mathematics

Article

A Prototype of a Decision Support System for Equine Cardiovascular Diseases Diagnosis and Management

María Villalba-Orero [1] and Eugenio Roanes-Lozano [2,*]

[1] Hospital Clínico Veterinario Complutense & Departamento de Medicina y Cirugía Animal, Facultad de Veterinaria, Universidad Complutense de Madrid, 28040 Madrid, Spain; mvorero@ucm.es
[2] Instituto de Matemática Interdisciplinar & Departamento de Didáctica de las Ciencias Experimentales, Sociales y Matemáticas, Facultad de Educación, Universidad Complutense de Madrid, 28040 Madrid, Spain
* Correspondence: eroanes@ucm.es; Tel.: +34-91-3946248

Abstract: Proper diagnosis and management of equine cardiac diseases require a broad experience and a specialization in the field, but acquisition of specific knowledge is difficult, due, among other reasons, to the limited literature in this field. Therefore, we have designed, developed, and implemented (on a computer algebra system) a Decision Support System (DSS) for equine cardiovascular diseases diagnosis and management based on clinical practise. At this step it is appropriate for equine science teaching, but this work paves the way for a clinical decision support system that facilitated equine clinicians the management of horses with cardiac diseases, allowing improving health care in this species. The latter would require extensive testing prior to its use. The novelty of this work relies on the organization of the equine cardiology workflow in mathematical logic form, that allowed designing, develop and implement a DSS in this new field. An innovation of this work is the part of the DSS devoted to data completion (motivated by the possible lack of specialization of the users—the veterinarians).

Keywords: equine cardiovascular diseases; veterinarian diagnosis; decision support systems; computer algebra systems

check for updates

Citation: Villalba-Orero, M.; Roanes-Lozano, E. A Decision Support System for Equine Cardiovascular Diseases Diagnosis and Management. *Mathematics* **2021**, *9*, 2580. https://doi.org/10.3390/math9202580

Academic Editor: Junseok Kim

Received: 2 August 2021
Accepted: 9 October 2021
Published: 14 October 2021

1. Introduction

1.1. Equine Cardiovascular Diseases

Heart murmurs and arrhythmias are commonly identified in riding horses [1,2]. Most horses with cardiac disease have a useful performance life, but they can occasionally develop clinical signs that vary from slightly altered clinical signs to poor performance, exercise intolerance, weakness and collapse, among others [1,3] Horses may suffer from physiologic murmurs and benign arrhythmias, and distinguishing those caused by cardiac abnormalities may be difficult [4], unless the veterinarian is a clinician veterinarian specialized in equine cardiology. In addition, guidelines in cardiovascular diseases and literature regarding prognosis and outcome of horses with cardiac disease are scarce [1]. Thus, an important challenge for the general equine clinician is to establish whether a cardiopathy has no clinical relevance or, conversely, it has an impact on the horse's performance and life expectancy or on the horse's and rider's safety [1]. This latter point is very important as horses collapsing during exercise may suffer catastrophic lesions and up to 22.8% of riders were injured during these episodes and while riding a horse that presented sudden death (Figure 1) [5]. Therefore, appropriately clinical management of heart disease in riding horses may be imperative, not only for increasing the horse's life-expectancy, but also for human safety concerns.

Figure 1. Low quality photogram of a real life video of a horse with an atrial fibrillation collapsing during exercise. We are used to seeing scenes of horses falling to the ground in Western films, but they are not real. The great value of this image is that an amateur video captured the precise moment when the poor horse unexpectedly collapses (its head is very low) and the rider (whose legs are clearly visible and is going to be thrown over the horse's head). We include the image here despite its low quality because we consider it a real document illustrating the dangers of a horse collapsing during exercise.

1.2. On the Need for Such a Tool

Diagnosis in the veterinary field is probably even more challenging than in medicine for two reasons: veterinarians usually deal with a variety of species and their patients, although very smart, cannot report their symptoms.

Moreover, veterinarian are usually specialized in the most common domestic animals, but, in many cases, a specialist in specific disorders of a certain species is needed.

Actually, even veterinarians specialized in horses usually lack deep training to properly address cardiovascular diseases in this species. Similarly, in the case of human beings, if a General Practitioner detects a patient with a cardiopathy, he/she refers the patient to the cardiologist. After evaluation, the cardiologist could ask for further tests, such as echocardiography, resting/exercise electrocardiography, etc. The same holds for horses (Figures 2 and 3).

