certain properties of the family of all its domains. Some of these ones are listed in the theorems given below (see [11]).

```
reserve X for TopSpace;


theorem :: TDLAT_3:34

 X is extremally_disconnected iff

  for A being Subset of X st A is_domain holds A is_closed & A is_open;


theorem :: TDLAT_3:35

 X is extremally_disconnected iff

  for A being Subset of X st A is_domain holds

    A is_closed_domain & A is_open_domain;


theorem :: TDLAT_3:36

 X is extremally_disconnected iff

  for A being Subset of X st A is_domain holds Int Cl A = Cl Int A;


theorem :: TDLAT_3:38

 X is extremally_disconnected iff

  for A being Subset of X holds
```

```
    (A is_open_domain implies A is_closed_domain) &

     (A is_closed_domain implies A is_open_domain);
```

It is well known that there exists an extremally disconnected topological space with a subspace which is not extremally disconnected (see e.g. [8]). So, the definition of the type TopSpace with the attribute hereditarily_extremally_disconnected is sensible. It has been done in [11] as follows.

```
definition

 attr hereditarily_extremally_disconnected ->

                                        TopSpace means :: TDLAT_3:def 5

   for X0 being SubSpace of it holds X0 is extremally_disconnected;

end;
```

Afterwards the existential cluster for the type hereditarily_extremally_disconnected TopSpace has been constructed.

```
definition

 cluster hereditarily_extremally_disconnected strict TopSpace;

end;
```

Then two conditional clusters have been defined. The first one guarantees that every type which widens to the type hereditarily_extremally_disconnected TopSpace widens also to the type extremally_disconnected TopSpace. The second one guarantees that every type which widens to the type almost_discrete TopSpace widens also to the type hereditarily_extremally_disconnected TopSpace.

```
definition

 cluster

  hereditarily_extremally_disconnected -> extremally_disconnected TopSpace;

 cluster

  almost_discrete -> hereditarily_extremally_disconnected TopSpace;

end;
```

So, the previous tree of types extends to the following one.

```
                        ┌──────────────┐
                        │   TopSpace   │
                        └──────────────┘
                                ▲
                                │
         ┌────────────────────────────────────────┐
         │   extremally_disconnected TopSpace      │
         └────────────────────────────────────────┘
                                ▲
                                │
   ┌──────────────────────────────────────────────────────┐
   │  hereditarily_extremally_disconnected TopSpace        │
   └──────────────────────────────────────────────────────┘
                                ▲
                                │
         ┌────────────────────────────────────────┐
         │      almost_discrete TopSpace           │
         └────────────────────────────────────────┘
```
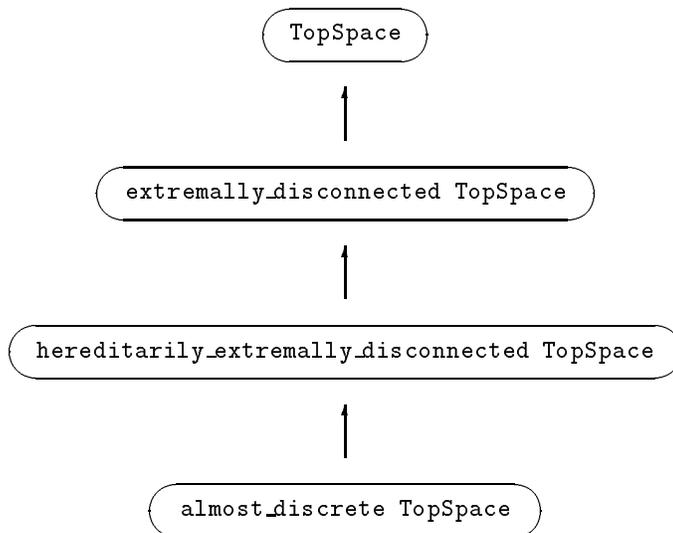
Remarks. The clustering constructions are very useful in Mizar. For instance the following Mizar sentence

```
 for X being almost_discrete TopSpace holds

   X is extremally_disconnected;
```

is obvious for the Mizar System. Moreover, the following sentence

```
 for X being discrete TopStruct holds

   X is hereditarily_extremally_disconnected TopSpace;
```

is also obvious (note that in it `TopSpace` can be dropped).

4   Lattices in Mizar

In this section we present all necessary facts from the theory of lattices formulated in the Mizar formalism. To the explanation we use extracts of the articles [24] and [1], which are available in the MML.

4.1   Mizared definition of lattices

We mean by a *lattice* (see [2], [9] and comp. [4]) a nonempty set L together with two binary operations $\vee$ and $\wedge$ defined on L satisfying the following conditions :

    (1) $a \vee b = b \vee a$ and $a \wedge b = b \wedge a$;

    (2) $a \vee (b \vee c) = (a \vee b) \vee c$ and $a \wedge (b \wedge c) = (a \wedge b) \wedge c$;

    (3) $a = a \vee (a \wedge b)$ and $a = a \wedge (a \vee b)$;

    (4) $a \vee a = a$ and $a \wedge a = a$.

The operations $\vee$ and $\wedge$ are called the *join* and the *meet*, respectively. It should be noted that there is the other way of defining lattices, based on the notion of order (see [2]).

In Mizar the definition of lattices is quite similar to given above. It has been done in several steps. First, a three fields structure mode called `LattStr` with the prefix `1-sorted` has been constructed. Then, the attribute `Lattice-like` for the type `LattStr`, which forms the type `Lattice`, has been defined.

The structure definition of `LattStr` looks as follows (comp. [24]).

```
struct(1-sorted) LattStr ≪ carrier -> non-empty set,
                    L_join,L_meet -> BinOp of the carrier ≫;
```

In the definition of this structure mode there occur the following three symbols of selectors of fields : `carrier`, `L_join` and `L_meet`. If `G` has a type which widens to the type `LattStr`, then the selector term `the carrier of G` has the type `non-empty set` and the selector terms `the L_join of G` and `the L_meet of G` have the type of `the BinOp of the carrier of G`. Here every function of the type `BinOp of A` has in Mizar the meaning of a binary operation on the set `A` (i.e. has the type `Function of [:A,A:],A`).

