# ROUGH AND MODAL ALGEBRAS

**Laurent VIGNERON**
**CRIN & INRIA Lorraine**
**BP239, 54506 Vandoeuvre Cedex, France**
e-mail: vigneron@loria.fr

**Anita WASILEWSKA**
**Department of Computer Science,**
**SUNY at Stony Brook,**
**Stony Brook, NY 11794, USA**
e-mail: anita@cs.sunysb.edu

## ABSTRACT

We have used here a theorem prover `DATAC` (*Déduction Automatique dans des Théories Associatives et Commutatives*) to discover, study and compare properties of rough and corresponding modal algebras. The preliminary results were reported in [19]. The prover was developed at CRIN & INRIA Lorraine, Nancy (France), by the first author. The rough algebras were introduced by the second author in [17] as a generalization of rough algebra of formulas of a modal logic S5, defined and examined in [1]. The rough algebras provide a mathematical characterization of a notion of *rough equality* introduced by Pawlak in 1985 ([7]). The modal algebras were introduced by McKinsey and Tarski in [4] under the name of closure algebras. They were first (algebraic) models for modal logics S4 and S5, as opposed to Kripke models invented some 20 years later ([3]). We also present here some detailed proofs chosen out of thousands proofs of theorems discovered and proved automatically by the theorem prover.

## 1. INTRODUCTION

The theory of rough sets provides a methodological framework to study classification problems with imprecise or incomplete information. The theory was originally introduced by Pawlak in [7]. It provides a complementary model to fuzzy ([9]) or evidence ([14]) models for dealing with imprecise, noisy, or incomplete information. It is also a departure point for studies in machine learning, knowledge discovery and lately in database mining. It has also been applied to various domains such as medical diagnosis, information retrieval, control algorithms acquisition, and market analysis.

We present here two new connections for the rough set theory. We link it with the automated theorem proving and algebraic logic. More precisely, we first use the long established ([6]) relationship between the rough sets and modal S5 logic, and hence with some topological Boolean algebras and secondly, we use their newly established ([1], [17]) relationship with *rough algebras*. Then we use a recently developed, by the first author, theorem prover `DATAC` to discover new theorems in two modal and two rough classes of algebras and to examine the relationship between these classes.

`DATAC` is a new kind of a prover. It proposes either to prove theorems by refutation, as usually provers do, or by straightforward deduction from axioms. The possibility of generating, in a very sophisticated way, a straightforward deduction from axioms is a distinct and the most interesting feature of the prover. In the straightforward deduction, the prover acts as a deductive system, i.e. its end product is a set of theorems with their formal proofs. In our research we have used mainly the second technique, but we also use the first technique for comparison of theorems for modal and rough algebras, respectively. The rough algebras under consideration are quasi-Boolean version of some closure algebras introduced in [4]. We call them here modal S4 and S5 algebras. We single out two classes of rough algebras and call them because their similarity with modal S4 and S5 algebras, the rough R4 and R5 algebras.

The structure of rough algebras is very complicated and their properties are much less intuitive than the standard modal algebras. The role of the prover in finding and proving new theorems or properties of these algebras is hence essential in developing their consistent theory.

To make our paper self contained we first review in Section 2 some basic definitions concerning the rough sets, modal and rough algebras and discuss their relationship. In the following Section 3 we present the prover `DATAC`. In Section 4 we give some statistics concerning the complexity of our theorem building and discuss methods of finding theorems which are characteristic for each class of our algebras. In the last Section 5 we present some proofs generated by the prover. These are proofs of previously unknown theorems and moreover, finding the mathematical proofs of them (we know only their strictly formal, syntactic proofs discovered by the prover) seems to us not easy at all. We leave it as an open question to the readers.

## 2. ROUGH SETS AND ALGEBRAS

Rough sets provide one of the first non-statistical methodologies for data analysis, knowledge discovery and data mining ([20]). One of the major advantages of this approach is the existence of a well-founded theoretical model ([8]) and long term advanced foundational studies.

To make our paper self contained we first review in this section some basic definitions.

**Approximation space [8]:** Let $U$ be a non-empty set called a universe, and let $R$ be an equivalence relation on $U$. The triple $(U, \emptyset, R)$ is called an *approximation space.*

**Lower, upper approximations:** Let $(U, \emptyset, R)$ and $A \subset U$. Denote by $[u]$ the equivalence class of $R$. The sets

$$IA = \bigcup\{[u] \in A/R : [u] \subset A\},$$
$$CA = \bigcup\{[u] \in A/R : [u] \cap A \neq \emptyset\}$$

are called *lower* and *upper approximations* of $A$, respectively. We use here a topological notation for the lower and upper approximation because of their topological interpretation and future considerations.