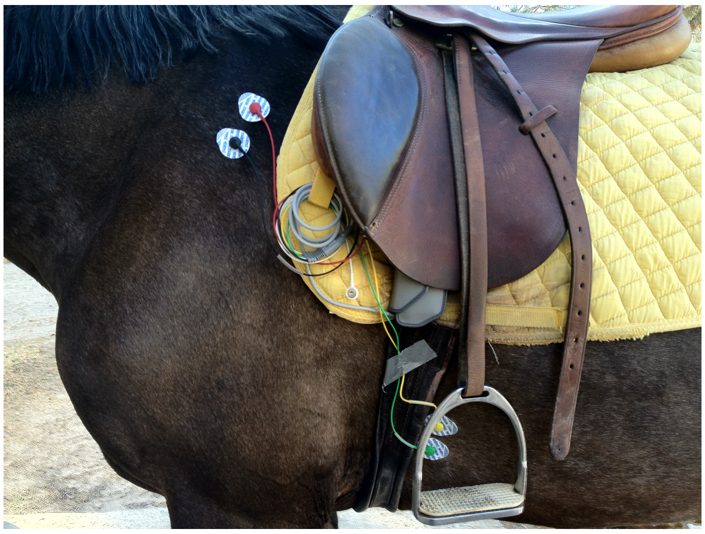
Figure 2. A horse with the electrodes for the exercise electrocardiography already placed.

Figure 3. A horse during an exercise electrocardiography.

Although there are general works in the field of equine diagnosis, such as [6], we know of no one specialized in equine cardiovascular diseases. Reference [6] is an equine general disease diagnosis expert system that was developed using an ontology system that uses the clinical signs as input. It is capable of reliably diagnosing 40 of the most common equine diseases. However, cardiac alterations are not common in this specie and this ontology system merely addresses heart failure (without going into detail).

We consider that such a DSS would be very useful for the many non-specialized veterinarians that have to deal with horses from time to time.

2. Overview of the Decision Support System Designed and Developed

2.1. Some Generalities about Decision Support Systems

DSS are information systems that support decision-making [7,8]. One kind of DSS are knowledge-driven DSS (based on the use of a Knowledge Based System –KBS).

A very common type of KBS are Rule Based Expert Systems (RBES), where the expertise is stored as facts, rules and integrity constraints. RBES do not use conventional programming: they reason through bodies of knowledge, using IF THEN rules that are fired when the variables in their antecedent hold (observe that rule firing concatenates as many rules as necessary). The underlying logic in RBES is not restricted to classic Boolean logic.

2.2. Different Approaches to DSS in Previous Works of the Authors

In the past, we used different approaches for knowledge extraction and verification of RBES in which the underlying logic is either classic Boolean logic or many-valued modal logic, and the similar problem of decision making in a railway interlocking system:

- Gröbner bases-based approach to RBES: a polynomial Boolean ring (a residue class ring), that can be obtained from a polynomial Boolean algebra that is isomorphic to the propositional Boolean algebra translating the logical knowledge in the RBES is used. Knowledge extraction in the RBES is translated in terms of (polynomial) normal forms while the formal verification of the RBES is translated in terms of the degeneracy of the residue class ring into $\{0\}$ (what can be checked using Gröbner bases). This approach is very well suited to address RBES in which the underlying logic is many-valued modal —with a prime number if truth values) [9] and has been used in different diagnostic systems in medicine such as in [10].
- Decision making in railway interlocking systems using Gröbner bases: this topology independent approach evaluates a proposed situation of the trains and the switches of the turnouts and the indication of the colour-light signals, which is translated into the reflexive-transitive closure of a digraph, itself interpreted as a polynomial ideal. That this polynomial ideal is not the whole polynomial ring is proven to be equivalent to the safeness of the proposed situation. It was implemented in the Computer Algebra System (CAS) *Maple* (*Maple* is a trademark of Waterloo Maple Inc.) [11].
- The same problem, interpreting the digraph from a logic point of view was treated in [12] (the implementation is written in *Maple* too).
- Finally, a similar approach to decision making in a railway interlocking using answer set programming (the notion of answer set [13] is an extension of the notion of stable model [14]) was implemented in *Smodels* [15–17] and is detailed in [18].

There are recent innovative approaches in the line of research of algebraization of logic and RBES such as [19,20].

2.3. Generalities about This Equine Cardiology Decision Support System

The two authors decided to collaborate to design, develop and implement a prototype of a DSS about equine cardiovascular diseases. It should address both the diagnosis and the management of the horse.

We wanted the DSS to both easily guide a veterinarian through the different steps of the complex diagnostic procedure (that sometimes requires several tests) and to make a final diagnosis together with recommendations about the horse's management (derived from its health state valuation).