Using the *Functor Definition*, for given structure `G` of the type `LattStr` two functors `p ⊔ q` and `p ⊓ q` of the type `Element of the carrier of G` have been defined. These ones correspond to the binary operations ∨ and ∧, respectively, given above. But earlier the identifiers `G` of the type `LattStr` and `p`, `q` of the type `Element of the carrier of G` have been reserved globally.

```
reserve G for LattStr;

reserve p,q for Element of the carrier of G;


definition let G,p,q;
  func p ⊔ q -> Element of the carrier of G means :: LATTICES:def 1
   it = (the L_join of G).(p,q);
  func p ⊓ q -> Element of the carrier of G means :: LATTICES:def 2
   it = (the L_meet of G).(p,q);
end;
```

Next, using the *Predicate Definition*, for given structure `G` of the type `LattStr` the predicate `p ≤≤ q` has been introduced. It corresponds to a relation, which will has been defined a partial order in every structure of the type `Lattice`.

```
definition let G,p,q;
  pred p ≤≤ q means :: LATTICES:def 3
   p ⊔ q = q;
end;
```

For given structure `G` of the type `LattStr` the type `Subset of G` can be defined. It has been done in [1], using the *Expandable Mode Definition*.

```
definition let G be LattStr;

 mode Subset of G is Subset of the carrier of G;

end;
```

We are ready now to give the definition of lattices in Mizar. We can define for every structure of a type which widens to the type `LattStr` the attribute `Lattice-like` with the

definiens given by (1) - (3), using the *Clustered Attribute Definition*. It has been done as follows (comp. [24]).

```
definition
  mode Lattice-like -> LattStr means :: LATTICES:def 4
    (for a,b being Element of the carrier of it holds a ⊔ b = b ⊔ a) &
     (for a,b,c being Element of the carrier of it
                                   holds a ⊔ (b ⊔ c) = (a ⊔ b) ⊔ c) &
    (for a,b being Element of the carrier of it holds (a ⊓ b) ⊔ b = b) &
     (for a,b being Element of the carrier of it holds a ⊓ b = b ⊓ a) &
    (for a,b,c being Element of the carrier of it
                                   holds a ⊓ (b ⊓ c) = (a ⊓ b) ⊓ c) &
    (for a,b being Element of the carrier of it holds a ⊓ (a ⊔ b) = a);
end;
```

Now every structure of a type which widens to the type `Lattice-like LattStr` is a structure satisfying all of the conditions given in the definition of a lattice. So, we can introduce the notion of a lattice as the type `Lattice` coinciding with the type `Lattice-like LattStr`, using the *Expandable Mode Definition*. It has been done as follows (comp. [6]).

```
definition

 mode Lattice is Lattice-like LattStr;

end;
```

In a correctness condition to previous definition the existence of a structure of the type `Lattice-like LattStr` has been proved. In effect that definition produces an existential cluster, which guarantees that the class of all structures of the type `Lattice-like LattStr` is nonempty. On the other hand the following existential cluster has been constructed. It guarantees that the class of all structures of the type `strict Lattice`, hence the class of all structures of the type `Lattice`, is nonempty.
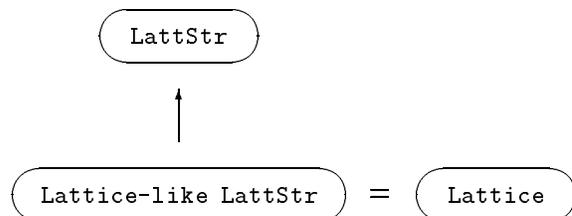
```
definition

 cluster strict Lattice;

end;
```

The tree of the Mizar types defined above looks like this.

In [5], using the *Mode Definition*, for every structure `L` of a type which widens to the type `Lattice` the type `SubLattice of L` has been introduced. Afterwards the existential cluster for this type has been done.

```
definition let L be Lattice;
  mode SubLattice of L -> Lattice means :: NAT_LAT:def 16
    the carrier of it ⊆ the carrier of L &
  the L_join of it =
          (the L_join of L)|[:the carrier of it,the carrier of it:]  &
  the L_meet of it =
          (the L_meet of L)|[:the carrier of it,the carrier of it:];
end;


definition let L be Lattice;

 cluster strict SubLattice of L;

end;
```

Note that in the previous definition | is the restriction operation and [:A,B:] denotes the Cartesian product of the sets A and B

4.2   Special lattices

In this subsection, based on [24], we present mizared definitions of the distributive, modular, complemented and Boolean lattices (comp. [2], [9]). Let us recall that a *distributive lattice* is a lattice which satisfies one of the equivalent conditions :

(5) $a \wedge (b \vee c) = (a \wedge b) \vee (a \wedge c)$,

(6) $a \vee (b \wedge c) = (a \vee b) \wedge (a \vee c)$.

In mizared definition we use the first one. Namely, we introduce for every structure of a type which widens to the type Lattice the attribute distributive, using *Clustered Attribute Definition*. It has been done in the following way.

```
definition
  mode distributive -> Lattice means :: LATTICES:def 5
    for a,b,c being Element of the carrier of it
                      holds a ⊓ (b ⊔ c) = (a ⊓ b) ⊔ (a ⊓ c);
end;
```

We define next the type D_Lattice as the type distributive Lattice, using the *Expandable Mode Definition*. Then we can construct the existential cluster for this type. It has been done as follows.

```
definition

 mode D_Lattice is distributive Lattice;

end;



definition

 cluster strict D_Lattice;

end;
```

Let us recall that a *modular lattice* is a lattice which satisfies the following conditions :
(7) if $a \leq b$, then $a \vee (b \wedge c) = a \wedge (b \vee c)$.

