A Symbolic-Numeric Approach to MPL Continuous Logic and to Rule Based Expert Systems whose Underlying Logic is MPL

E. Roanes-Lozano*,[1], Luis M. Laita[2] and E. Roanes-Macías[1]

[1]*Algebra Department, Universidad Complutense de Madrid, Spain*

[2]*Artificial Intelligence Department, Universidad Politécnica de Madrid, Spain*

Abstract: We first briefly describe an algebraic model of classical and modal many-valued logics due to the authors and introduced in previous works. A similar approach, also using Computer Algebra techniques (Gröbner bases) and oriented to perform effective calculus in a continuous logic: Minimal Polynomial Logic (MPL) is presented. The implementation has been developed in the Computer Algebra System *Maple*. The possibility to perform knowledge extraction and to check consistency in Rule Based Expert Systems (RBES) whose underlying logic is MPL, has also been explored. The article is illustrated with examples of very simple RBES.

INTRODUCTION

Motivation

Since the early nineties, we have been developing an algebraic approach to knowledge extraction and consistency checking in Rule Based Expert Systems (RBES) using residue class rings and Gröbner bases (GB). The underlying logic can be either classical Boolean logic or many-valued modal logic.

The effective computations are carried out using Computer Algebra Systems and the implementations are surprisingly brief.

Different applied works have been successfully carried out, mainly in Medicine.

This paper studies the possibility to extend this treatment to the continuous logic that is, possibly, most similar: Minimal Polynomial Logic (MPL).

Some Introductory Notes About RBES

Expert Systems (ES) are a particular type of Knowledge Based Systems. An ES can be considered as a system of automated deduction from the information provided by experts. ES basically consist of three components:

- a Knowledge Base (KB), the construction of which requires to choose an strategy leading to a model of knowledge gathering, the aim of which is to orderly collect and symbolically represent the available information;
- an Inference Engine (IE) which both verifies the consistency of the KB and extracts consequences automatically;
- and, possibly, an interactive Graphic Users Interface (GUI), for people not necessarily familiar with the logical and mathematical details of the system construction.

The ES treated here are RBES. In this case, the strategy leading to its logic formulation consists of translating all the information formulated in natural language into a set of propositional logic formulae, called "production rules", of the form $\delta \rightarrow \phi$, where δ is a literal[1] or a conjunction of literals, and ϕ is a literal or a disjunction of literals. If the underlying logic is classical Boolean logic, an example of production rule is:

$$x_1 \wedge x_2 \wedge \neg x_3 \wedge \neg x_5 \rightarrow x_9 \vee x_{13}$$

that is read as: "if x_1 and x_2 and not x_3 and not x_5 hold, then x_9 or x_{13} holds".

The set of all literals in the KB contains a distinguished subset called "the set of potential facts". The "potential facts", are the literals that do appear in the left side of one production rule (at least) and such that neither them nor their contraries appear in the right side of any production rule[2], together with their contraries. For instance, the whole set of potential facts can be something like:

$$\{x_1, \neg x_1, x_2, \neg x_2, x_5, \neg x_5, x_8, \neg x_8, x_{16}, \neg x_{16}\}$$

That a given set of facts (i.e., a subset of the set of potential facts that is stated as true) is "consistent" means that, from each pair formed by a potential fact and its contrary, like $(x_5, \neg x_5)$, only one literal, for instance $\neg x_5$, is chosen. That it is "maximal" means that such a choosing must be done in all pairs.

Some KB also contain other items known as "integrity constraints" (IC). An IC is a logical formula that translates some experts' opinions about the convenience of stating that two or more potential facts, for instance x_8 and x_{16}, ought not to occur simultaneously.

A (consistent) set of potential facts, together with the set of production rules and IC, constitute the KB.

*Address correspondence to this author at the Algebra Department, Facultad de Educacion, Universidad Complutense de Madrid, c/Rector Royo Villanova s/n, 28040-Madrid, Spain; E-mail: eroanes@mat.ucm.es

[1]A "literal" is a propositional variable of the underlying logic or its negation.

[2]In case the underlying logic is modal many-valued, modal connectives also have to be taken into account here.

1874-1142/08

Intuitively, "forward firing" is the way in which, what is stated as "true", propagates through the RBES. It corresponds to the logic concept of tautological consequence[3].

The IE is a an automated tool (in our case, a program in the Computer Algebra System -CAS), that verifies consistency (that is, checks that the system does not lead to contradictions) and draws consequences from the information contained in the KB (that corresponds to the logical concept of "tautological consequence").

