Rough Set Theory from a Math-Assistant Perspective

Adam Grabowski* and Magdalena Jastrzębska

Institute of Mathematics, University of Białystok
ul. Akademicka 2, 15-267 Białystok, Poland
adam@math.uwb.edu.pl, magjas0@poczta.onet.pl

Abstract. In the paper, we draw a perspective of the computer-assisted theory exploration within rough set theory. We examine two well-known approaches to the topic, drawing some paradigms for a machine math-assistant to be feasible tool any researcher can use to verify his own results. Some features of a Mizar language chosen for the verification task are also presented.

1 Introduction

This paper is a survey of the development of rough set theory from a machine proof-assistant viewpoint, and a brief summary of basic results encoded in the computer-checked repository of mathematical knowledge formalized using the Mizar system. By *formalization* we mean the encoding of mathematics in a formal language sufficiently detailed for a computer program to verify the correctness. The greatest projects of this kind of a pre-computer age were Whitehead and Russell's "Principia Mathematica" and project Bourbaki. "Checking Landau's 'Grundlagen' in the Automath system" by Jutting (1977) was the first significant step of translating human efforts in a machine-checkable language.

The need of the computer verification of hardware and software emerged pretty recently. After the bug in the first Pentium processor was discovered in 1994, the Intel company established a special group of people doing research in the field of the hardware verification. But the issue of the uncertainty of results appears not only in the industry – also academia can face this problem, especially when publishing is taken into account. As the referees can be uncertain if a proof is really correct, the review procedure can take months, but the situation gets even more frustrating when we take into account the case of Thomas Hales, who has been waiting for five years to hear whether the mathematical community has accepted his 1998 proof of Kepler's conjecture that the most efficient way to pack equal-size spheres is to stack them in the usual pyramid. In 2003 a review panel of world experts appointed by the journal *Annals of Mathematics* finally declared that, whereas they had not found any irreparable error in the proof, they were still not sure that it was correct. The journal finally decided

* Partial support for this work provided by the EU FP6 IST grant TYPES (Types for Proofs and Programs) No. 510996.

M. Kryszkiewicz et al. (Eds.): RSEISP 2007, LNAI 4585, pp. 152–161, 2007.

to publish Hales's proof, but the disclaimer saying they were not sure that it was right, was added. Then Hales started the FlySpeck project to formalize his proof with the help of a computer and hence a new paradigm of using machine proof-checker – not as the experimental tool but to solve real-life problems – was confirmed.

The paper is organized as follows. The next section contains a brief summary of basic notions of RST formalized in the Mizar language, as well as some highlights of the system used for this purpose. Section 3 provides the discussion of various approaches to rough sets while in the fourth section we deal with the more general model for I-sets, i.e. interval sets, and then we discuss rough sets from a lattice-theoretical point of view. In Section 6 we sketch some advantages of machine support in the process of knowledge exploring. The paper ends with some conclusions and plans for future work.

2 A Primer of Rough Set Theory, Formal Approach

The previous century has brought many automated theorem proving projects (and so the work of Pawlak in this direction reflected contemporary trends) and also a few realizations of the idea of machine-checking proofs for their correctness. As a first, probably most notable, we can point out the aforementioned work of Jutting in the de Bruijn's system Automath. Usually, a researcher which is not well acquainted with automated theorem proving, gets know only about very large and successful formalization projects. The most impressive (and/or probably also best advertised) examples were the solution of the Robbins problem which was open for over sixty years, solved by automated equational theorem prover EQP/Otter, Four Color Theorem with the proof done in Coq, the Jordan Curve Theorem recently completed in HOL and later in Mizar.