Mizared definition of modularity is the same. Similarly as in the definition of distributivity, we can introduce for every structure of a type which widens to the type **Lattice** the attribute **modular** with the definiens given by (7), using the *Clustered Attribute Definition*. One can define next the type **M_Lattice** as the type **modular Lattice**, using the *Expandable Mode Definition*. Then we can construct an existential cluster for this type. It has been done as follows.

```
definition
  mode modular -> Lattice means :: LATTICES:def 6
    for a,b,c being Element of the carrier of it st a ≤≤ c
                                  holds a ⊔ (b ⊓ c) = (a ⊔ b) ⊓ c;
end;
```

```
definition

 mode M_Lattice is modular Lattice;

end;
```

```
definition

 cluster strict M_Lattice;

end;
```

Before the mizared definition of complemented lattices we give first the definitions of certain special lattice elements. Using the *Conditional Functor Definition*, we can define for given lattice L (of the type **Lattice** of course) two functors $^{\perp}$L and $_{\top}$L of the type **Element of the carrier of L**. The first one corresponds to the least element 0 and the second one corresponds to the greatest element 1 in a bound lattice (comp. [9]). It has been done in [24] as follows.

```
reserve L for Lattice, a,b,c for Element of the carrier of L;
```

```
definition let L;
   assume ex c st for a holds c ⊓ a = c;
 func ⊥L -> Element of the carrier of L means :: LATTICES:def 10
   it ⊓ a = it;
end;
```

```
definition let L;
   assume ex c st for a holds c ⊔ a = c;
 func ⊤L -> Element of the carrier of L means :: LATTICES:def 11
   it ⊔ a = it;
end;
```

In [24], the type **01_Lattice** which extends to the type **Lattice** has been introduced. It corresponds to the concept of *bound lattices* with the least element 0 and the greatest

element 1 (comp. [9]). Let us recall that in a bound lattice L two its elements a and b are said to be *complementary*, whenever

$\quad$ (8) $a \vee b = 1$ and $a \wedge b = 0$.

Any element complementary to a is called a *complement* of a in L (see e.g. [9]). This concept has been defined in Mizar similarly. It uses the *Conditional Predicate Definition* and it is defined as the predicate a is_a_complement_of b with the definiens given by (8), where a,b have the type Element of the carrier of L and L has the type Lattice with the permissivness condition : L is 01_Lattice. So, this predicate is defined in fact to the type 01_Lattice. Precisely, it looks as follows.

```
definition let L,a,b;
  assume L is 01_Lattice;
 pred a is_a_complement_of b means :: LATTICES:def 12
  a ⊔ b = ⊤L & a ⊓ b = ⊥L;
end;
```

We are ready now to define the concept of complemented lattices. Let us recall that (comp. [9]) a *complemented lattice* is a bound lattice which satisfies the following conditions :

$\quad$ (9) for every a there exists b such that a is complement of b.

This definition in Mizar has been done in the same way (comp. [24]). First, using the *Clustered Attribute Definition*, For every structure of a type which widens to the type 01_Lattice the attribute complemented, with the definiens given by (9), has been introduced. Next, the type C_Lattice as the type complemented 01_Lattice has been defined and the existential cluster for this type has been constructed.

```
definition

 mode complemented -> 01_Lattice means :: LATTICES:def 13

  for b being Element of the carrier of it

   ex a being Element of the carrier of it st a is_a_complement_of b;

end;
```

```
definition

 mode C_Lattice is complemented 01_Lattice;

end;
```

```
definition

 cluster strict C_Lattice;

end;
```

Let us recall that a *Boolean lattice* is any distributive complemented lattice (see [9] and comp. [2]). In Mizar this definition is just the same. Namely, for every structure of a type

which widens to the type C_Lattice the attribute Boolean, with the definiens which says
that this structure is distributive, has been introduced (comp. [24]).

```
definition

 mode Boolean -> C_Lattice means :: LATTICES:def 14

   it is distributive;

end;
```

Afterwards the type B_Lattice as the type Boolean C_Lattice has been defined and
the existential cluster for this type has been constructed.
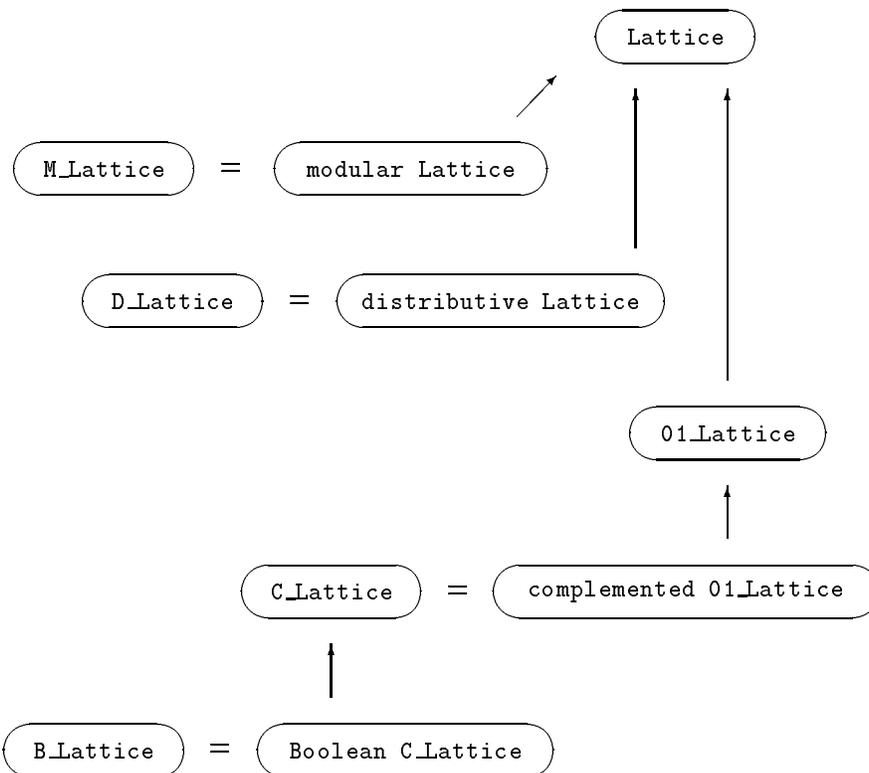
```
definition

 mode B_Lattice is Boolean C_Lattice;

end;
```

```
definition

 cluster strict B_Lattice;

end;
```

The tree of the lattice types looks like this.