**Rough equality [5]:** Given an approximation space $(U, \emptyset, R)$ and any $A, B \subset U$. We say that the sets $A$ and $B$ are *roughly equal* and denote it by $A \sim_R B$ if and only if $IA = IB$ and $CA = CB$.

**Boolean algebra:** An abstract algebra $(A, 1, \cap, \cup, \neg)$ with unit element 1 is said to be a *Boolean algebra* if it is a distributive lattice and every element $a \in A$ has a complement $\neg a \in A$.

**Quasi-Boolean algebra [10]:** An abstract algebra $(A, 1, \cup, \cap, \sim)$ is called a *quasi-Boolean algebra* if $(A, 1, \cup, \cap)$ is a distributive lattice with unit element 1 and for any $a, b \in A$, $\sim(a \cup b) = \sim a \cap \sim b$ and $\sim\sim a = a$.

**Topological Boolean algebra [10]:** By a topological Boolean algebra we mean an abstract algebra $(A, 1, \cap, \cup, \neg, I)$ where $(A, 1, \cap, \cup, \neg)$ is a Boolean algebra and, moreover, the following conditions hold: $I(a \cap b) = Ia \cap Ib$, $Ia \cap a = Ia$, $IIa = Ia$, and $I1 = 1$, for any $a, b \in A$.

The element $Ia$ is called an *interior of* $a$. The element $\neg I \neg a$ is called a *closure of* $a$ and will be denoted by $Ca$. Thus the operations $I$ and $C$ are such that $Ca = \neg I \neg a$ and $Ia = \neg C \neg a$. The element $a$ is said to be *open* (resp. *closed*) if $a = Ia$ (resp. $a = Ca$).

**Modal S4 algebra:** Any topological Boolean algebra is called here a modal S4 algebra, or *S4 algebra*.

**Modal S5 algebra:** A modal S4 algebra is called *S5 algebra*, if additionally all open elements are closed and all closed elements are open, i.e. when we add the axiom $CIa = Ia$ or $ICa = Ca$.

We consider here two classes of rough algebras: R5 and R4 algebras. They correspond to the modal S4 and S5 algebras. They are particular cases of topological rough algebras defined in [17]. They are not purely mathematical invention. The rough algebra of formulas of modal logic S5 as developed and examined in [1] is an example the R5 algebra. The R4 algebra is a quasi-Boolean correspondent of the topological Boolean algebra, i.e. of the S4 algebra. The R5 algebra is a quasi-Boolean version of the topological Boolean algebra such that each open element is closed and each closed element is open, i.e. of the S5 algebra. The formal definition of R4 and R5 algebras is the following.

**R4 and R5 algebras [19]:** An abstract algebra $(A, 1, \cup, \cap, \sim, I, C)$ is called *a R4 algebra* if it is *a distributive lattice with unit element* 1 and additionally for all $a, b \in A$ the following equations are satisfied: $\sim\sim a = a$, $\sim(a \cup b) = \sim a \cap \sim b$, $I(a \cap b) = Ia \cap Ib$, $Ia \cap a = Ia$, $IIa = a$, $I1 = 1$, $Ca = \sim I \sim a$.

The algebra obtained from the R4 algebra by adding the axiom $CIa = Ia$ is called a *R5 algebra*.

A natural set theoretical interpretation of the properties of the topological Boolean algebras is established by Stone representation theorem ([15]). For example, $a \cap Ia = Ia$ means that any set $A$ contains its interior IA. The representation theorem provides an intuitive motivation for new properties and is an useful source of counter-examples.

The case of R4 and R5 algebras is more complicated and much less intuitive. While the operations $\cup$ and $\cap$ are represented as set theoretical union and intersection, the operation $\sim$ cannot be represented as a set theoretical complementation. The set theoretical interpretation of the rough complement depends on the mapping $g : A \longrightarrow A$ such that for all $a \in A$, $g(g(a)) = a$, called *involution*. The representation theorem for R4 or R5 algebras ([17]) states that their properties have to hold in R4, R5 algebras (fields) of sets. For example, the set theoretical meaning of a R4 algebra property $a \cap \sim 1 = \sim 1$ is the following. Given any non empty set $X$, given any involution $g$ on $X$, for any $A \subset X$, $A \cap (X - g(X)) = (X - g(X))$. This property is intuitively obvious, because any involution has to map the set $X$ onto itself.