Andrzej Trybulec, the designer of the Mizar system, in a private communication admitted that the person who influenced positively his early researches on the translation from a natural mathematical vernacular into a machine-understandable language (so, also the development of the Mizar language), was Zdzisław Pawlak. When they met in the seventies of the previous century and discussed a bit the problems emerging somewhere at the intersection of the human–computer spheres, Professor Pawlak suggested the application for a grant at IPI PAN (Polish acronym of the Institute of Computer Science of the Polish Academy of Sciences, although the name was yet slightly different). Trybulec and his group followed the advice of the designer of a first Polish digital computer and the unquestionable authority in the field of the young emerging discipline of the computer science (exploring topics of the automated reasoning and the mathematical model of a computer, both reflected in Trybulec's system), they did so, succeded and eventually got the financing. It was extremely important, because as yet it can be remembered, the access to a computing machine, necessary for experiments with automated reasoning, was rather complicated and highly cost-consuming those days. For sure, the language evolved from its predicative form which was popular some thirty years ago, to somewhat closer to

its natural original. Also some physical bounds vanished – computer parameters are better and better, which allows us to face the problems inaccessible a couple decades ago. Some paradigms about human–computer interaction remained unchanged, though; we guess that many visionary ideas Zdzisław Pawlak had in mind, were influential for lots of people involved in the computer science, treated in the very general setting.

The Mizar language is a formal language close to the vernacular used in mathematical publications. An implemented Mizar verifier is available for checking correctness of Mizar texts according to Jaśkowski natural deduction. The perpetual development of the Mizar system (see [17]) has resulted in the Mizar Mathematical Library (MML) – a centrally maintained library of formalized mathematics based on Tarski-Grothendieck set theory which is a variant of ZFC.

The MML is organized as a cross-linked collection of the items called Mizar *articles*. As of the time of writing, there are 959 articles in the whole library, occupying 70 MB, containing 43149 theorems and 8185 definitions. It is commonly considered the biggest library of computer proof-checked mathematics (possessing e.g., recent proof of the Jordan Curve Theorem) and as such is also the subject of the research of data-miners (e.g., TPTP – Thousands of Problems for Theorem Provers). However not yet based on GNU license, the system is free, available for most popular platforms: MS Windows, Unixes, and MacOS on PowerPC. System requirements for installing both binaries and database are rather modest; about 200 MB of disk space to uncompress the full distribution.

2.1 Towards Formal Approximation Spaces

According to the classical paper of Pawlak [9], rough sets are based on the equivalence relations. Shortly thereafter, there were considered in the literature more general approaches, e.g. transitivity of the indiscernibility relation was dropped (see [7,11,10] for some paths of research, not only without transitivity).

Some of the natural properties are true only for the case of equivalence relations, which may make the theorems heterogeneous in some sense – some of them will require more complex assumptions under which they remain valid. So to keep his/her work more unified in style, the author could decide to formulate all of them in terms of approximation spaces. This approach, although transparent from the user's perspective, is hardly acceptable from the viewpoint of the knowledge reusability. We will write e.g. the upper approximation of the subset A of a universe U with respect to indiscernibility relation I classically as

$$upp_I(A) = \{x \in U : [x]_I \cap A \neq \emptyset\},$$

but using some hidden arguments (we take into account the space R with determined universe U and I being not necessarily an equivalence relation).

```
definition let X be Tolerance_Space, A be Subset of X;
  func UAp A -> Subset of A equals :: ROUGHS_1:def 5
    { x where x is Element of X :
      Class (the InternalRel of X, x) meets A };
end;
```

It should be noticed here that to start with rough sets we formalized the definition of approximation spaces based on tolerance relations whenever possible, we introduced membership functions with selected basic properties, and also provided the definition of rough sets. Some lines are devoted to various predicates of rough inclusion and rough equality. This primary development can be browsed under MML Identifier `ROUGHS_1` from the Mizar home page [1].