Lattice

M_Lattice = modular Lattice

D_Lattice = distributive Lattice

01_Lattice

C_Lattice = complemented 01_Lattice

B_Lattice = Boolean C_Lattice

Unfortunately the widening of the type **D_Lattice** to the type **M_Lattice** and the widening of the type **B_Lattice** to the type **D_Lattice** are not supported automatically. Because in the MML the corresponding conditional clusters do not exist. But these widenings are accessible because of the following two theorems.

```
reserve L for D_Lattice;
```

```
theorem :: LATTICES:35
```

```
 L is modular;
```

```
reserve L for B_Lattice;
```

```
theorem :: LATTICES:59
```

```
 L is D_Lattice;
```

The second widening is accessible also by the definitional expansion of the definition of the attribute **Boolean** (it is supported automatically), or by the definitional theorem, which is automatically created from the definition of the attribute **Boolean** (here a label is needed).

4.3 Complete lattices

Let us recall that a lattice is said to be *complete* if for every its subset A both the supremum $\bigvee A$ of A and the infimum $\bigwedge A$ of A exist (see [2], [9]). Mizared definition of complete lattices is somewhat different. It uses in fact of the existence of the supremum $\bigvee A$. It has been done in [1] as follows. First, using the *Predicate Definition*, for given structure L of the type LattStr two predicates p is_less_than A and A is_less_than p with the synonyms A is_great_than p and p is_great_than A, respectively, have been introduced.

```
definition let L be LattStr, p be Element of L, A be set;
  pred p is_less_than A means :: LATTICE3:def 16
   for q being Element of L st q ∈ A holds p ≤≤ q;
  synonym A is_great_than p;
  pred A is_less_than p means :: LATTICE3:def 17
   for q being Element of L st q ∈ A holds q ≤≤ p;
  synonym p is_great_than A;
end;
```

Then using the *Attribute Definition*, for every structure of a type which widens to the type LattStr the attribute complete has been introduced as in the following definition.

```
definition
  attr complete -> LattStr means :: LATTICE3:def 18
   for A being set
    ex p being Element of it st A is_less_than p &
     for r being Element of it st A is_less_than r holds p ≤≤ r;
end;
```

Afterwards using the *Attribute Definition*, for every structure of a type which widens to the type LattStr two attributes $\bigvee$-distributive and $\bigwedge$-distributive have been defined and the following existential cluster for the type Lattice with the attributes complete $\bigvee$-distributive $\bigwedge$-distributive strict has been constructed. In particular, it ensures that the class of all structures of the type complete Lattice is nonempty.

```
definition
  cluster complete ⋁-distributive ⋀-distributive strict Lattice;
end;
```

Next, for every structure L of a type which widens to the type LattStr and satisfying the condition : L is complete Lattice, two functors $\bigvee$(A,L) and $\bigwedge$(A,L) have been introduced. They have meaning of the supremum and of the infimum, respectively.

```
definition let L be LattStr such that L is complete Lattice; let A be set;
  func ⋁(A,L) -> Element of L means :: LATTICE3:def 21
   A is_less_than it &
    for r being Element of L st A is_less_than r holds it ≤≤ r;
end;
```

```
definition let L be LattStr, A be set;
  func ⋀(A,L) -> Element of L means :: LATTICE3:def 22
   it = ⋁({p where p is Element of L : p is_less_than A},L);
end;
```

The existence conditions to the both definitions have been proved. These guarantee that for every structure L of a type which widens to the type LattStr and satisfying the condition : L is complete Lattice, both $\bigvee$(A,L) and $\bigwedge$(A,L) exist. In particular, it is also true for the structures of the type complete Lattice.

5 Lattices of domains of a topological space

5.1 The lattice of domains

We are ready now to introduce in Mizar the concept of the lattice of domains of a topological space. It has been done by Toshihiko Watanabe in [23]. First, for a given topological space X, i.e. for X of the type `TopSpace` (more for X of the type widens to the type `TopSpace`), the aggregate

> `LattStr` ≪ `Domains_of X,Domains_Union X,Domains_Meet X` ≫

has been constructed. It is created by application of the aggregate functor

> `LattStr` ≪   ,   ,   ≫

to the following list of functor terms `Domains_of X`, `Domains_Union X` and `Domains_Meet X`. These functors, defined below, have the required types as in the definition of the structure mode `LattStr` given in the subsection 4.1. Namely, `Domains_of X` has the type `non-empty set`, and both `Domains_Union X` and `Domains_Meet X` have the type `BinOp of the Domains_of X`. So, this aggregate is a structure of the type `LattStr`.

`Domains_of X` consists of all domains of the topological space X. Therefore in particular, it has the type `Subset-Family of the carrier of X`. Precise Mizar definition looks as follows.

```
definition let X be TopSpace;

 func Domains_of X ->

       non-empty Subset-Family of the carrier of X means :: TDLAT_1:def 1

  it = {A where A is Subset of X : A is_domain};

end;
```

Both `Domains_Union X` and `Domains_Meet X` are binary operations on `Domains_of X`. The value of the first one is Int $\overline{A \cup B} \cup (A \cup B)$, and the value of the second one is $\overline{\text{Int}(A \cap B)} \cap (A \cap B)$, where A and B are arbitrary domains in X. Observe that the functor `Domains_Union X` has the abbreviation `D-Union X` and the functor `Domains_Meet X` has the abbreviation `D-Meet X`.

```
definition let X be TopSpace;
  func Domains_Union X -> BinOp of Domains_of X means :: TDLAT_1:def 2
   for A,B being Element of Domains_of X holds
                           it.(A,B) = Int(Cl(A ∪ B)) ∪ (A ∪ B);
  synonym D-Union X;
end;

definition let X be TopSpace;
  func Domains_Meet X -> BinOp of Domains_of X means :: TDLAT_1:def 3
   for A,B being Element of Domains_of X holds
                           it.(A,B) = Cl(Int(A ∩ B)) ∩ (A ∩ B);
  synonym D-Meet X;
end;
```