The cardiological knowledge underlying the proposed system was selected and synthesized by the first author. The sources were her experience (consider, for instance [21–23]) and specialized literature such as the already mentioned [1–6]. This knowledge was translated into logic expressions in collaboration with the second author, which is also responsible for the implementation. Observe that no study involving animals has been carried out to develop this prototype.

As the data available are not always exhaustive in medicine and veterinary medicine, we thought about using a three-valued logic. The main candidates were Łukasiewicz's and Kleene's. Summarizing, in both of them a third truth value is considered ("indeterminate"/"undefined") and their main difference is the truth value of "implies" for this third truth value. This difference is derived from considering that either the third truth value is "indeterminate", i.e., it could take any of the values true or false (but which one is presently unknown), or that this third truth value consists of a real third "undefined" possibility. The use of these sort of truth values is especially interesting in medical and veterinarian DSS for beginning extracting knowledge meanwhile the results of a test are not yet known.

Nevertheless, our DSS is aimed at a user that is a veterinarian not specialized in equine cardiology and we wanted to go one step further, not only extracting knowledge in the case of indetermination, but even guiding the user in the tests to be carried out.

Therefore we decided to organize the DSS in two separate parts:

- A "normal" RBES whose underlying logic is classic Boolean logic that extracts knowledge (partial conclusions or diagnoses and the need to new tests to be carried out). Observe that the DSS asks for new tests to be carried out (while necessary, which can happen more than once) and it does not provide a diagnosis unless the situation is clear. Therefore there is no need to adopt other more computationally expensive logics such as many-valued modal logic or fuzzy logic).
- A complementary set of procedures that, in the case of data incompleteness, that is, in case the user has not informed the system whether certain variables in the antecedent of the rules are stated as true or false, asks the unaware user to provide this information. We believe that this tailor-made complement is very useful and we know of no similar one. Unfortunately this approach is laborious.

This new computer tool could be classified as a a RBES which underlying logic is classic Boolean, completed with a data completion system.

We have used for this first prototype the *Logic* package provided by the CAS [24] *Maple 2021* [25–30], resulting in a brief, simple and fast implementation. It uses the connectives provided by its specialized *Logic* package:

`&and`, `&or`, `&not`, `&xor`, `&implies`

as well as he command `Implies` that tests the logical implication specified. Please note that these logical operators designed to build propositional formulae and perform computations in logic. They are different from the usual *Maple* programming language operators `and`, `or`, `not`, `xor`, and `implies` used in conditional statements (in *Maple* the underlying logic is three-valued, as a third `FAIL` truth value is considered, meanwhile the logic used by the *Logic* package is classic Boolean logic).

In this case, with a few dozen rules, the implementation returns the answers to both issues in 3–4 s on a standard portable computer. Nevertheless, we will also consider different approaches in the future.

All input data have to be typed by the end user. Some can be judged by a general veterinarian (such as whether the horse has a "high heart rate" or not) while others (such as interpreting some details of the electrocardiogram) may require the help of an equine cardiologist. Let us underline that the DSS does not try to substitute the equine cardiologist, but to help a general equine veterinarian in assessing the severity of the disease. It could also be used by an equine cardiologist that would like to have a second opinion.

The whole process has taken a long time. A preliminary version of the work was already presented at the ESCO 2020 conference.

From the academic point of view the system has two innovative aspects: the data completion part and its topic (equine cardiology). It has two possible uses: teaching, and real clinical practice (the latter once extensively tested and approved —for obvious reasons).

3. Details of the New Decision Support System Designed and Developed

The DSS system is split into five subsystems:

- Previous Definitions Subsystem
- Subsystem I (first level input data and conclusions/ask for more data)
- Subsystem II (second level input data and conclusions/ask for more data)
- Subsystem III (diagnoses): IIIa: arrhythmias/IIIb: murmurs (third level input data and conclusions/ask for more data)
- Subsystem IV (management).

3.1. Processes

As said above, two different kinds of processes are carried out at each subsystem:

(i) Knowledge extraction: the DSS obtains conclusions from the input data introduced by the end user by forward firing (it can be understood as a standard RBES knowledge extraction). For instance, if from the expert's knowledge we have that $p \wedge q \rightarrow u$ and the user has stated p and q as true (or they have been obtained by forward firing of other rules), the system deduces u, where u is usually

(a) a partial conclusion or partial diagnosis

but can also be

(b) the need to perform a test.

(ii) Data completion: the DSS checks if more data should have been introduced in order to reach the next partial or final diagnosis. It should not be confused with the DSS asking for further tests (after performing the knowledge extraction) mentioned in (i)(b). Here the system checks if the end user has not introduced all the needed data and asks which input data to complete. It can be due to two reasons:

(c) The DSS has performed some knowledge extraction and, with the new information, more data have to been given to the system to correctly continue performing the knowledge extraction.