The set theoretical interpretation of the definition of the closure operation in R4 is the following: $CA = X - g(I(X - g(A)))$, for any involution $g$. One can see that it becomes less intuitive than the "normal" Boolean topological definition of closure as complement of the interior of the complement of the set. The situation becomes even more complex when we think about possible (or impossible) properties. For example, one of the simplest properties of R4 algebras proven by the prover ([19]) is $a \cap Ib \subseteq C(a \cap Ib) \cap b$. Its set theoretical R4 interpretation is that for any $A, B \subset X$ and for any involution $g$ on $X$, $A \cap IB \subseteq (X - g(I(X - g(A \cap IB)))) \cap B$.

The above examples show that it is much more difficult to build an intuitive understanding of properties of R4 and R5 algebras, as well as to discover their mathematical proofs than it is in classical case of S4 and S5 algebras. It is not only difficult to prove new properties, it is also difficult to think how they should look like. We strongly believe that the study presented in this paper would not be possible without the use of the prover.

## 3. THE PROVER

The theorem prover `DATAC`[1], for *Déduction Automatique dans des Théories Associatives et Commutatives*, has been developed at CRIN & INRIA Lorraine, Nancy (France). This software is written in CAML Light (18000 lines), a language of the ML family.

`DATAC` can be used for theorem proving or for straightforward deduction. It manipulates formulas of first order logic with equality expressed in a clausal form $A_1 \wedge \ldots \wedge A_n \Rightarrow B_1 \vee \ldots \vee B_m$. The deduction techniques implemented are detailed in [13]. We present here only a short overview of them.

### Deduction Techniques

The prover is based on the first order logic with equality, hence the clauses use the equality predicate. But, we

[1] For any information about `DATAC`, please contact the first author, <*vigneron@loria.fr*>.

do not need to state all the equality axioms. The symmetry, transitivity and functional reflexivity of the equality are simulated by a deduction rule called *Paramodulation* [11]. Its principle is to apply replacements in clauses. A paramodulation step in a positive literal is defined by

$$\frac{L_1 \Rightarrow l = r \vee R_1 \qquad L_2 \Rightarrow A[l'] \vee R_2}{(\ L_1 \wedge L_2 \Rightarrow A[r] \vee R_1 \vee R_2\ )\sigma}$$

where the substitution $\sigma$ unifies the term $l$ and the subterm $l'$ of $A$, i.e. $l\sigma$ is equal to $l'\sigma$. This last subterm is replaced by the right-hand side $r$ of the equality $l = r$, and the substitution is applied on the whole deduced clause. A similar paramodulation rule is defined for replacements in negative literals, i.e. literals on the left-hand side of the $\Rightarrow$ sign.

The reflexivity property of the equality predicate is simulated by a rule called *Reflexion*, defined by:

$$\frac{l = r \wedge L \Rightarrow R}{(\ L \Rightarrow R\ )\sigma}$$

where the substitution $\sigma$ unifies the terms $l$ and $r$.

The *Resolution* rule [12] permits to deal with the other predicate symbols than the equality:

$$\frac{A_1 \wedge L_1 \Rightarrow R_1 \qquad L_2 \Rightarrow A_2 \vee R_2}{(\ L_1 \wedge L_2 \Rightarrow R_1 \vee R_2\ )\sigma}$$

where $\sigma$ unifies $A_1$ and $A_2$.

### Strategies

In order to limit the number of possible deductions and to avoid useless deductions, the prover uses several strategies of deduction.

The first one is an *ordering strategy* [2]. It uses an ordering for comparing the terms and for orienting the equations. Hence, when a paramodulation step is applied from an equation $l = r$, it is checked that a term is never replaced by a bigger one, i.e. that the term $r$ is not greater than the term $l$ for this ordering. This technique is similar to the use of a rewriting rule $l \rightarrow r$.
The ordering is also used for selection of literals. For instance, it is possible to impose the condition that each deduction step has to use the maximal literal of a clause.

Another essential strategy is the *simplification*. We have defined some simplification rules whose purpose is to replace a clause by a simpler one, using term rewriting.
We have also defined some *deletion rules*. For instance, clauses which contain a positive equation $l = l$, or a same atom on the left-hand side and on the right-hand side of the implication sign $\Rightarrow$, are deleted. Another deletion technique is the *subsumption*, which can be schematized by: if a clause $L \Rightarrow R$ belongs to a set of clauses $S$, then any clause of the form $L\sigma \wedge L' \Rightarrow R\sigma \vee R'$ can be removed from $S$.

### Deduction modulo $E$

The most important feature of `DATAC` is the deduction modulo a set of equations $E$. The motivation for such deduction is the following.

The *commutativity property* of an operator $f$, $f(a,b) = f(b,a)$, cannot be oriented as a rewrite rule. This is a major problem when applying paramodulation steps.

