Computer-Assisted Lattice Theory Exploration in Mizar

Adam Grabowski
University of Białystok, Poland
adam@math.uwb.edu.pl

Small TYPES Workshop
Chambery, 13–14.04.2005
Math. Knowledge Storing

- Databases need to be checked for correctness and coherence
- Language – preferably close to the natural
- Type information should be carried
- Reusability is important and desirable
- Proofs
  - reasonably short
  - possibly translated from/into other assistants
- Choice of the topic
  - general topology
  - lattice theory
Lattices in the MML: Structures

definition

struct (1-sorted) \-SemiLattStr

(# carrier -> set,
  L_meet -> BinOp of the carrier #);

struct (1-sorted) \/-SemiLattStr

(# carrier -> set,
  L_join -> BinOp of the carrier #);

struct (\/-SemiLattStr,\/-SemiLattStr) LattStr

(# carrier -> set,
  L_join -> BinOp of the carrier,
  L_meet -> BinOp of the carrier #);
end;
Lattices in the MML: Attributes

Axiomatization: extending structures via adjectives:

- join-commutative, join-associative, join-absorbing
- dually, for meet (infimum) operation
- modular, distributive, bounded, implicative, complemented etc.

```plaintext
definition let IT be non empty LattStr;
  attr IT is meet-absorbing means
  :: LATTICES:def 8
    for a,b being Element of IT holds
      (a "/" b) "\" b = b;
end;
```

Lattices in the MML: Clusters

• Boolean lattices are just distributive bounded complemented lattices by definition

• Automatic acceptance e.g. that every distributive lattice is modular (when proving and hence registering conditional cluster) so that all theorems formulated for modular lattices are accepted also for distributive ones

Basic type

mode Lattice is
  join-commutative join-associative meet-absorbing
  meet-commutative meet-associative join-absorbing
  (non empty LattStr);
The Lattice of Substitutions

The carrier and normalization operator $\mu$:

$$\mathcal{V} := \{ A \in \text{Fin PF}(V, C) : \forall s, t \in A \; s \subseteq t \Rightarrow s = t \}$$

$$A \sqcap B := \mu \{ s \cup t : s \in A \land t \in B \land s \cup t \text{ is a function} \}$$

$$A \sqcup B := \mu (A \cup B)$$

$$\mathcal{K}_\mathcal{V} = \langle \mathcal{V}, \sqcup, \sqcap \rangle$$

$$\bot_{\mathcal{K}_\mathcal{V}} = \emptyset \quad \top_{\mathcal{K}_\mathcal{V}} = \{ f_{\emptyset} \}$$
Substitutions & Formula Trees

```
p\lor q
  \downarrow
p \land (p \lor q)
  \uparrow
p\land q

\{p/0\} \lor \{q/0\}
  \downarrow
\{pq/00\}

\{p/0\} \lor \{p/1\}
  \downarrow
\emptyset
```
The Ordering of Substitutions

Natural ordering

\[ a \leq b \text{ iff } a \sqcap b = b \text{ (or } a \sqcup b = a) \]

Sets of substitutions’ ordering

\[ A \leq B \text{ iff } \forall f \in A \ \exists g \in B \ g \subseteq f \]

Smyth ordering on Scott powerdomains

\[ A \leq B \text{ iff } \forall f \in B \ \exists g \in A \ g \subseteq f \]
Examples, Wroński Operation

2 vars, 1 const

1 var, 2 consts
The Incompleteness

\[ \alpha_1 = \left\{ \begin{array}{c|c|c|c|c} \hline 1 & 2 & 1 & 3 \\ \hline 0 & 0 & 0 & 0 \\ \end{array} \right\}, \]

\[ \alpha_2 = \left\{ \begin{array}{c|c|c|c|c|c|c} \hline 1 & 2 & 1 & 3 & 4 & 1 & 3 & 5 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \end{array} \right\}, \]

\[ \alpha_3 = \left\{ \begin{array}{c|c|c|c|c|c|c|c|c} \hline 1 & 2 & 1 & 3 & 4 & 1 & 3 & 5 & 6 & 1 & 3 & 5 & 7 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array} \right\} \]

There is no infimum of \( A = \{ \alpha_1, \alpha_2, \alpha_3, \ldots \} \)

Formalizing counterexamples is also important!
Examples of Mizared Objects

\[ \mathcal{V} := \{ A \in \text{Fin PF}(V, C) : \forall s, t \in A \; s \subseteq t \Rightarrow s = t \} \]

**definition** let \( V, C \) be set;

\[
\text{func SubstitutionSet}(V, C) \rightarrow \text{Subset of Fin PFuncs}(V, C) \; \text{equals} \; \text{SUBSTLAT: def 1}
\]

\[
\{ A \text{ where } A \text{ is Element of Fin PFuncs } (V,C) : \\
\text{ for } s, t \text{ being Element of } A \text{ st } s \sqsubseteq t \text{ holds } s = t \} \;
\]