A RBES is inconsistent when a consistent set of facts, such that any formulae formed with the variables contained in the KB is a tautological consequence of the information contained in the RBES, exists. In particular, contradiction would be a tautological consequence of such information.

Example 1: Let us consider the trivial RBES formed by the three rules:

$R1: x \wedge y \rightarrow v$

$R2: v \rightarrow z$

$R2: w \rightarrow \neg z$

The potential facts are $\{x,\neg x,y,\neg y,w,\neg w\}$. If the given set of facts is $\{x,w\}$, only $\neg z$ is obtained by forward firing (no inconsistency is found starting with this set of facts). But if $\{x,y,\neg w\}$ are given as facts (it is a consistent set of facts), both z and $\neg z$ are obtained by forward firing, so the RBES is inconsistent.

In case the underlying logic is many-valued, two different types of consistency can be distinguished [1].

An Algebraic Model of RBES Whose Underlying Logic is Classical Boolean or Modal Many-Valued

The next theorem relates tautological consequence in logic with an ideal membership problem in polynomial ring theory. The theorem is applied to verify consistencies in, and draw consequences from, the KB.

The logical and mathematical ideas behind the GB-based IE can be found in the mathematical model of logic and RBES in [2,3], that are based on previous works for rewriting classical Boolean [4,5] and many-valued logics [6].

For instance, in classical Boolean logic, the polynomial translations of the logic connectives are

$$x_1 \vee x_2 = x_1 + x_2 - x_1 x_2$$

$$x_1 \wedge x_2 = x_1 x_2$$

$$\neg x_1 = 1 - x_1$$

(these are the same polynomials George Boole used in his seminal works on logic).

Similarly, the polynomial expressions corresponding to the basic logical formulae in Lukasiewicz's three-valued logic (if *2* is assigned to "true", *1* to "undetermined" and *0* to "false", and computations are performed in $\mathbb{Z}_3$) are:

$$\neg x_1 = 2x_1 + 2$$

$$\Box x_1 = x_1^2 + 2x_1$$

$$\Diamond x_1 = 2x_1^2$$

$$x_1 \vee x_2 = x_1^2 x_2^2 + x_1^2 x_2 + x_1 x_2^2 + 2x_1 x_2 + x_1 + x_2$$

$$x_1 \wedge x_2 = 2x_1^2 x_2^2 + 2x_1^2 x_2 + 2x_1 x_2^2 + x_1 x_2$$

(where $\Box$ means "necessarily" and $\Diamond$ means "possibly").

Theorem 1

A logical formula, α, is a tautological consequence of the set of formulae $M=\{A_1,A_2,\ldots, A_m\}$ in a certain p-valued modal logic (where p is a prime number) if and only if the polynomial translation[4] of $\neg\alpha$ in the ring $\mathbb{Z}_p[x_1, x_2, \ldots, x_n]/I$ (where[5] $I=\langle x_1^p - x_1, x_2^p - x_2, \ldots, x_n^p - x_n\rangle$), denoted $Pol(\neg\alpha)$, belongs to the (polynomial) ideal $\langle Pol(\neg A_1), Pol(\neg A_2), \ldots, Pol(\neg A_m)\rangle$.

The effective methods for calculating normal forms (NF), i.e., reduction modulo ideals, in polynomial rings and characterizing ideals *via* some very special bases (GB) are due to Bruno Buchberger [7]. Implementations of GB and NF are nowadays included in most CAS.

We usually denote by K the polynomial ideal generated by the polynomial translations of the negation of the elements of a consistent subset of the set of potential facts. Similarly, J is the ideal generated by the polynomial translations of the negation of the production rules and IC of the RBES. We shall use this notation hereinafter.

So we have an effective method what can decide whether a logical formula, α, can be obtained by forward firing from a consistent set of facts and the production rules and IC of the RBES or not:

a logical formula, α, can be obtained by forward firing from a consistent set of facts and the rules and IC in the RBES $\Leftrightarrow$

$$\Leftrightarrow NF(Pol(\neg\alpha), K+J)=0$$

(in the ring $\mathbb{Z}_p[x_1, x_2, \ldots, x_n]/I$).

We can alternatively check with the CAS whether $NF(Pol(\neg\alpha), I + K + J)$ is *0* or not (in the ring $\mathbb{Z}_p[x_1, x_2, \ldots, x_n]$).

We ask whether $Pol(\neg\alpha)$ belongs to $I+K+J$ or not in the ring $\mathbb{Z}_p[x_1, x_2, \ldots, x_n]$, instead of asking whether $Pol(\neg\alpha)$ belongs to $K+J$ or not in the ring $\mathbb{Z}_p[x_1, x_2, \ldots, x_n]/I$, because most CAS cannot perform calculations in residue class rings.