Also, the *associativity property* of an operator $f$, $f(f(a,b),c) = f(a, f(b,c))$, has the disadvantage to provoke infinite sequences of paramodulation steps. For example, from an equation $f(d_1, d_2) = d_3$, we can derive

$$\begin{array}{rcl} f(d_1, f(d_2, e_1)) & = & f(d_3, e_1) \\ f(d_1, f(f(d_2, e_1), e_2)) & = & f(f(d_3, e_1), e_2) \\ & \vdots & \end{array}$$

Moreover, when these two properties (commutativity and associativity) are combined, it becomes very difficult to deduce useful clauses. For example, there are 1680 ways to write the term $f(a_1, f(a_2, f(a_3, f(a_4, a_5))))$, where $f$ is associative and commutative. These 1680 terms are all semantically equivalent but none of them can be omitted, for the completeness of the deductions.

So, `DATAC` is defined for being run modulo a set of equations $E$, composed of commutativity, and associativity and commutativity properties. These equations do not appear in the initial set of clauses. They are simulated by specific algorithms for equality checking, pattern matching and unification, in conjunction with some specially adapted paramodulation rules.

### Advantages of `DATAC`

`DATAC` is an entirely automatic tool which, given a set of clauses, deduces new properties, consequences of these clauses. `DATAC` is refutationally complete: if a set of clauses is incoherent, it will find a contradiction. The only reason that may it fail in its research of a contradiction is the limit of the memory size of the computer.

The implemented techniques involve ordering and simplification strategies combined with a deduction system based on paramodulation and resolution, as mentioned earlier, but other important strategies are also available, as the superposition and basic strategies [16].

At the end of an execution, the user can ask for a lot of extra information. Especially the prover can present a proof of a derived property, or of the contradiction found. Many statistics are also available, such as the number of deduction steps, the number of simplification steps, and the number of deletions.

## 4. STATISTICS

In this section, we first describe a technique for comparing two algebras. Next, we present the results obtained by this technique for comparing R4 and R5, R4 and S4, R5 and S5, and S4 and S5.

### Comparison of Algebras

Let $A$ and $B$ be two algebras represented by a set of clauses in the first-order logic. For comparing those algebras we apply the following technique, using the theorem prover `DATAC`:

1. First, for each algebra, we deduce a large number of new properties. Let $A_1$ and $B_1$ be the sets of properties obtained for $A$ and $B$ respectively.

2. Given $A_1$, we try to load each clause of $B_1$. Some of the clauses $B_1$ are deleted because subsumed by clauses of $A_1$. Some other clauses may subsume

clauses of $A_1$. Let $A_2$ and $B_2$ be the sets of clauses kept from $A_1$ and $B_1$ resp.

3. We apply the same operation with $B_1$ as initial set, loading each clause of $A_1$. Let $A_3$ and $B_3$ be the sets of clauses kept from $A_1$ and $B_1$ resp.

4. The first important result is the *intersection between $A_1$ and $B_1$*. The set of properties representing $A_1 \cap B_1$ is defined by: $A_2 \backslash A_3$, equivalent to $B_3 \backslash B_2$. This way for obtaining the intersection may look tricky. The only other way to find it is by hand. But we would have to be very courageous: $A_1$ and $B_1$ may contain thousands of clauses; renaming of variables and permutation of subterms because of associative and commutative operators may let two equivalent clauses look very different.

5. Given $A_2$ and $B_2$, we delete as many clauses of $B_2$ as possible by simplification steps. Let $B_4$ be the sets of persistent clauses of $B_2$.
Given $A_3$ and $B_3$, we delete as many clauses of $A_3$ as possible by simplification steps. Let $A_4$ be the sets of persistent clauses of $A_3$.

6. We have all the information needed for sorting the properties $A_1$ of $A$: the clauses in $A_2 \backslash A_3$ are common with $B_1$; the clauses in $A_3 \backslash A_4$ are *tautologies* for $B_1$; the remaining clauses, $A_4$, are *candidates for being pure A properties*.

7. We can also sort the properties $B_1$ of $B$: the clauses in $B_3 \backslash B_2$ are common with $A_1$; the clauses in $B_2 \backslash B_4$ are *tautologies* for $A_1$. Finally, the clauses in $B_4$ are *candidates for being pure B properties*.

8. We can study more precisely the candidates for being pure $A$ (resp. $B$). `DATAC` does not only generate clauses, but it also permits to find easily their detailed proof. Therefore we can check the proof of each clause of $A_4$ (resp. $B_4$). The properties whose proof uses only clauses common to $A_1$ and $B_1$ are not pure $A$ (resp. $B$) properties. The other ones are said to be *strong candidates for being pure A (resp. B) properties*.
Note that if $A \subseteq B$, either $A_4$ is empty, or we can check that each clause of $A_4$ is proved using clauses common to $A_1$ and $B_1$.