**end;**

\[
\alpha \hat{\rightarrow} \beta = \bigsqcup_{\gamma \subseteq \alpha} \neg \gamma \sqcap ((\alpha \setminus \gamma) \Rightarrow \beta).
\]

**theorem** :: HEYTING2:33

\[
u \Rightarrow v = \text{FinJoin}(\text{SUB } u, \\
(\text{the L_meet of SubstLatt } (V, C)).:(\text{pseudo_compl}(V, C), \\
\text{StrongImpl}(V, C)[:](\text{diff } u, v))).
\]
Formalizing Lattice Problems

- alternative axiomatizations of Boolean algebras
- Robbins problem
- Sheffer stroke algebras
- short single axioms
  - in terms of disjunction and negation
  - in terms of Sheffer stroke
- ortholattices vs. orthocomplemented posets
- Rough Set Theory

All these resulted in Mizar articles
The Robbins Problem

\[ \langle L, +, ' \rangle \]

\[ a + b = b + a \]
\[ a + (b + c) = (a + b) + c \]

Huntington equation:

\[ (a' + b')' + (a' + b)' = a \]

Robbins equation:

\[ ((a + b)' + (a + b')')' = a \]

Are Robbins algebras Boolean?
The Solution

Robbins algebras and Boolean are the same!

registration
  cluster Robbins de_Morgan ->
    Boolean preOrthoLattice;
  cluster Boolean -> Robbins
    (well-complemented preOrthoLattice);
end;

definition let L be non empty OrthoLattStr;
  attr L is de_Morgan means
    :: ROBBINS1:def 23
    for x, y being Element of L holds
      x "/\" y = (x` "\" y`)`;
end;
Short Single Axioms for BAs

disjunction and negation

definition let L be non empty ComplLattStr;
attr L is satisfying_DN_1 means
for x, y, z, u being Element of L holds
  (((x + y)' + z)' + (x + (z' + (z + u)'))')' = z;
end;

registration
  cluster satisfying_DN_1 de_Morgan ->
    Boolean preOrthoLattice;
end;

Sheffer stroke algebras

definition let L be non empty ShefferStr;
attr L is satisfying_Sheffer_3 means
for x, y, z being Element of L holds
  (x|(y|z)) | (x|(y|z)) = ((y|y)|x) | ((z|z)|x);
end;
Merging Structures

definition

struct (\-SemiLattRelStr, \-SemiLattRelStr, LattStr)
  LattRelStr
  (# carrier -> set,
   L_join, L_meet -> (BinOp of the carrier),
   InternalRel -> Relation of the carrier #);
end;

We may benefit from apparatus of:

• relational structures: ORDERS series
  (InternalRel selector)

• lattices with binary operations: LATTICE series
  (all other selectors)
Structure Hierarchy (A Part of)
Ortholattices

definition let L be non empty \/-SemiLattRelStr;
  attr L is naturally_sup-generated means
  :: ROBBINS3:def 10
    for x, y being Element of L holds
    x <= y iff x \_\_ y = y;
end;

registration let L be Ortholattice;
  cluster -> Orthocomplemented
    (naturally_sup-generated CLatAugmentation of L);
end;

registration
  cluster Boolean well-complemented Lattice-like ->
    involutive with_Top de_Morgan
    (non empty OrthoLattStr);
end;
Rough Set Theory

\[ X_R = \{ x \in U : [x]_R \subseteq X \} \]

definition let A be Tolerance_Space;
    let X be Subset of A;
    func LAp X \to Subset of A equals :: ROUGHS_1:def 4
        { x where x is Element of A :
            Class (the InternalRel of A, x) c= X };
end;

definition let A be Tolerance_Space;
    let X be Subset of A;
    attr X is rough means :: ROUGHS_1:def 7
        BndAp X <> {};
end;
Proof Transformation

1. Call OTTER to produce proof object
2. External Lisp script ott2miz.el produces “untyped” Mizar source, MML-independent
3. Mizar auxiliary utilities remove unnecessary proof steps, etc. obvious for the Mizar checker
4. Type hierarchy should be created by hand or reused from the MML
5. Labels and variables renaming preferred

Automatically derived proofs can result in useful auxiliary lemmas
Proof Transform. – Otter Input

Derive idempotency of supremum from lattice axioms.

```
set(auto).
set(build_proof_object_2).
op(400, xfx, ^).
op(400, xfx, v).

list(usable).
x = x.

A v A != A.

x ^ (x v y) = x.
x v (x ^ y) = x.
end_of_list.
```
Proof Transform. – Output