And we also have an effective method that can decide whether a RBES is inconsistent or not. With the notation above,

a consistent set of facts leads to inconsistency $\Leftrightarrow$

$$\Leftrightarrow GB(J+K)=\langle 1\rangle$$

(in the ring $\mathbb{Z}_p[x_1, x_2, \ldots, x_n]/I$)[6], that can be decided in the CAS by checking whether $GB(I + K + J)$ is *[1]* or not (in the ring $\mathbb{Z}_p[x_1, x_2, \ldots, x_n]$), because, if this basis is *[1]*, from Theorem 1, any formula is a tautological consequence of the information provided by the KB (which means inconsistency).

[3]A logical formula, α, is a "tautological consequence" of a set S of formulae, denoted $S \models \alpha$, if and only if, whenever all formulae in S are true, α is also true.

[4]The polynomial translation of the logic connectives will depend on p and on the logic chosen. For instance, Lukasiewicz's three-valued logic and Kleene's three-valued logic lead to different polynomial expressions.

[5]"<" and ">" are used to represent the ideal generated by the elements between those symbols.

[6]An ideal is the whole ring if and only if its GB is *[1]*.

A BRIEF OVERVIEW OF MPL

MPL [8,9] is a generalization of classical propositional logic allowing truth values in the closed interval *[0,1]*. The polynomials

$$x_1 \vee x_2 = 1-(1-x_1)(1-x_2)$$

$$x_1 \wedge x_2 = x_1 x_2$$

$$\neg x_1 = 1-x_1$$

fit the truth-table entries of the ordinary logic connectives, and are the ones used by the "lowest degree polynomial logic", denoted PL_0 (note that these polynomials are the same as those used in the previous section).

Now MPL is obtained from PL_0 the following way (Definition 3 of [8]):

"Given a propositional formula *e*, its MPL version e_m is obtained from the PL_0 version e_p by distributing + over × throughout and then substituting subexpressions of the form x_i^k (with $k>1$) with x_i. This substitution will be sometimes be denoted $(\bullet)_m$."

Truth values in MPL can be interpreted either as probabilities of assertions being true, as in Nilsson's probabilistic logic or, also, as the degrees of truth of those assertions, as in fuzzy logic.

As in Boolean logic, $e \vDash e'$ is trivially equivalent to $e \rightarrow e'$. It can be proved (Corollary 7 of [8]) that $e \vDash e'$ if and only if $e_m \equiv (e_m e'_m)_m$. It can also be proved that $e \vDash e'$ if and only if $e_m \leq e'_m$ for any truth values in *[0,1]* of the propositional variables (Theorem 11 of [8]).

The theoretical details about MPL can be found in [8,9] (we shall only deal here with an straightforward algebraic approach and its implementation in a CAS).

APPROACHING MPL FROM ALGEBRA AND COMPUTER ALGEBRA

Let us observe that the substitution $(\bullet)_m$ introduced in *Definition 3* of [8] (quoted above) can be interpreted as the NF of the PL_0 version of the propositional formula (a polynomial of[7] $\mathbb{R}[x_1,...,x_n]$) modulo the ideal $\langle x_1^2-x_1,..., x_n^2-x_n\rangle$.

Therefore, from an algebraic point of view [10], it is straightforward and trivial to consider the MPL version of the propositional formulae as polynomials lying in the residue class ring

$$\mathbb{R}[x_1,...,x_n]/I \; ; \; I=\langle x_1^2-x_1,...,x_n^2-x_n\rangle$$

Nevertheless, there is a quantitative difference w.r.t. the model of classical Boolean and many-valued logic that we had previously developed. Let us detail it.

THE ALGEBRAIC APPROACH TO MPL COMPARED TO THE ALGEBRAIC APPROACH TO CLASSICAL BOOLEAN LOGIC

In the Boolean case (and also in the *p*-valued case, substituting all the *2* by *p* in the definitions of the ring and of ideal *I*), the residue class ring

$$\mathbb{Z}_2[x_1,...,x_n]/I \; ; \; I=\langle x_1^2-x_1,...,x_n^2-x_n\rangle$$

is a polynomial model where ideals of the logic Boolean algebra do correspond one-to-one with the ideals of the polynomial Boolean algebra (that are the same as the ideals of the polynomial Boolean ring). Therefore, it is not important which expression of the polynomial translation of a formula is chosen (i.e., which generator of the polynomial ideal[8] generated by a polynomial translation of the formula is chosen).