Afterwards the following important theorem has been proved. We also give the skeleton of the Mizar proof of it.

```
theorem :: TDLAT_1:30
  for X being TopSpace holds
    LattStr ≪ Domains_of X,D-Union X,D-Meet X ≫ is C_Lattice;

proof let X be TopSpace;
    set L = LattStr ≪ Domains_of X,D-Union X,D-Meet X ≫ ;
  L is Lattice
      @proof end;
    then reconsider L as Lattice;
  L is 01_Lattice
      @proof end;
    then reconsider L as 01_Lattice;
  L is C_Lattice
      @proof end;
  hence thesis;
end;
```

This theorem says that the aggregate

```
    LattStr ≪ Domains_of X,Domains_Union X,Domains_Meet X ≫
```

is a structure of the type **C_Lattice**. So, we can introduce the functor **Domains_Lattice X**, which is a structure of the type **complemented Lattice** and it is equal to that aggregate. Precise Mizar definition has been done in [23] as follows.

```
definition let X be TopSpace;
  func Domains_Lattice X -> C_Lattice means :: TDLAT_1:def 4
    it = LattStr ≪ Domains_of X,Domains_Union X,Domains_Meet X ≫ ;
end;
```

Roughly speaking, **Domains_Lattice X** is the (complemented) lattice of all domains of the topological space X. Basic properties of this lattice are listed below.

```
theorem :: TDLAT_2:86

 the carrier of Domains_Lattice X = Domains_of X;


theorem :: TDLAT_2:87
  for a, b being Element of Domains_Lattice X
   for A, B being Element of Domains_of X st a = A & b = B holds
    a ⊔ b = Int(Cl(A ∪ B)) ∪ (A ∪ B) & a ⊓ b = Cl(Int(A ∩ B)) ∩ (A ∩ B);

theorem :: TDLAT_2:88
 ⊥(Domains_Lattice X) = ∅X & ⊤(Domains_Lattice X) = ΩX;

theorem :: TDLAT_2:89
  for a, b being Element of Domains_Lattice X
   for A, B being Element of Domains_of X st a = A & b = B holds
                                          a ⊑ b iff A c= B;
```

5.2 The lattices of closed domains and of open domains

The concepts of the lattices of closed domains and of open domains of a topological space have been introduced in Mizar similarly as the lattice of domains (see [23] and comp. [2], [16]). For given X of the type TopSpace (more for X of a type which widens to the type TopSpace) the aggregate

LattStr≪ Closed_Domains_of X,Closed_Domains_Union X,Closed_Domains_Meet X ≫

corresponding to the lattice of closed domains and the aggregate

LattStr ≪ Open_Domains_of X,Open_Domains_Union X,Open_Domains_Meet X ≫

corresponding to the lattice of open domains, are constructed. From the definitions of Closed_Domains_of X, CLD-Union X, CLD-Meet X, Open_Domains_of X, OPD-Union X and OPD-Meet X it follows that these aggregates are structures of the type LattStr (note that we use here the corresponding abbreviations of these functors). The definitions of the above mentioned functors have been done in Mizar as follows.

```
definition let X be TopSpace;

 func Closed_Domains_of X ->

       non-empty Subset-Family of the carrier of X means :: TDLAT_1:def 5

  it = {A where A is Subset of X : A is_closed_domain};

end;



definition let X be TopSpace;

 func Open_Domains_of X ->

       non-empty Subset-Family of the carrier of X means :: TDLAT_1:def 9

  it = {A where A is Subset of X : A is_open_domain};

end;

definition let X be TopSpace;
  func Closed_Domains_Union X ->
        BinOp of Closed_Domains_of X means :: TDLAT_1:def 6
    for A,B being Element of Closed_Domains_of X holds
      it.(A,B) = A ∪ B;
  synonym CLD-Union X;
end;

definition let X be TopSpace;
  func Open_Domains_Union X ->
        BinOp of Open_Domains_of X means :: TDLAT_1:def 10
    for A,B being Element of Open_Domains_of X holds
      it.(A,B) = Int(Cl(A ∪ B));
  synonym OPD-Union X;
```

```
end;

definition let X be TopSpace;
  func Closed_Domains_Meet X ->
         BinOp of Closed_Domains_of X means :: TDLAT_1:def 7
     for A,B being Element of Closed_Domains_of X holds
       it.(A,B) = Cl(Int(A ∩ B));
  synonym CLD-Meet X;
end;

definition let X be TopSpace;
  func Open_Domains_Meet X ->
         BinOp of Open_Domains_of X means :: TDLAT_1:def 11
     for A,B being Element of Open_Domains_of X holds
       it.(A,B) = A ∩ B;
  synonym OPD-Meet X;
end;
```

In [23], the following two theorems have been proved. Similarly as in the previous case, we give also the skeleton of the Mizar proof of the first one. The skeleton of the Mizar proof of the second one looks the same.

```
theorem :: TDLAT_1:34
  for X being TopSpace holds
    LattStr ≪ Closed_Domains_of X,CLD-Union X,CLD-Meet X ≫ is B_Lattice;

proof let X be TopSpace;
  set L = LattStr ≪ Closed_Domains_of X,CLD-Union X,CLD-Meet X ≫ ;
   L is Lattice
      @proof end;
    then reconsider L as Lattice;
   L is 01_Lattice
      @proof end;
    then reconsider L as 01_Lattice;
   L is C_Lattice
      @proof end;
    then reconsider L as C_Lattice;
   L is B_Lattice
      proof
       thus L is D_Lattice
         @proof end;
      end;
  hence thesis;
end;

theorem :: TDLAT_1:38
  for X being TopSpace holds
    LattStr ≪ Open_Domains_of X,OPD-Union X,OPD-Meet X ≫ is B_Lattice;
```

It follows that the aggregates given above are structures of the type B_Lattice. Thus we can introduce the functors Closed_Domains_Lattice X and Open_Domains_Lattice X as in the following definitions.

```
definition let X be TopSpace;
  func Closed_Domains_Lattice X -> B_Lattice means :: TDLAT_1:def 8
```

```
     it =
  LattStr ≪ Closed_Domains_of X,Closed_Domains_Union X,Closed_Domains_Meet X ≫;
end;

definition let X be TopSpace;
  func Open_Domains_Lattice X -> B_Lattice means :: TDLAT_1:def 12
    it = LattStr ≪ Open_Domains_of X,Open_Domains_Union X,Open_Domains_Meet X ≫;
end;
```