9. A solution for proving a property $P$ is a pure $A$ (resp. $B$) property is to prove all the properties of $A$ (resp. $B$) from $B \cup \{P\}$ (resp. $A \cup \{P\}$).

## Study of R4, R5, S4 and S5

We have studied the rough R4 and R5 algebras, and the modal S4 and S5 algebras.

The R4 algebra $(A, 1, +, ., \tilde{}, \mathtt{I}, \mathtt{C})$ is defined with 16 clauses (see Section 5). In addition, `+` and `.` are associative and commutative. We have run `DATAC` with these 16 clauses and have stopped it arbitrarily after 5000 deduced clauses. Only 407 clauses have been kept as useful; these clauses represent what we have denoted $R4_1$ in the description of our technique for comparing algebras.

R5 is defined with the 16 clauses of R4 and with the property `CIa = Ia`. We have run `DATAC` with these 17 clauses and have stopped it arbitrarily after 5000 deduced clauses. The final set of useful clauses contains 294 clauses.

S4 is defined with the 16 clauses of R4 and with the properties `a+~a = 1` and `a.~a = ~1`. We have run `DATAC` with these 18 clauses and have stopped it arbitrarily after 10000 deduced clauses. The final set of useful clauses contains 863 clauses.

S5 is defined with the 18 clauses of S4 and with the property `CIa = Ia`. We have run `DATAC` with these 19 clauses and have stopped it arbitrarily after 10000 deduced clauses. The final set of useful clauses contains 676 clauses.

We describe in the following table all the results obtained by our technique when comparing R4 and R5, R4 and S4, R5 and S5, and S4 and S5. The column *trivial A* shows the number of clauses of $A_1$ subsumed by $B_1$ or simplified into a trivial property by $B_1$. The column *future B* shows the number of clauses of $A_4$ that are proved using only clauses of $A \cap B$. The last column presents the number of strong candidates for being pure $A$ properties. There is no value in this column if $A \subseteq B$.

| *A vs B* | $A \cap B$ | *trivial A* | *future B* | *"pure" A* |
|---|---|---|---|---|
| *R4 vs R5* | 222 | 173 | 12 | – |
| *R5 vs R4* | 222 | 2 | 22 | 48 |
| *R4 vs S4* | 137 | 0 | 270 | – |
| *S4 vs R4* | 137 | 25 | 0 | 701 |
| *R5 vs S5* | 137 | 0 | 157 | – |
| *S5 vs R5* | 137 | 22 | 0 | 517 |
| *S4 vs S5* | 470 | 383 | 10 | – |
| *S5 vs S4* | 470 | 10 | 151 | 45 |

Let us give some examples of the deduced clauses, in which `a` $\sqsubseteq$ `b` if and only if `a+b = b` and `a.b = a`.

### Comparison of R4 and R5

Some of the 222 properties common to R4 and R5 are:

```
    C1  =  1            Ia  ⊑  a
   C~1  =  ~1            a  ⊑  Ca
   I~1  =  ~1           Ia  ⊑  Ca
C(a+b)  =  Ca+Cb      Ia.b  ⊑  a
   CCa  =  Ca          a.b  ⊑  Ca
   ~Ca  =  I~a         ~Ia  =  C~a
```

```
     Ia.b.c  ⊑  C(Ia.b).a
       a.Ib  ⊑  C(a.Ib).b
     I(a+b)  ⊑  (I(Ca+b).a)+(I(Ca+b).b)
   Ia.Ib.Ic  ⊑  a.b.C(a.b.c)
       Ia.b  ⊑  I(a+c).C(b.Ia)
       a.Ib  ⊑  a.C(a.C(a.Ib)).b
       b.Ia  ⊑  C(C(b.Ia).a)
Ca.I(Ia+b).c ⊑  Ia+(Ca.b)
```

Some of the 48 strong candidates for being pure R5 properties are:

```
     ICa  =  Ca             a  ⊑  I(Ca+b)
I(Ca+Cb)  =  Ca+Cb   C(Ca.Cb)  =  Ca.Cb
I(Ca+Ib)  =  Ca+Ib   C(Ca.Ib)  =  Ca.Ib
I(Ia+Ib)  =  Ia+Ib   C(Ia.Ib)  =  Ia.Ib
```

```
      I(Ia+Cb+Cc) = Ia+Cb+Cc
        C(Ia.b).Ia = C(Ia.b)
        C(Ca.b).Ib = Ca.Ib
I(C(a.b)+b).I(C(a.b)+a) = C(a.b)
  (Ca.b)+C(b.I(Ca+c)) = C(b.I(Ca+c))
```