Summarizing, in the Boolean case we have the beautiful diagram in Fig. (**1**).

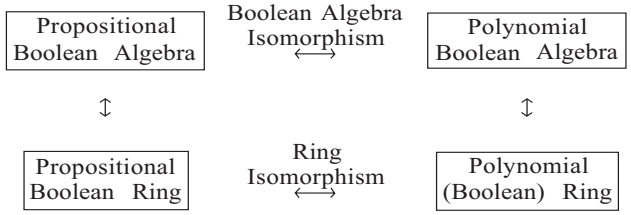

Fig. (1). Isomorphisms relating the propositional Boolean algebra and the polynomial residue class ring.

Example 2

The polynomial translation of the formula $x_1 \wedge x_1$ (of classical Boolean logic) in the residue class ring $\mathbb{Z}_2[x_1,...,x_n]/I$ is $x_1^2=x_1$. And there is no inconvenience in considering, e.g., x_1^7 as a polynomial translation of $x_1 \wedge x_1$ in this residue class ring: for any truth value in *{0,1}*, it is the same to consider x_1 or x_1^7.

Nevertheless, when we try to do something similar for MPL, $\mathbb{R}[x_1,...,x_n]/I \; ; \; I=\langle x_1^2-x_1,...,x_n^2-x_n\rangle$ is not a polynomial model of MPL in the same sense, with a one-to-one correspondence between ideals, but a residue class ring where only a certain polynomial translation of each logic formula (the canonical representative of the residue class) behaves as the logic formula that it translates!

Example 3

Let us consider the same formula of Example 2. The polynomial translation of the formula of MPL $x_1 \wedge x_1$ in $\mathbb{R}[x_1,...,x_n]/I$ would be x_1. Despite the fact that in that residue class ring, e.g. $x_1^7=x_1^2=x_1$, x_1 and only x_1 behaves as $x_1 \wedge x_1$ for all values in the closed interval *[0,1]*.

EFFECTIVE COMPUTATIONS IN MPL USING COMPUTER ALGEBRA TECHNIQUES

The NF of a polynomial in $\mathbb{R}[x_1,...,x_n]$ modulo the ideal $\langle x_1^2-x_1,..., x_n^2-x_n\rangle$, can be effectively computed in any CAS including the possibility to perform GB computations, like *Maple*, *Mathematica*, *CoCoA*…

Let us observe that the polynomials of MPL are reduced modulo ideal *I,* and therefore an upper bound of its degree in each variable is *1* (and, consequently, an upper bound of its total degree is the number of propositional variables considered).

MAPLE IMPLEMENTATION

We can easily implement this idea in a CAS, for instance, in *Maple*.

[7]They could also be considered as polynomials of $\mathbb{Z}[x_1,..., x_n]$, although we shall consider their real roots and shall substitute *x* by real values.

[8]All ideals in these rings are principal.

We have to begin loading the GB package and defining the sequence of propositional variables, for instance $x_1,...,x_{10}$:

```
> with(Groebner):
> SV:=x[i] $ i=1..10:
```

Then we can define ideal *I* (renamed *iI*, as *I* is a reserved word in *Maple*), using an auxiliary function, denoted *fu* (in fact, a GB of the ideal, w.r.t. pure lexicographic order, is directly assigned to *iI*):

```
> fu:=var->var^2-var:
> iI:=Basis(map(fu,[SV]),plex(SV)):
```

Now we are ready to define the basic logic connectives:

```
> `NEG`:=(m::algebraic)-> 1-`m`:
> `&AND`:=(m::algebraic,
           n::algebraic)
 -> NormalForm(expand(`m`*`n`),
    iI, plex(SV)):
```

and the "derived" ones:

```
> `&OR`:=(m::algebraic,
          n::algebraic)
 -> NormalForm(NEG(NEG(`m`)
    &AND NEG(`n`)),
    iI, plex(SV)):
> `&IMP`:=(m::algebraic,
           n::algebraic)
 -> NormalForm(NEG(`m`) &OR `n`,
    iI, plex(SV)):
> `&XOR`:=(m::algebraic,
           n::algebraic)
 -> NormalForm((NEG(`m`) &AND
    `n`) &OR (`m` &AND NEG(`n`)),
    iI, plex(SV)):
> `&IFF`:=(m::algebraic,
           n::algebraic)
 -> NormalForm((`m` &IMP `n`)
    &AND (`n` &IMP `m`),
    iI, plex(SV)):
```

(the binary ones will be input to *Maple* in infix mode) and to define tautology and contradiction as constants:

```
> t:=1:
> c:=0:
```

Let us observe that the reductions modulo *iI* performed by command `NormalForm` above are computed considering the order for variables of sequence *SV* and the order for monomials "pure lexicographic".