2.2 The State of the Art

A more or less formally axiomatized view for rough sets is not a novelty: Bryniarski [1] or Yao [12] are good representatives, not to enumerate yet classical [9] and [11]. But if we require the possibility of proof checking by the computer, the choice is not that wide although the idea of automatic correctness checking is also known. This approach presents relative uniformity – usually there is a unique definition because the Library Committee which takes care of the collection of articles does not allow for duplication of concepts and the library users report such repetitions. But also heterogeneity is not excluded completely – one can introduce constructions called redefinitions, which can result in having two approaches effectively benefitting from their equivalence. A good example is the definition of the rough equality of sets which is, on the one hand, the simultaneous equality of their upper and lower approximations, on the other hand – the conjunction of two rough inclusions.

The MML is roughly divided in three parts – concrete (based on pure set theory), abstract (where structures, including algebraic ones, as e.g. groups, lattices, vector and topological spaces are defined), and that devoted to the formalization of random access Turing machines, i.e. mathematical model of a computer, first decisions which had to be made were how to define approximations; because it was pretty clear approximation spaces should have been put in the abstract part of the library due to its strong algebraic flavour.

The correspondence between the very basic notions of the rough set theory chosen and their formal translated counterparts is given in Table 1. In the table the dot sign "`.`" stands for the application of a membership function, the brackets are used mainly for grouping multiple arguments of Mizar functors (i.e. a kind of language functions), formulas, etc. During the process of the automatic translation into the natural language (resulting also in the LaTeX source) Mizar functors are not typeset verbatim, but translated either in a way proposed by the author, or according to some simple transition rules applied automatically.

3 Two Views for Rough Sets

As widely known, the central notion of a rough set does not have its formal definition uniquely determined. We mean here two set-oriented views for rough sets. The first one is classical, due to Pawlak (P-sets). The other one (sometimes called I-sets in the literature for Iwiński [6]) is based on pairs of definable subsets. The more thorough discussion about such classification can be found in [12].

[1] `http://www.mizar.org/`

Table 1. The correspondence between natural language and Mizar objects

the notion	the Mizar counterpart
$[x]_R$	`Class(R,x)`
the upper approximation of A	`UAp A`
the lower approximation of A	`LAp A`
the boundary region of A	`BndAp A`
the membership function $\mu_X^I(x)$	`MemberFunc(X,I).x`
approximation space	`Approximation_Space`
tolerance approximation space	`Tolerance_Space`
rough upper inclusion	`c=^`
rough lower inclusion	`_c=`
rough inclusion combined	`_c=^`
I-rough set in the universe U	`rough Subset of U`
I-definable set in the universe U	`exact Subset of U`
P-rough set in the universe U	`RoughSet of U`

3.1 Classical Rough Sets

The notion of P-set is based on the original concept of equivalence relation which induces a partition on the field of the underlying relation. In the MML it has no clear representation, even if classes of abstraction are natural mathematical constructions. Classes of P-sets are identified via predicate of rough equality which reads as follows:

```
definition let A be Tolerance_Space, X, Y be Subset of A;
  pred X _=^ Y means
    LAp X = LAp Y & UAp X = UAp Y;
end;
```

In fact, the granularity of definitions is even better – we considered feasible to have distinct predicates for $X =_* Y$ and $X =^* Y$ (where X and Y have resp. their lower and upper approximations equal). Naturally, two sets are equal in the sense of $=_*^*$ iff they are equal in both senses – the lower and the upper equality. Alternatively, we claim that a subset of a tolerance approximation space is rough, if its boundary approximation, i.e. the set-theoretical difference between its upper and lower approximation is not equal to the empty set.

```
definition let A be Tolerance_Space, X be Subset of A;
  attr X is rough means
    BndAp X <> {};
end;
```

Otherwise, we claim that the subset is exact, that is it is a set in the ordinary sense (crisp). It is done via construction of antonyms for adjectives, which allow the user to divide all subsets formally into two disjoint classes.