From the pure R5 properties, we have proved the following patterns, where $n+m>0$:

$$\mathtt{I(Ca_1+\cdots+Ca_n+Ib_1+\cdots+Ib_m)} = \mathtt{Ca_1+\cdots+Ca_n+Ib_1+\cdots+Ib_m}$$

$$\mathtt{C(Ca_1.\cdots.Ca_n.Ib_1.\cdots.Ib_m)} = \mathtt{Ca_1.\cdots.Ca_n.Ib_1.\cdots.Ib_m}$$

**Comparison of R4 and S4**

Comparing R4 and S4, we have found 701 strong candidates for being pure S4 properties. Let us give some of them:

```
  a.I~a = ~1          ~a.Ia = ~1
  a+C~a = 1           ~a+Ca = 1
  Ca+I~a = 1          Ia+C~a = 1
  Ca.I~a = ~1         Ia.C~a = ~1

          I(~a+~b).IC(a.b) = ~1
          C(~a.~b)+CI(a+b) = 1
                C~a+I(a+b) = 1
         (a.C(a.b))+(a.C~b) = a
(a.~b)+(a.c.b.~d)+(a.~c)+(a.d) = a
```

**Comparison of R5 and S5**

The result of the comparison of R5 and S5 is 517 strong candidates for being pure S5 properties. Let us mention some of them:

```
~a.I(Ia+~a) = I~a      ~a.I(I~a+a) = I~a
a+C(a.C~a) = Ca        C(Ca.~a)+a = Ca
C(a.C~a)+Ia = Ca       I~a+I(Ca+b) = 1
~a+~b+C(a.b) = 1       a.b.I(~a+~b) = ~1

        I(~a+b).C(a.~b) = ~1
           (~a.b)+Ca+C~b = 1
          (a.Cb)+(a.I~b) = a
     Ia+Cb+c+C(C~a.I~b.~c) = 1
 (a.C~b)+(a.I~c)+(a.b.Cc) = a
```

**Comparison of S4 and S5**

The comparison of the 863 properties of S4 with the 676 properties of S5 has led to only 45 strong candidates for being pure S5 properties. Here are some of them:

```
Ca.I(I~a+b) = Ca.Ib     C~a.I(Ia+b) = C~a.Ib
~a.I(Ia+~a) = I~a       ~a.I(I~a+a) = I~a
C(a.C~a)+Ia = Ca        C(Ia.b)+Ia = Ia

          C(Ca.Ib.Ic) = Ca.Ib.Ic
          I~a+I(Ca+b) = 1
       (a.Cb)+(a.I~b) = a
 C(~a.~b).I(I(a+b)+c) = C(~a.~b).Ic
 C(a.b).I(I(~a+~b)+c) = C(a.b).Ic
(a.Cb)+(I~b.C(a.Cb).a) = C(a.Cb).a
(a.C~b)+(a.I~c)+(a.b.Cc) = a
```

## 5. PROOFS

In this section, we detail the proof of some properties. But first let us mention the initial set of clauses representing R5. It consists of the next 16 clauses representing the R4 algebra $(A, 1, +, ., \tilde{}, \mathtt{I}, \mathtt{C})$,

```
 1: (a.b)+b = b         2: a.(b+c) = (a.b)+(a.c)
 3: a+1 = 1             4: a.1 = a
 5: ~~a = a             6: ~(a+b) = ~a.~b
 7: a+a = a             8: a.a = a
 9: a.~1 = ~1          10: ~1+a = a
11: ~(a.b) = ~a+~b     12: I(a.b) = Ia.Ib
13: IIa = Ia           14: I1 = 1
15: Ia.a = Ia          16: ~I~a = Ca
```

to which we add the following clause:

```
17: CIa = Ia
```

**Example 1: a pure R5 Property**

Let us detail the demonstration of the property

`(Ca.b)+C(b.I(Ca+c)) = C(b.I(Ca+c))`

which is R5 but not R4.

Lemma 1: `~Ia = C~a`
*Proof:* `~I~~a` is equal to `~Ia` by 5, and also to `C~a` by 16.

Lemma 2: `~Ca = I~a`
*Proof:* `~~I~a` is equal to `~Ca` by 16, and also to `I~a` by 5.

Lemma 3: `Ia+a = a`
*Proof:* `(Ia.a)+a` is equal to `Ia+a` by 15, and to `a` by 1.