The restriction of the MPL connectives to the Boolean truth values *{0,1}* return the same truth-values as the Boolean connectives, as shown in Fig. (**2**).

x[1]	*x[2]*	*x[1]_and_x[2]*	*x[1]_or_x[2]*	*x[1]_xor_x[2]*
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

Fig. (2). MPL binary connectives restricted to *{0,1}* (computed with *Maple*'s implementation of MPL).

SYMBOLIC EXAMPLES

The CAS can perform computations with non-assigned variables, so we can obtain and compare polynomial expressions of formulae (and consequently compare the logical formulae themselves).

Example 4

For instance, we can check that $\vee$ is associative (observe that this is not formally a proof: we only check it for propositional variables *x[1]*, *x[2]* and *x[3]*; nevertheless it will also hold if *x[1]*, *x[2]* and *x[3]* are substituted by any polynomial expressions, so it will be true in general)[9]:

```
> x[1] &AND (x[2] &AND x[3]);
```

$$x_1x_2x_3$$

```
> (x[1] &AND x[2]) &AND x[3];
```

$$x_1x_2x_3$$

Example 5

Check idempotency of $\vee$:

```
> x[1] &OR x[1];
```

$$x_1$$

Example 6

Check one of the de Morgan laws:

```
> NEG(x[1] &AND x[2]);
```

$$1-x_1x_2$$

```
> NEG(x[1]) &OR NEG(x[2]);
```

$$1-x_1x_2$$

Example 7

Unlike what happens in many-valued modal logics, $x_1 \wedge \neg x_1$ is a contradiction:

```
> x[1] &AND NEG(x[1]);
```

$$0$$

The previous examples were really simple and could be easily performed by hand. For complex or long examples, using a computer is highly preferable, and this simple *Maple* implementation (or a similar one in another CAS) could therefore be a very convenient tool.

Let us underline that the formulae that can be handled in this approach are completely general (for instance, implications are not restricted to Horn clauses, like in some logic languages and programs).

[9]In Maple, statement separator "`;`" produces echo, meanwhile "`:`" produces no echo.

NUMERICAL EXAMPLES

The CAS can also perform numeric computations. Therefore we can calculate the truth value of a logical formula or the truth value of a propositional variable in a formula for the formula to have a certain truth value.

Example 8

Let P be the logical formula

$$x[1] \wedge x[2] \rightarrow x[3].$$

If the truth values of *x[1]*, *x[2]* and *x[3]* are *0.3*, *0.75* and *0.8* (respectively), we can easily compute the truth value of P substituting all the ocurrences of these propositional variables by their numerical values (what command `subs` does in *Maple*):

```
> P:=(x[1] &AND x[2]) &IMP x[3];
> subs(x[1]=0.3,x[2]=0.75,x[3]=0.8,
      P);
                    0.9550
```

Moreover, given the values of all but one propositional variables, we can compute the truth value of the other propositional variable in order the formula to have a given truth value.

Example 9

Let us consider the formula of Example 8. If the truth values of *x[1]* and *x[3]* are *0.3* and *0.8* (respectively), then we can compute the truth value of *x[2]* for the logical formula P: $x[1] \wedge x[2] \rightarrow x[3]$ to have truth value *0.955* using command `fsolve` (numeric solve) :

```
> fsolve(
    subs(x[1]=0.3,x[3]=0.8,P)=0.9550,
         x[2]);
            0.7500000000
```

so the solution is: *x[2]* should have truth value *0.75* (this procedure is the converse of Example 8).

Nevertheless, it is not always possible to find such a solution, as shown in Example 10.

Example 10

Let us consider, again, the simple formula P of Example 8 for the truth values of *x[1]* and *x[3]* being *0.3* and *0.8* (respectively). Then, if we try to compute the truth value of *x[2]* for the logical formula P to have truth value *0.2*:

```
> fsolve(
    subs(x[1]=0.3,x[3]=0.8,P)=0.2,
         x[2]);
            13.3333333
```

is obtained, a value that is outside interval *[0,1]* and therefore not a possible truth value in MPL.

Anyway that a certain formula cannot reach a certain truth value is not a critical drawback. For instance, in Lukasiewicz's three-valued logic, no value of the propositional variables q and r make the logical formula $\Diamond q \vee \Diamond r$ undetermined.

Finally, we can also ask the CAS to compute when a formula has a truth value fulfilling an inequality.