Roughly speaking, for a topological space X the functors `Closed_Domains_Lattice X` and `Open_Domains_Lattice X` have meaning of the (Boolean) lattice of all closed domains of X and of the (Boolean) lattice of all open domains of X, respectively. Basic properties of these lattices are listed below.

```
theorem :: TDLAT_2:94

 the carrier of Closed_Domains_Lattice X = Closed_Domains_of X;



theorem :: TDLAT_2:103

 the carrier of Open_Domains_Lattice X = Open_Domains_of X;



theorem :: TDLAT_2:95
 for a, b being Element of Closed_Domains_Lattice X
  for A, B being Element of Closed_Domains_of X st a = A & b = B holds
   a ⊔ b = A ∪ B & a ⊓ b = Cl(Int(A ∩ B));

theorem :: TDLAT_2:104
 for a, b being Element of Open_Domains_Lattice X
  for A, B being Element of Open_Domains_of X st a = A & b = B holds
   a ⊔ b = Int(Cl(A ∪ B)) & a ⊓ b = A ∩ B;

theorem :: TDLAT_2:96
  ⊥(Closed_Domains_Lattice X) = ∅X & ⊤(Closed_Domains_Lattice X) = ΩX;

theorem :: TDLAT_2:105
  ⊥(Open_Domains_Lattice X) = ∅X & ⊤(Open_Domains_Lattice X) = ΩX;

theorem :: TDLAT_2:97
 for a, b being Element of Closed_Domains_Lattice X
  for A, B being Element of Closed_Domains_of X st a = A & b = B holds
   a ≪ b iff A ⊆ B;

theorem :: TDLAT_2:106
 for a, b being Element of Open_Domains_Lattice X
  for A, B being Element of Open_Domains_of X st a = A & b = B holds
   a ≪ b iff A ⊆ B;
```

5.3 Sublattices of the lattice of domains

The lattices of all closed domains and of all open domains of a given topological space are sublattices of the lattice of all domains. It has been proved in [23].

First, it has been proved in fact that `the carrier of Closed_Domains_Lattice X` $\subseteq$ `the carrier of Domains_Lattice X` and `the carrier of Open_Domains_Lattice X` $\subseteq$ `the carrier of Domains_Lattice X`.

```
theorem :: TDLAT_1:31
  for X being TopSpace holds Closed_Domains_of X ⊆ Domains_of X;
```

```
theorem :: TDLAT_1:35
  for X being TopSpace holds Open_Domains_of X ⊆ Domains_of X;
```

Next, the following connections between lattice operations in the lattices of domains : `Domains_Lattice X`, `Closed_Domains_Lattice X` and `Open_Domains_Lattice X`, have been observed.

```
theorem :: TDLAT_1:32

 for A,B being Element of Closed_Domains_of X holds

            (CLD-Union X).(A,B) = (D-Union X).(A,B);
```

```
theorem :: TDLAT_1:33

 for A,B being Element of Closed_Domains_of X holds

              (CLD-Meet X).(A,B) = (D-Meet X).(A,B);
```

```
theorem :: TDLAT_1:36

 for A,B being Element of Open_Domains_of X holds

            (OPD-Union X).(A,B) = (D-Union X).(A,B);
```

```
theorem :: TDLAT_1:37

 for A,B being Element of Open_Domains_of X holds

              (OPD-Meet X).(A,B) = (D-Meet X).(A,B);
```

These properties have been reformulated as follows.

```
theorem :: TDLAT_1:39

 CLD-Union X = (D-Union X)|[:Closed_Domains_of X,Closed_Domains_of X:];
```

```
theorem :: TDLAT_1:40

 CLD-Meet X = (D-Meet X)|[:Closed_Domains_of X,Closed_Domains_of X:];



theorem :: TDLAT_1:42

 OPD-Union X = (D-Union X)|[:Open_Domains_of X,Open_Domains_of X:];



theorem :: TDLAT_1:43

 OPD-Meet X = (D-Meet X)|[:Open_Domains_of X,Open_Domains_of X:];
```

Thus according to the definition of sublattices, given at the end of subsection 4.1, we can get immediately the following theorems (see [23]).

```
theorem :: TDLAT_1:41

 Closed_Domains_Lattice X is SubLattice of Domains_Lattice X;



theorem :: TDLAT_1:44

 Open_Domains_Lattice X is SubLattice of Domains_Lattice X;
```

5.4 Completeness of the lattices of domains

In [12], it is proved that the lattice of all domains of a given topological space is complete. First, the following theorem has been formulated. We give also the skeleton of the Mizar proof of it.

```
theorem :: TDLAT_2:90
 for F being Subset of Domains_Lattice X
  ex a being Element of Domains_Lattice X st F is_less_than a &
   for b being Element of Domains_Lattice X st F is_less_than b holds a ≤ b;

proof
 let G be Subset of Domains_Lattice X;
    G ⊆ the carrier of Domains_Lattice X by AXIOMS:2;
  then INC: G ⊆ Domains_of X by TDLAT_2:86;
  then reconsider F = G as Subset-Family of X by TOPS_2:3;
 set A = (union F) ∪ (Int Cl(union F));
   F is domains-family by INC,TDLAT_2:65;
  then A is_domain by TDLAT_2:68;
  then A ∈ {C where C is Subset of X : C is_domain};
  then A ∈ Domains_of X by TDLAT_1:def 1;
  then A ∈ the carrier of Domains_Lattice X by TDLAT_2:86;
  then reconsider a = A as Element of Domains_Lattice X;
```

```
  take a;
    LS: G is_less_than a
          @proof
          end;
    for b being Element of Domains_Lattice X st G is_less_than b holds a ≤≤ b
      @proof
      end;
  hence thesis by LS;
end;
```

Next, according to the definition of complete lattices, given in the subsection 4.3, from this theorem we can get immediately the following one (see [23]).

```
theorem :: TDLAT_2:91

 Domains_Lattice X is complete;
```