Lemma 4: `I((a.b)+(a.c)) = Ia.I(b+c)`
*Proof:* `I(a.(b+c))` is equal to `I((a.b)+(a.c))` by 2, and also to `Ia.I(b+c)` by 12

Lemma 5: `~a.C~a = ~a`
*Proof:* `~(a+Ia)` is equal to `~a.~Ia` by 6 and `~a.C~a` by Lemma 1, and also to `~a` by Lemma 3.

Lemma 6: `IC~a = C~a`
*Proof:* `~CIa` is equal to `I~Ia` by Lemma 2 and `IC~a` by Lemma 1, and also to `~Ia` by 17 and `C~a` by Lemma 1.

Lemma 7: `a.Ca = a`
*Proof:* `~~a.C~~a` is equal to `a.Ca` by 5, and also to `~~a` by Lemma 5 and `a` by 5.

Lemma 8: `Ia.I(a+b) = Ia`
*Proof:* `I((a.a)+(a.b))` is equal to `Ia.I(a+b)` by Lemma 4, and also to `I(a+(a.b))` by 8 and `Ia` by 1.

Lemma 9: `ICa = Ca`
*Proof:* `IC~~a` is equal to `ICa` by 5, and also to `C~~a` by Lemma 6 and `Ca` by 5.

Lemma 10: `Ia.I(Ia+b) = Ia`
*Proof:* `IIa.I(Ia+b)` is equal to `Ia.I(Ia+b)` by 13, and also to `IIa` by Lemma 8 and `Ia` by 13.

Lemma 11: `(a.b)+Ca = Ca`
*Proof:* `(a.b.Ca)+Ca` is equal to `(a.b)+Ca` by Lemma 7, and also to `Ca` by 1.

Lemma 12: `Ca.I(Ca+b) = Ca`
*Proof:* `ICa.I(ICa+b)` is equal to `Ca.I(Ca+b)` by Lemma 9, and also to `ICa` by Lemma 10 and `Ca` by Lemma 9.

Theorem 1: `(Ca.b)+C(b.I(Ca+c)) = C(b.I(Ca+c))`
*Proof:* `(Ca.b.I(Ca+c))+C(b.I(Ca+c))` is equal to `(Ca.b)+C(b.I(Ca+c))` by Lemma 12, and also to `C(b.I(Ca+c))` by Lemma 11.

**Example 2: a pure S5 Property**
Let us detail the demonstration of the property

`(a.C~b)+(a.I~c)+(a.b.Cc) = a`

which is S5 but neither S4, nor R5. Note S5 is defined by the 17 clauses of R5 plus the clause:

`18: a+~a = 1`

Lemma 13: `a+C~a = 1`
*Proof:* `Ia+a+~Ia` is equal to `a+~Ia` by Lemma 3 and `a+C~a` by Lemma 1, and also to `1+a` by 18 and `1` by 3.

Lemma 14: `~a+Ca = 1`
*Proof:* `~a+C~~a` is equal to `~a+Ca` by 5, and also to `1` by Lemma 13.

Lemma 15: `C(~a+~b) = C~a+C~b`
*Proof:* `~I(a.b)` is equal to `C~(a.b)` by Lemma 1 and `C(~a+~b)` by 11, and also to `~(Ia.Ib)` by 12 and `~Ia+~Ib` by 11 and `C~a+C~b` by Lemma 1.

Lemma 16: `C(a+~b) = Ca+C~b`
*Proof:* `C(~~a+~b)` is equal to `C(a+~b)` by 5, and also to `C~~a+C~b` by Lemma 15 and `Ca+C~b` by 5.

Lemma 17: `(a.Cb)+(a.~b) = a`
*Proof:* `a.(Cb+~b)` is equal to `(a.Cb)+(a.~b)` by 2, and also to `a.1` by Lemma 14 and `a` by 4.

Lemma 18: `C(a+b) = Ca+Cb`
*Proof:* `C(a+~~b)` is equal to `C(a+b)` by 5, and also to `Ca+C~~b` by Lemma 16 and `Ca+Cb` bu 5.

Lemma 19: `(a.Cb)+(a.Cc)+(a.~b.~c) = a`
*Proof:* `(a.C(b+c))+(a.~(b+c))` is equal to `(a.C(b+c)) +(a.~b.~c)` by 6 and `(a.(Cb+Cc))+(a.~b.~c)` by Lemma 18 and `(a.Cb)+(a.Cc)+(a.~b.~c)` by 2, and also to `a` by Lemma 17.

Lemma 20: `(a.C~b)+(a.Cc)+(a.b.~c) = a`
*Proof:* `(a.C~b)+(a.Cc)+(a.~~b.~c)` is equal to `(a.C~b) +(a.Cc)+(a.b.~c)` by 5, and also to `a` by Lemma 19.