Example 11

If the truth values of *x[1]* and *x[3]* are *0.3* and *0.8* (respectively), for which truth values of *x[2]* does formula P of Example 8 have a truth value strictly greater than 0.95?

```
> solve(
    subs(x[1]=0.3,x[3]=0.8,P)>0.95,
    x[2])
  intersect RealRange(0,1);
     RealRange(0, 1) intersect
     RealRange(-infinity,
               Open(0.8333333333))
```

that is, interval *[0,0.8333333333...)*. Let us underline that we have used the non-numeric solving command `solve` instead of `fsolve` in this case.

A COMPARISON WITH KLEENE OR LUKASIEWICZ THREE-VALUED LOGIC CONNECTIVES

We have computed with *Maple* the truth tables of $\wedge$ and $\vee$ in MPL, restricting the input values to *{0,½,1}* (see Fig. **3**). If *0, ½, 1* represented false, undertermined/undecided and true (respectively), the truth table obtained in Kleene or Lukasiewicz three-valued logics would have been the same, except the central line, that would have been: *½, ½, ½, ½*

x[1]	*x[2]*	*x[1]_and_x[2]*	*x[1]_or_x[2]*
0	0	0	0
0	1/2	0	1/2
0	1	0	1
1/2	0	0	1/2
1/2	1/2	1/4	3/4
1/2	1	1/2	1
1	0	0	1
1	1/2	1/2	1
1	1	1	1

Fig. (3). MPL binary connectives restricted to *{0,1/2,1}* (computed with *Maple*'s implementation of MPL).

The *¼* and *¾* would be the expected answers if we were thinking of probabilities and *x[1]* and *x[2]* were independent events.

DESIGNING RBES WHICH UNDERLYING LOGIC IS MPL

Let us observe that we are going to consider the most general case, where:

- any value in *[0,1]* can be assigned as truth value of a fact, rule and IC,
- the variables that are not potential facts (i.e., that appear in consequents) can have truth values ranging in *[0,1]* too.

Note that in our previous works based on the theory for many-valued modal logics in [2,3], we have always considered that the rules and IC, as well as the given facts are stated as "true" (never as undertermined/undecided), although the forward reasoning is performed according to the many-valued modal logic chosen (modal connectives can appear within the antecedents and the consequents of the rules).

KNOWLEDGE EXTRACTION IN RBES WHICH UNDERLYING LOGIC IS MPL

If the underlying logic of a RBES is Boolean logic, what can be asked is whether a formula can be obtained by forward firing from the facts, rules and IC or not.

If the underlying logic is MPL, the certainty with which a formula can be obtained from the facts, rules and IC can be asked too (as done above for only one implication).

Example 12 (Part I)

Let us consider now as propositional variables *x, y, z, p, q, r, s, u, v, w*. The only difference w.r.t. to the implementation used above is that, in the beginning, its second line of code will be

```
> SV:=x,y,z,p,q,r,s,u,v,w:
```

Let the rules be, for instance:

R1: $x \wedge \neg y \rightarrow z$

R2: $z \rightarrow p \vee u$

R3: $v \rightarrow \neg u$

R4: $p \rightarrow w$

R5: $r \rightarrow \neg w$

that is, in *Maple*:

```
> R1:=(x &AND NEG(y))&IMP z:
> R2:= z              &IMP(p &OR u):
> R3:= v              &IMP NEG(u):
> R4:= p              &IMP w:
> R5:= r              &IMP NEG(w):
```

Then, the potential facts are: *x, y, v, r* (and their negations).

The ideal of rules and IC (there are no IC in this case) is:

```
> iK:=[NEG(R1),NEG(R2),NEG(R3),
       NEG(R4),NEG(R5)];
    iK := [x-x*z-x*y+x*y*z, z-z*u-
      z*p+z*p*u, u*v, p-p*w, r*w]
```

A consistent set of facts is $\{x, \neg y, v\}$, and the corresponding ideal is:

```
> iJ:=[NEG(x),NEG(NEG(y)),NEG(v)];
        iJ = [1-x, y, 1-v]
```

The GB of *I+J+K* with respect to the variable order in sequence *SV* and pure lexicographical ordering for monomials is:

```
> base:=Basis([op(iI),op(iJ),
              op(iK)], plex(SV));
    [-1+w, v^2-v, u*v, u^2-u,
    s^2-s, r^2-r, q^2-q, p-1,
      -1+z, -1+y, -1+x]
```

Let us exit the example for a moment.

We can now perform knowledge extraction two ways.