3.2 Pairs of Definable Sets

I-sets can be considered a natural RST-counterpart of the interval sets – with respect to the same Pawlak approximation space both are uniquely determined by each other. In the Mizar formalism it can be described just as below:

```
definition let A be Tolerance_Space, X be Subset of A;
  mode RoughSet of X means  :: ROUGHS_1:def 8
    it = [LAp X, UAp X];
end;
```

Note however that this does not reduce to the ordinary set even if both approximations are equal. Because the I-model provides the better mathematical description, it was chosen by us to define a lattice of rough sets. Even if set-theoretical operators on the set of all I-sets do not rather have a well-defined semantics, this interpretation provides a mathematical model which is both elegant and can be a subject to further generalizations.

4 Interval Sets

Let us recall the notion of an interval set [12]:

$$[A_1, A_2] = \{A \in 2^U : A_1 \subseteq A \subseteq A_2\} \tag{1}$$

where U is a finite set called the universe. Usually, the assumption of $A_1 \subseteq A_2$ is granted, but we define an interval set also in case when A_1 is not a subset of A_2. The set of all interval sets over a universe U, with operations $\sqcap$ and $\sqcup$ defined componentwise, forms a lattice. Moreover, it is a distributive and bounded lattice, where $[\emptyset, \emptyset]$ is its bottom and $[U, U]$ – its top. Firstly, an interval set is defined as a family of subsets with two parameters being its boundaries (we dropped an assumption of $X \subseteq Y$ since it can be proven otherwise the resulting interval is empty).[2]

```
definition let U be set, X, Y be Subset of U;
  func Inter (X,Y) -> Subset-Family of U equals
    { A where A is Subset of U : X c= A & A c= Y };
end;
```

An interval set of the form $[A, A]$ is equivalent to the set in an ordinary sense (but of course direct replacement is just erroneous). Furthermore, we gave the notion needed to characterize the carrier of the interval set algebra; its elements are all intervals.

```
definition let U;
  mode IntervalSet of U -> Subset-Family of U means
    ex A, B be Subset of U st it = Inter (A, B);
end;
```

[2] Due to the lack of space we usually drop proofs; they can be tracked in the full source.

The next part of the article is introducing operations on the interval sets:

```
definition let U be non empty set, A, B be non empty IntervalSet of U;
  func A _/\_ B -> IntervalSet of U equals
    INTERSECTION (A, B);
end;
```

(and similarly `A _\/_ B`, `A _\_ B`), where `INTERSECTION` is an ordinary Boolean operation taken componentwise. Equivalently, it was natural to give characterization of the aforementioned objects in terms of the operations on their bounds. Let us cite only the case of the difference of intervals.

```
theorem
  A _\_ B = Inter (A‘‘1 \ B‘‘2, A‘‘2 \ B‘‘1);
```

where `A‘‘1`, `A‘‘2` denote the boundaries of interval set `A`.

Although the complementation operator is neither Boolean nor a pseudocomplement, it is definitely worth introducing (the symbols "`[#]U`" and "`{}U`"are introduced to add to the set U its proper type, i.e. a subset of itself).

```
definition let U be non empty set, A be non empty IntervalSet of U;
  func A ^ -> non empty IntervalSet of U equals :Def8:
    Inter ([#]U,[#]U) _\_ A;
end;
```

Obviously, an ordinary inclusion cannot be used as the ordering relation in the lattice of interval sets. Since the types of all objects are extended to the most general type `set`, the usual notation of set-theoretical inclusion ("`c=`") could not be used.

```
definition let U be non empty set, A, B be non empty IntervalSet of U;
  pred A _c=_ B means
    A‘‘1 c= B‘‘1 & A‘‘2 c= B‘‘2;
end;
```

It is hardly the same relation, which can be illustrated by the fact that the identity $A \setminus B = \emptyset$ is true for arbitrary sets A, B such that $A \subseteq B$, but it is not the case of interval sets. Hence the following statement:

```
theorem
  ex A,B being non empty IntervalSet of U st
    A _c=_ B & A _\_ B <> Inter ({}U,{}U);
```

Of course, in the proof of this fact, the appropriate concrete example of two sets should have been constructed. After we have defined necessary binary operations on interval sets, the structure of the lattice of such sets (called `InterLatt` in our formalization) can be described – but let us omit the full citation here as we will provide it in the next section.