Lemma 21: `(a.C~b)+(a.Ic)+(a.b.C~c) = a`
*Proof:* `(a.C~b)+(a.CIc)+(a.b.~Ic)` is equal to `a` by Lemma 20, and also to `(a.C~b)+(a.Ic)+(a.b.~Ic)` by 17 and `(a.C~b)+(a.Ic)+(a.b.C~c)` by Lemma 1.

Theorem 2: `(a.C~b)+(a.I~c)+(a.b.Cc) = a`
*Proof:* `(a.C~b)+(a.I~c)+(a.b.C~~c)` is equal to `(a.C~b) +(a.I~c)+(a.b.Cc)` by 5, and also to `a` by Lemma 21.

## REFERENCES


[1] M. Banerjee and M.K. Chakraborty. Rough Algebra. *Bull. Polish Acad. Sci. (Math.)*, 41(4):299–297, 1993.

[2] J. Hsiang and M. Rusinowitch. Proving Refutational Completeness of Theorem Proving Strategies: The Transfinite Semantic Tree Method. *J. of the ACM*, 38(3):559–587, 1991.

[3] S. Kripke. Semantics Analysis of Intuitionistic Logic. In J.N. Crossley and M.N.E. Dummet, ed., *Proc. of the Eight Logic Colloqium, Oxford 1963*, pp. 92–130. North Holland Publishing C., 1965.

[4] J.C.C. McKinsey and A. Tarski. The Algebra of Topology. *Annales of Mathematics*, 45:141–191, 1944.

[5] M. Novotny and Z. Pawlak. On Rough Equalities. *Bull. Polish Acad. Sci. (Math.)*, 33(1-2):99–104, 1985.

[6] E. Orlowska. Expressive Power of Knowledge Representation Systems. *Int. J. of Man-Machine Studies*, vol. 20, pp. 485–500.

[7] Z. Pawlak. Rough Sets. *Int. J. Comp. Inf. Sci.*, 11:341–356, 1982.

[8] Z. Pawlak. Rough Sets. *Theory and Decision Library.* Kluwer Academic Publishers, 1991.

[9] Z. Pawlak. Rough Sets - A New Approach to Vagueness. In L.A. Zadeh and J. Kacprzyk, ed., *Fuzzy Logic for Management of Uncertainty*, pp. 105–118. Wiley and Sons, New York, 1992.

[10] H. Rasiowa. An Algebraic Approach to Non-Classical Logics. In *Studies in Logic and the Foundations of Mathematics*, vol. 78. North-Holland Publishing Company, Amsterdam, London - PWN, Warsaw, 1974.

[11] G. A. Robinson and L. T. Wos. Paramodulation and First-order Theorem Proving. In B. Meltzer and D. Mitchie, ed., *Machine Intelligence*, vol. 4, pp. 135–150. Edinburgh University Press, 1969.

[12] J. A. Robinson. A Machine-oriented Logic Based on the Resolution Principle. *J. of the ACM*, 12:23–41, 1965.

[13] M. Rusinowitch and L. Vigneron. Automated Deduction with Associative-Commutative Operators. *Applicable Algebra in Engineering, Communication and Computing*, 6(1):23–56, 1995.

[14] A. Skowron, J. Grzymala-Busse. From Rough Set Theory to Evidence Theory. In *Advances in Dempster-Shafer Theory of Evidence.* Wiley and Sons, New York, 1993.

[15] M. H. Stone. Boolean algebras and their relation to topology. *Proc. of National Academy of Sciences*, vol. 20, pp. 197–202, 1934.

[16] L. Vigneron. Associative-Commutative Deduction with Constraints. In A. Bundy, ed., *Proc. 12th Int. Conf. on Automated Deduction, Nancy (France)*, vol. 814 of *LNAI*, pp. 530–544. Springer-Verlag, 1994.

[17] A. Wasilewska. Topological Rough Algebras. In Y. Lin, ed., *Rough Sets and Database Mining.* Kluwer. To appear.

[18] A. Wasilewska and M. Banerjee. Rough Sets and Topological Quasi-Boolean Algebras. In *ACM CSC'95 23rd Annual Workshop on Rough Sets and Database Mining*, Conf. Proc., San Jose State University, San Jose, California, pp. 61–67, 1995.

[19] A. Wasilewska and L. Vigneron. Rough Equality Algebras. In *Annual Joint Conf. on Information Sciences*, Conf. Proc., Wrightsville Beach, North Carolina, pp. 26–30, 1995.

[20] W. P. Ziarko. Rough Sets, Fuzzy Sets and Knowledge Discovery. In *Workshops in Computing Series.* Springer-Verlag, 1994.