One is using GB in a "booleanized" environment: truth values strictly greater than *0* are upgraded to *1* and *0s* are left as they are. This is a only sufficient condition[10] for a formula to follow by forward firing (without determining truth values).

Example 12 (Part II)

In this particular RBES, if $x, \neg y, v$ are the given facts, and facts, rules and IC with a truth value strictly greater than *0* are associated the truth value *1*, we have: from *R1* and *R2*, $p \vee u$ is obtained, but from *R3* we have $\neg u$, so we have *p*, and then, by *R4, w* is obtained.

This can be checked in the same RBES using classical Boolean logic:

```
> NormalForm(NEG(p),base,plex(SV));
                    0
> NormalForm(NEG(w),base,plex(SV));
                    0
```

The interpretation of these outputs (in continuous logic) would be that the formulae *p* and *w* can be reached by forward firing (with a non null truth value, not determined).

The "booleanization" process could be improved in case each propositional variable always appeared in the antecedents of the rules either preceded by $\neg$ or not, but not in some rules one way and in other rules the other way (unfortunately this is not always the case). Those preceded by $\neg$ and with truth values in *(0,1)* could be assigned the truth value *0* and those not preceded by $\neg$ and with truth values in *(0,1)* could be assigned the truth value *1*.

The second method for knowledge extraction considered is straightforwardly obtained from the algebraic-numeric interpretation of formulae in continuous logic above and can precise the numerical truth values of the different formulae. Unfortunately, it has a drawback: substituting numerical values in the polynomial equations can lead to a non compatible system. This is not a logic inconsistency but a numerical one. Let us illustrate with a trivial example.

Example 13

Let us consider the trivial RBES

R1: $x \rightarrow z$

R2: $y \rightarrow z$

[10]It is not a necessary condition, as shown in the following example. If the truth values of *x* and *y* are *0.9* and *0.1*, respectively, then $x \wedge \neg y$ has truth value $0.9 \cdot (1-0.1) = 0.81$, meanwhile, if booleanized as suggested, $1 \cdot (1-1) = 0$ would be obtained.

In the Boolean case, we have that either none or some of the two rules can be fired. If none can be fired, z is not obtained by forward firing. If exactly one or both can be fired, z is reached by forward firing.

Let us consider now the same rules in the continuous case. In the particular example, for instance, that the truth values of *x, y, R1* and *R2* are *8/10, 2/10, 7/10, 85/100* (respectively), solving the two linear equations for z, we would have:

```
> solve(subs(x=8/10,R1)=7/10,z);
                    5/8
> solve(subs(y=2/10,R2)=85/100,z);
                    1/4
```

that is, $z=5/8$ and $z=1/4$, that, from the algebraic point of view, are incompatible.

One possibility would be to allow only RBES where this did not happen. But this often happens. For instance in a RBES about car care, different rules lead to "stop engine immediately".

Another option, if we wanted to precise the truth values of all formulae, would be to proceed step by step performing numerically the forward firing and actualizing (at each step) the truth values obtained for each variable to its maximum. Unfortunately this is not a polynomial process.

LOGIC INCONSISTENCY OF RBES WHICH UNDERLYING LOGIC IS MPL USING GB

We shall consider a "worse case consistency": we shall say that the RBES is inconsistent if and only if a consistent set of facts can be found, from which, if stated as true, by forward firing the rules and considering the IC (all stated as true), any formula can be obtained.

Example 14

Let us consider the trivial RBES formed by the two rules:

$R1: x \wedge y \rightarrow z$

$R2: w \rightarrow \neg z$

that is, in *Maple*:

```
> R1:=(x &AND y) &IMP z:
> R2:=         w  &IMP NEG(z):
```

There is inconsistency, because *x, y, w* are potential facts, and, if they are all true (i.e., if they have the truth value *1*), as R1 and R2 are also true, we would have:

```
> subs(x=1,y=1,t=1,R1)=1;
                    z=1
> subs(x=1,y=1,t=1,R2)=1;
                   1-z=1
```

that is an incompatible system, as $1=z=0$ is obtained.