;; BEGINNING OF PROOF OBJECT
(
(1 (input) (not (= (v (A) (A)) (A))) (1))
(2 (input) (= (^ v0 (v v0 v1)) v0) (3))
(3 (input) (= (v v0 (^ v0 v1)) v0) (5))
(4 (instantiate 2 ((v0 . v64))) (= (^ v64 (v v64 v1)) v64) NIL)
(5 (instantiate 3 ((v0 . v64)(v1 . (v v64 v1)))) (= (v v64 (^ v64 (v v64 v1))) (v v64 (v v64 v1))) NIL)
(6 (paramod 4 (1) 5 (1 2)) (= (v v64 v64) v64) NIL)
(7 (instantiate 6 ((v64 . v0))) (= (v v0 v0) v0) (7))
(8 (instantiate 7 ((v0 . (A)))) (= (v (A) (A)) (A)) NIL)
(9 (resolve 1 () 8 ()) false (9))
)

;; END OF PROOF OBJECT
Proof – Mizar without MML

( not (A |_| A) = A) &
(for v1,v0 holds (v0 |^| (v0 |_| v1)) = v0) &
(for v1,v0 holds (v0 |_| (v0 |^| v1)) = v0) implies contradiction

proof

assume A1: not (A |_| A) = A;
assume A2: for v1,v0 holds (v0 |^| (v0 |_| v1)) = v0;
assume A3: for v1,v0 holds (v0 |_| (v0 |^| v1)) = v0;
A4: for v1,v64 holds (v64 |^| (v64 |_| v1)) = v64 by A2;
A5: for v1,v64 holds (v64 |_| (v64 |^| (v64 |_| v1))) = v64 by A3;
A6: for v64 holds (v64 |_| v64) = v64

proof

now let v1,v64;

(v64 |^| (v64 |_| v1)) = v64 by A4;

hence (v64 |_| v64) = v64 by A5;
end;

hence thesis;
end;
A7: for v0 holds (v0 |_| v0) = v0 by A6;
A8: (A |_| A) = A by A7;

hence A9: contradiction by A1;
end;
Proof – Mizar without MML

(after RELPREM, RELINFER, TRIVDEMO, etc.)

theorem
(for v1,v0 holds (v0 |ˆ| (v0 |_| v1)) = v0) &
(for v1,v0 holds (v0 |_| (v0 |^| v1)) = v0) implies
   for x being set holds x |_| x = x

proof
   assume A2: for v1,v0 holds (v0 |^| (v0 |_| v1)) = v0;
   assume A3: for v1,v0 holds (v0 |_| (v0 |^| v1)) = v0;
   let v64;
   AA: v64 = v64 |^| (v64 |_| v64) by A2; then
      v64 |^| (v64 |_| v64) = v64 |_| v64 by A3,A2;
      hence thesis by A1;
   end;
Proof – Mizar with MML

environ

vocabularies ROBBINS3, LATTICES;
notations TARSKI, XBOOLE_0, SUBSET_1, STRUCT_0, LATTICES, ROBBINS3;
constructors TARSKI, SUBSET_1, LATTICES, STRUCT_0, ROBBINS3;
registrations LATTICES;
theorems LATTICES;

begin

reserve L for Lattice;
reserve x for Element of L;

theorem
\[ x |\_\_| x = x \]
proof
A1: \[ x = x |^\_| (x |\_\_| x) \] by LATTICES: def 9; then
\[ x |^\_| (x |\_\_| x) = x |\_\_| x \] by LATTICES: def 8;

hence thesis by A1;
end;
Theory Exploration Activities

- formalization of classical results (SHEFFER1, LATSUM_1)
- new research using Mizar (HEYTING2)
- formal description of checker (SUBSTLAT)
- formalization of recent mathematics (ROBBINS1, ROUGHS_1)
- cooperation with provers (SHEFFER2)
- unification of independent disciplines (ROBBINS3)
- lattices as a suitable tool for describing non-lattice-theoretical notions (ROUGHS_1)
- revisions of the MML (redesign of lattices and orders, new attributes and clusters)
Mizar Capabilities

- mathematicians develop large disciplines, provers can help when solving small puzzles
- apparatus of structures and attributes
- possibility of introducing synonyms to fulfill authors’ needs
- mechanism of revisions to correct earlier authors’ developments – done mainly by Library Committee of the Association of Mizar Users
- Properties: e.g. automatization of commutativity can shorten proofs significantly
- synonyms for lattice operations when necessary
Conclusions

- theory exploration is a good testbed for the language
- Mizar is a suitable machine-assistant for human
  the language: structures, adjectives, clusters
- general lattice theory and general topology as the leading disciplines in the MML
- projects: “Compendium of Continuous Lattices”, Jordan Curve Theorem
- XML representation needed, partially available
- the aim: to attract more mathematicians suggesting the improvement of the language and developing the MML
Future Work

• to provide easier way to generalization of types (not all lattice axioms are necessary)
• box products of lattices
• challenge: formalize one of the handbooks:
  • Grätzer: “General Lattice Theory”
  • Davey & Priestley: “Introduction to Lattices and Order”
  • Birkhoff: “Lattice Theory”
  • Koppelberg: “Handbook of Boolean Algebras”
• lattices of rough sets
• lattices in education