In the same way the completeness of the lattices of closed domains and of open domains has been proved (comp. [2], [16]).

```
theorem :: TDLAT_2:98
  for F being Subset of Closed_Domains_Lattice X
   ex a being Element of Closed_Domains_Lattice X st
    F is_less_than a &
     for b being Element of Closed_Domains_Lattice X st
      F is_less_than b holds a ≤≤ b;
```

```
theorem :: TDLAT_2:107
  for F being Subset of Open_Domains_Lattice X
   ex a being Element of Open_Domains_Lattice X st
    F is_less_than a &
     for b being Element of Open_Domains_Lattice X st
      F is_less_than b holds a ≤≤ b;
```

```
theorem :: TDLAT_2:99

 Closed_Domains_Lattice X is complete;
```

```
theorem :: TDLAT_2:108

 Open_Domains_Lattice X is complete;
```

6 The lattice of domains of an extremally disconnected space

In [11], a condition whether the lattice of all domains of a given topological space is Boolean has been done. Namely, the lattice of all domains of a topological space X is Boolean if and only if X is extremally disconnected. The aim of this section is to present this result. We start with the properties of the lattice of domains of extremally disconnected spaces.

```
reserve X for extremally_disconnected TopSpace;


theorem :: TDLAT_3:41

 Domains_of X = Closed_Domains_of X;


theorem :: TDLAT_3:42

 D-Union X = CLD-Union X & D-Meet X = CLD-Meet X;


theorem :: TDLAT_3:43

 Domains_Lattice X = Closed_Domains_Lattice X;


theorem :: TDLAT_3:44

 Domains_of X = Open_Domains_of X;


theorem :: TDLAT_3:45

 D-Union X = OPD-Union X & D-Meet X = OPD-Meet X;


theorem :: TDLAT_3:46

 Domains_Lattice X = Open_Domains_Lattice X;


theorem :: TDLAT_3:47
  for A, B being Element of Domains_of X holds
    (D-Union X).(A,B) = A ∪ B & (D-Meet X).(A,B) = A ∩ B;

theorem :: TDLAT_3:48
  for a, b being Element of Domains_Lattice X
    for A, B being Element of Domains_of X st a = A & b = B holds
                                   a ⊔ b = A ∪ B & a ⊓ b = A ∩ B;
```

In [11] the following lattice-theoretic characterization of extremally disconnected spaces has been obtained. It says that a topological space is extremally disconnected if and only if its lattice of all domains is modular. We give below the Mizar proof of it.

```
reserve X for TopSpace;
```

```
theorem :: TDLAT_3:51

 X is extremally_disconnected iff Domains_Lattice X is M_Lattice;


proof
     CAR: Domains_Lattice X =
         LattStr ≪ Domains_of X,D-Union X,D-Meet X ≫ by TDLAT_1:def 4;

   thus X is extremally_disconnected implies Domains_Lattice X is M_Lattice

    proof

     assume X is extremally_disconnected;

      then Domains_Lattice X = Open_Domains_Lattice X by TDLAT_3:46;

      then Domains_Lattice X is D_Lattice by LATTICES:def 14;

     hence Domains_Lattice X is M_Lattice by LATTICES:35;

    end;
 assume MD: Domains_Lattice X is M_Lattice;

  assume not X is extremally_disconnected;

   then consider D being Subset of X such that

     DOM: D is_domain and NEQ: Int Cl D <> Cl Int D by TDLAT_3:36;

    set A = Int Cl D, C = Cl Int D, B = C';

     ODCD: A is_open_domain & C is_closed_domain by TDLAT_1:23,22;

    then ACD: A is_domain & C is_domain by TOPS_1:106,107;

        B' is_closed_domain by ODCD,PRE_TOPC:20;

     then B is_open_domain by TOPS_1:101;

    then B is_domain by TOPS_1:107;

     then B ∈ {E where E is Subset of X : E is_domain};
     then reconsider b = B
            as Element of Domains_Lattice X by CAR,TDLAT_1:def 1;
        _657_:  A ∈ {E where E is Subset of X : E is_domain} by ACD;
       then AD: A ∈ Domains_of X by TDLAT_1:def 1;
     reconsider a = A
            as Element of Domains_Lattice X by CAR,TDLAT_1:def 1,_657_;
        _658_:  C ∈ {E where E is Subset of X : E is_domain} by ACD;
       then CD: C ∈ Domains_of X by TDLAT_1:def 1;
     reconsider c = C
```

```
                    as Element of Domains_Lattice X by CAR,TDLAT_1:def 1,_658_;
             A ⊆ D & D ⊆ C by DOM,TOPS_1:def 6;
           then A ⊆ C by BOOLE:29;
        then MODEQ: a ≤≤ c by AD,CD,TDLAT_2:89;
           ODE: A = Int Cl A by ODCD,TOPS_1:def 8;
           EIN: B ∩ C = ∅X by TOPS_1:12;
           AEM: A ∪ ∅X = A ∪ ∅ by PRE_TOPC:11
                       .= A by BOOLE:60;
          b ⊓ c = Cl(Int(B ∩ C)) ∩ (B ∩ C) by CAR,TDLAT_2:87
               .= ∅X by EIN,TOPS_1:4;
        then LS: a ⊔ (b ⊓ c) = Int(Cl A) ∪ A by CAR,TDLAT_2:87,AEM
                          .= A by ODE,BOOLE:62;
           CDE: C = Cl Int C by ODCD,TOPS_1:def 7;
                                      TR: B = Int (Int D)' by TDLAT_3:4
                                         .= Int Cl D' by TDLAT_3:3;
                          CDCL: Cl D is_closed by PCOMPS_1:4;
          Cl(A ∪ B) = Cl A ∪ Cl B by PRE_TOPC:50
                    .= Cl Int(Cl D ∪ Cl D') by TR,CDCL,TDLAT_1:6
                    .= Cl Int Cl(D ∪ D') by PRE_TOPC:50
                    .= Cl Int Cl ΩX by TOPS_1:11
                    .= Cl Int ΩX by TOPS_1:27
                    .= Cl ΩX by TOPS_1:43
                    .= ΩX by TOPS_1:27;
          then a ⊔ b = Int(ΩX) ∪ (A ∪ B) by CAR,TDLAT_2:87
                    .= ΩX ∪ (A ∪ B) by TOPS_1:43
                    .= ΩX by TOPS_1:2;
        then (a ⊔ b) ⊓ c = Cl(Int(ΩX ∩ C)) ∩ (ΩX ∩ C) by CAR,TDLAT_2:87
                       .= Cl(Int C) ∩ (ΩX ∩ C) by TOPS_1:3
                       .= Cl(Int C) ∩ C by TOPS_1:3
                       .= C by CDE,BOOLE:65;
  hence contradiction by NEQ,LS,MODEQ,MD,LATTICES:def 6;
end;
```