We can alternatively check it using ideals, computing the GB of the ideal $I+K+J$ (i.e., the ideal generated by the polynomials in I and the negation of the facts, rules and IC):

```
> iK:=[NEG(x),NEG(y),NEG(t)];
       iK = [1 - x, 1 - y, 1-w]
> iJ:=[NEG(R1),NEG(R2)];
       iJ = [x y - x y z, z w]
> Basis([op(iI),op(iK),op(iJ)],
          plex(SV));
                [1]
```

Example 15

Let us consider again the simple RBES of Example 12, which rules were:

$R1: x \wedge \neg y \rightarrow z$

$R2: z \rightarrow p \vee u$

$R3: v \rightarrow \neg u$

$R4: p \rightarrow w$

$R5: r \rightarrow \neg w$

As above:

```
> iK:=[NEG(R1),NEG(R2),NEG(R3),
       NEG(R4),NEG(R5)]:
```

and for the consistent set of facts $\{x, \neg y, v\}$:

```
> iJ:=[NEG(x),NEG(NEG(y)),NEG(v)]:
```

This consistent set of facts does not lead to inconsistency (the following GB is not *[1]*):

```
> Basis([op(iI),op(iJ),op(iK)],
          plex(SV));
    [w-1, -1+v, u, s^2-s, r,
    q^2-q, -1+p, -1+z, y, -1+x]
```

Meanwhile, the consistent set of facts $\{x, \neg y, v, r\}$ does lead to an inconsistency:

```
> iJ:=[NEG(x),NEG(NEG(y)),NEG(v),
       NEG(r)]:
> Basis([op(iI),op(iJ),op(iK)],
          plex(SV));
                [1]
```

CONCLUSIONS

A surprisingly tiny implementation for performing effective computations in Minimal Polynomial Logic (MPL) that is based on algebraic methods (Gröbner bases) and can be implemented in any Computer Algebra System has been presented. We consider it very convenient.

Although this approach can be applied to perform knowledge extraction and verification of general RBES whose underlying logic is MPL, it only provides partial results and sufficient conditions, unlike the Boolean and many-valued cases.

ACKNOWLEDGMENTS

This work was partially supported by the research projects *MTM2004-03175* (Ministerio de Educación y Ciencia, Spain) and *UCM2005-910563* (Comunidad de Madrid–Universidad Complutense de Madrid, research group *ACEIA*).

We would like to thank the anonymous referees, whose comments and suggestions have greatly improved this paper.

ABBREVIATIONS

ES	=	Expert System(s)
KB	=	Knowledge Base
IE	=	Inference Engine
GUI	=	Graphic Users Interface
RBES	=	Rule Based Expert System(s)
IC	=	Integrity Constraint(s)
CAS	=	Computer Algebra System(s)
NF	=	Normal Form
GB	=	Gröbner Basis/Bases
MPL	=	Minimal Polynomial Logic
PL_0	=	Lowest Degree Polynomial Logic

REFERENCES

[1] Roanes-Lozano E, Roanes-Macías E, Laita LM. Geometric Interpretation of Strong Inconsistency in Knowledge Based Systems. Computer Algebra in Scientific Computing. Proceedings of CASC'99. Springer-Verlag, Berlin 1999; 349-63.

[2] Roanes-Lozano E, Laita LM, Roanes-Macías E. A Polynomial Model for Multivalued Logics with a Touch of Algebraic Geometry and Computer Algebra. Math Comp Simul 1998; 45(1): 83-99.

[3] Laita LM, Roanes-Lozano E, de Ledesma L, Alonso JA. A Computer Algebra Approach to Verification and Deduction in Many-Valued Knowledge Systems. Soft Comp 1999; 3/1: 7-19.

[4] Kapur D, Narendran P. An Equational Approach to Theorem Proving in First-Order Predicate Calculus. General Electric Corporate Research and Development Report 84CRD296, Schenectady, NY 1984.

[5] Hsiang J. Refutational Theorem Proving using Term-Rewriting Systems. Art Intell 1985; 25: 255-300.

[6] Chazarain J, Riscos A, Alonso JA, Briales E. Multivalued Logic and Gröbner Bases with Applications to Modal Logic. J Symb Comp 1991; 11: 181-94.

[7] Buchberger B. An Algorithm for Finding a Basis for the Residue Class Ring of a Zero-Dimensional Polynomial Ideal. Ph.D. Thesis in German. Math. Inst. Univ. of Innsbruck, Insbruck 1965.

[8] Poli R, Ryan M, Sloman A. A new continuous propositional logic. Procs. Of 7th Portuguese Conf. on AI, EPIA 95. Springer-Verlag LNAI 990, Berlin-Heidelberg 1995; 17-28.

[9] Poli R, Kerber M. Minimal Polynomial Logic: Properties and Extensions. The University of Birmingham CSRP-96-2, Birmingham, UK 1996.

[10] Cox D, Little J, O'Shea D. Ideals, Varieties, and Algorithms. Springer-Verlag, New York 1992.

Received: March 27, 2008 Revised: June 27, 2008 Accepted: August 13, 2008