5 Lattices of Rough Sets

As many of the objects occuring in the MML base on the notion of a structure, the type "Lattice" is also the structure type (with the carrier and two binary operations). Underlying properties (commutativity, associativity, and absorption laws) are added to this radix type via six adjectives (attributes). The general lattice theory in the MML is formalized mainly according to Grätzer's *General Lattice Theory* and this development contains many standard results from this classical book. Also recent automatically obtained equational characterizations were translated into Mizar (as the Robbins problem about the alternative axiomatization of Boolean algebras, short single axioms for Boolean algebras based on the Sheffer stroke, ortholattice bases and so on).

Below we quote the definition of the lattice of rough sets. Its carrier consists of the set of all rough sets over an arbitrary but fixed tolerance approximation space X and the lattice operations are defined here elementwise.

```
definition let X be Tolerance_Space;
  func RSLattice X -> strict LattStr means
    the carrier of it = RoughSets X &
    for A, B being Element of RoughSets X,
        A', B' being RoughSet of X st A = A' & B = B' holds
     (the L_join of it).(A,B) = A' _\/_ B' &
     (the L_meet of it).(A,B) = A' _/\_ B';
end;
```

A similar definition of the interval set algebra `InterLatt` differs only in the case of carrier – we decided to have binary operations, although on various universes, encoded under the same symbols. Furthermore, it can be defined over an arbitrary non-empty set. We have proved formally that the lattice of rough sets is distributive and complete, it has also the lower and the upper bound. We decided for the binary operation approach because in this way the ordering is defined automatically[3] which makes this approach somewhat stronger.

Also, after we have defined

```
definition let X;
  func RoughIso X -> Homomorphism of RSLattice X,
                       InterLatt the carrier of X means
    for x being Element of RSLattice X holds
      it.x = [x‘1, x‘2];
  correctness;
end;
```

as the homomorphism (under `correctness` conditions we had to prove that `RoughIso` preserves suprema and infima, the existence and the uniqueness of such a mapping) between both structures (note that we forget in some sense about abstract structure connected with the lattices of rough sets, i.e. about the indiscernibility relation – we are interested only on the subsets of the carrier of X), usual properties can be proved only for the one of these objects.

[3] Recall that a poset to be a lattice should meet some additional requirements.

6 Learning Rough Sets

Rough sets definitely proved its feasibility and usefulness in various fields of the engineering (also biology or medicine, here especially systems as RSES and Rosetta are potentially useful), however they can be successfully used not only in the industry, but also in purely academic environment. For two consecutive years now (2005/2006 and 2006/2007), a group of students in University of Białystok (Poland) trained their ability in the field of reasoning on rough sets based on the Mizar proof-assistant. The main advantages of the application of this machine proof-checker seem to be the following:

Automatic verification – the results are checked without teacher's help, so very objective ("prove until computer will report no errors, I won't complain"); once the student knows the language, he/she can write syntactically correct texts (if not – computer parser points out the errors); once he/she learned the semantics – he/she tries to formulate the facts and prove them by him/herself;

Self-study enabled – the exercises are the best way for a student to getting knowledge ("experience, not only doctrine");

Human-friendly approach – if the justification is right, the computer also suggests the possible improvements of a proof, so students can benefit in various ways;

Logical correctness – while learning rough sets also classical predicate logic is taught as a side-effect; the logical correctness is crucial, so the student continuously has to remember about the rules; learning proving tactics (direct and indirect proof) and basic rules (exemplification, generalization);

Similarity to the natural language – even if readable for the machine, the language is not that artificial; so it is relatively not much time to get the syntax right;

Multi-purpose systems – rethinking and showing counterexamples is possible – e.g., develop concrete example of an approximation space one of the inclusions is not valid;

Real-life applications – verification of data – we can apply rough set methods to concrete systems; this is probably most difficult – the analysis of even small portions of data could be highly time-consuming;

Distance learning approach – the geographical diversity of learners is no problem any longer.