From this theorem it follows the other lattice-theoretic characterization of extremally disconnected spaces (comp. [11]). Namely, a topological space is extremally disconnected if and only if its lattice of all domains is Boolean. We give also the Mizar proof of this theorem.

```
theorem :: TDLAT_3:55

 X is extremally_disconnected iff Domains_Lattice X is B_Lattice;


 proof

   thus X is extremally_disconnected implies Domains_Lattice X is B_Lattice

    proof

      assume X is extremally_disconnected;

       then Domains_Lattice X = Open_Domains_Lattice X by TDLAT_3:46;
```

```
  hence thesis;

 end;

assume Domains_Lattice X is B_Lattice;

 then Domains_Lattice X is D_Lattice by LATTICES:def 14;

 then Domains_Lattice X is M_Lattice by LATTICES:35;

hence X is extremally_disconnected by TDLAT_3:51;

end;
```

Remarks. Because of the clustering constructions in Mizar, the following (true) sentence requires a single direct justification only by the theorem given above.

```
for X being discrete TopSpace holds

 Domains_Lattice X is Boolean Lattice by TDLAT_3:55;
```

But the following Mizar sentence is incorrect.

```
for X being almost_discrete TopStruct holds

 Domains_Lattice X is modular by TDLAT_3:51;

::>                    *103
```

The Mizar analyzer reports an error 103: `Unknown functor`.

Acknowledgments

References

[1] G. Bancerek, *Complete lattices*, Formalized Math., **2** (1991), 719–725, MML: LATTICE3.

[2] G. Birkhoff, *Lattice Theory*, Providence., Rhode Island, New York, 1967.

[3] E. Bonarska, *An Introduction to PC Mizar*, Mizar Users Group. Foundation Philippe le Hodey., Brussels, 1990.

[4] S. Burris and H. P. Sankappanavar, *A Course in Universal Algebra*, Springer-Verlag., New York, 1981.

[5] M. Chmur, *The lattice of natural numbers and the sublattice of it. The set of prime numbers*, Formalized Math., **2** (1991), 453–459, MML: NAT_LAT.

[6] A. Darmochwał and B. Padlewska, *Topological spaces and continuous functions*, Formalized Math., **1** (1990), 223–230, MML: PRE_TOPC.

[7] A. Darmochwał and M. Wysocki, *Subset of topological spaces*, Formalized Math., **1** (1990), 231–237, MML: TOPS_1.

[8] R. Engelking, *General Topology*, Monografie Matematyczne Vol.**60**, PWN - Polish Scientific Publishers, Warsaw 1977.

[9] G. Grätzer, *General Lattice Theory*, Akademie - Verlag., Berlin 1978.

[10] Y. Isomichi, *New concepts in the theory of topological space - supercondensed set, subcondensed set, and condensed set*, Pacific Journal of Math., **38** (1971), 657–668.

[11] Z. Karno, *The lattice of domains of an extremally disconnected space*, Formalized Math., **3** (1992), MML: TDLAT_3.

[12] Z. Karno and T. Watanabe, *Completeness of the lattice of domains of a topological space*, Formalized Math., **3** (1992), 71–79, MML: TDLAT_2.

[13] K. Kuratowski, *Topology, Vol. I*, PWN - Polish Scientific Publishers and Academic Press., Warsaw, New York and London, 1966.

[14] K. Kuratowski, *Topology, Vol. II*, PWN - Polish Scientific Publishers and Academic Press., Warsaw, New York and London, 1968.

[15] K. Kuratowski, *Sur l'opération $\overline{A}$ de l'Analysis Situs*, Fundamenta Math., **3** (1922), 182–199., (*Selected Papers*, PWN – Polish Scientific Publishers., Warsaw 1988, pp. 95–112).

[16] K. Kuratowski and A. Mostowski, *Set Theory (with an introduction to descriptive set theory)*, Studies in Logic and The Foundations of Mathematics Vol.**86**, PWN - Polish Scientific Publishers and North-Holland Publishing Company., Warsaw-Amsterdam, 1976.

[17] H. MacNeille, *Partially ordered sets*, Trans. Amer. Math. Soc., **42** (1937), 416–460.

[18] P. Rudnicki, *An overview of the Mizar project*, Proceedings of the 1992 Workshop on Types for Proofs and Programs, Eds. B. Nordström, K. Petersson and G. Plotkin, pp. 311–332.

[19] M. H. Stone, *Application of the theory of Boolean rings to General Topology*, Trans. Amer. Math. Soc., **41** (1937), 375–481.

[20] M. H. Stone, *Algebraic characterizations of special Boolean rings*, Fundamenta Math., **29** (1937), 223–303.

[21] A. Tarski, *Über additive und multiplikative Mengenkörper und Mengenfunktionen*, Comptes Rendus Soc. Sci. Letter. Varsovie, **30** (1937), 151–181., (*Collected Papers, Vol. 2*, Birkhäuser, Boston 1985, pp. 289–322).

[22] A. Trybulec, *Some features of the Mizar language*, (these proceedings).

[23] T. Watanabe, *The lattice of domains of a topological space*, Formalized Math., **3** (1992), 41–46, MML: TDLAT_1.

[24] S. Żukowski, *Introduction to lattice theory*, Formalized Math., **1** (1990), 215–222, MML: LATTICES.