Note that for obvious reasons in a proof checker illustrative features of figures (e.g. proof suggestions based on diagrams) are not available.

7 Conclusions and Further Work

A kind of computer certification of mathematical proofs seems to become an important issue in the contemporary science. We believe that the use of the Mizar system can be attractive for mathematicians and the development of RST in it is really feasible, although of rather challenging character due to the broad nature

of the discipline. We tried to show on selected examples of the development of the lattice of rough/interval sets that the formalization of this theory can be continued not paying the high price for raising duplicate notions from scratch, but via reusing the existing formal apparatus yet available in the MML.

As the future work, we may point out the formalization of [3] we started some time ago. We are interested in further development of the lattice-theoretical approach to the notion of RST (as in [8]) which are influential even in the broader algebraic sense [2] as well as in extending rough set model as [13]. Thanks to the structure of interval algebras we could obtain some direct correspondence with the described lattice of rough sets, so the results can easily be exchanged and proven only in one of both cases. We plan also to provide another axiomatic base for the MML, more RST-specific [1]. There is no doubt that logical foundations are harder to change (Mizar is based on the classical logic). On the other hand, we hope that rough sets techniques can be applied to improve the MML Query searching engine.

References

1. Bryniarski, E.: Formal conception of rough sets. Fundamenta Informaticae 27(2–3), 109–136 (1996)
2. Düntsch, I., Winter, M.: Construction of Boolean contact algebras. AI Communications 13, 235–246 (2004)
3. Gomolińska, A.: A comparative study of some generalized rough approximations. Fundamenta Informaticae 51(1–2), 103–119 (2002)
4. Grabowski, A.: On the computer-assisted reasoning about rough sets. In: Dunin-Kęplicz, B., et al. (ed.) Monitoring, Security, and Rescue Techniques in Multiagent Systems, Advances in Soft Computing, pp. 215–226. Springer, Heidelberg (2005)
5. Grabowski, A., Schwarzweller, Ch.: Rough Concept Analysis – theory development in the Mizar system. In: Asperti, A., Bancerek, G., Trybulec, A. (eds.) MKM 2004. LNCS, vol. 3119, pp. 130–144. Springer, Heidelberg (2004)
6. Iwiński, T.B.: Algebraic approach to rough sets. Bull. Pol. Acad. Sci. Math. 35, 673–683 (1987)
7. Järvinen, J.: Approximations and rough sets based on tolerances. In: Ziarko, W., Yao, Y. (eds.) RSCTC 2000. LNCS (LNAI), vol. 2005, pp. 182–189. Springer, Heidelberg (2001)
8. Järvinen, J.: Ordered set of rough sets. In: Tsumoto, S., Słowiński, R., Komorowski, J., Grzymała-Busse, J.W. (eds.) RSCTC 2004. LNCS (LNAI), vol. 3066, pp. 49–58. Springer, Heidelberg (2004)
9. Pawlak, Z.: Rough Sets. International Journal of Information and Computer Science 11, 341–356 (1982)
10. Pomykała, J.A.: About tolerance and similarity relations in information systems. In: Alpigini, J.J., Peters, J.F., Skowron, A., Zhong, N. (eds.) RSCTC 2002. LNCS (LNAI), vol. 2475, pp. 175–182. Springer, Heidelberg (2002)
11. Skowron, A., Stepaniuk, J.: Tolerance approximation spaces. Fundamenta Informaticae 27(2–3), 245–253 (1996)
12. Yao, Y.Y.: Two views of the theory of rough sets in finite universes. International Journal of Approximation Reasoning 15(4), 291–317 (1996)
13. Ziarko, W.: Variable precision rough set model. Journal of Computer and System Sciences 46(1), 39–59 (1993)