(d) An unaware end user has not introduced all the data of a certain step of the process.

For example, if from the expert's knowledge we have the rule $p \wedge q \wedge r \wedge s \rightarrow u$ and the user has stated p and q as true (or they have been obtained by forward firing of previous rules) but has said nothing about r and s, the DSS will ask him/her to declare whether r and s are true or false. If both of them are true, the rule will be fired (that is, u will be reached), otherwise it will not be fired (that is, u will not be reached as a conclusion).

This data completion is important as, otherwise, it could happen that the DSS did not reach all the corresponding consequences (diagnoses).

This part can be understood as a facts completion system and it is inspired by and loosely related to the process of adding hypotheses in geometrical theorems introduced in [31].

3.2. General Architecture

The subsystems are arranged in a cascade. The end user is informed of the data he/she should introduce to the previous definitions subsystem. From here onwards each subsystem could return no more information or more information could be extracted by the system. Anyway, all the available data are used by the next subsystem as input.

If the user introduced all the necessary information at every step and the DSS did not ask for more test to be carried out (situation (i)-(b), see Section 3.1), the flow between subsystems would be trivially straightforward (Figure 4, left).

However, as said in Section 3.1, in all subsystems the need to perform further tests (situation (i)-(b)) or a data completion (situation ii)) could be required by the DSS.

For instance, if the situation described in (i)-(b) took place, for example, in Subsystem IIIa, and the DSS asked for an "exercise electrocardiogram" to be carried out, when these new data were obtained, the knowledge extraction process would have to be restarted (Figure 4, right). Observe that this sort of loops could arise more than once.

Something similar would happen if the end user had not given the system all the required information. This is the case when one variable in the antecedent of a rule is neither stated as true nor as false, meanwhile another variable in the antecedent of the same rule is stated as true or deduced to be true by a previous subsystem (situation (ii)). The DSS would need to be rerun in such a case (after the data completion).

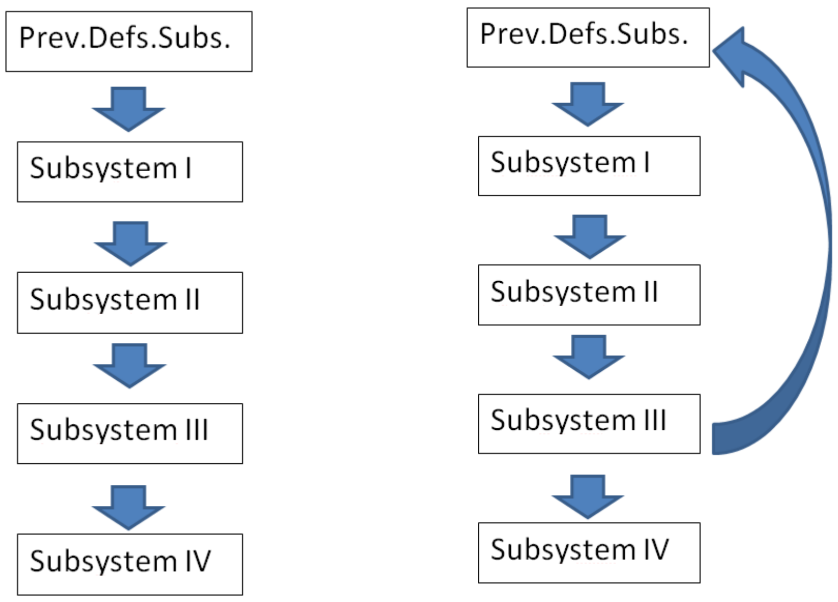

Figure 4. Example of perfect flow in the cascade of subsystems (**left**). Example of flow in the cascade of subsystems when Subsystem IIIa asks an "exercise electrocardiogram" to be carried out: when these data are obtained the knowledge extraction process is restarted (**right**).

3.2.1. Variables Considered

The variables in the antecedents and consequents of the rules in the five subsystems are organized below in in Tables 1–6. Note that these tables just list these variables, there is no special relation between two variables in the same row.

Table 1. Previous definitions subsystem (initial input data and conclusions/ask for more data).

Variables in Antecedents	**Variables in Consequents**
diastolic_murmur	physiologic_murmur
systolic_murmur	pathologic_murmur
left_side	AoI (aortic insufficiency)
right_side	MI (mitral insufficiency)
PMI_aorta (point of maximal intensity)	AoCF (aortocardiac fistula)
PMI_mitral	TI (tricuspid insufficiency)
continuous_murmur	VSD (ventricular septal defect)
grade_1_to_3	sinus_rhythm
grade_4_to_6	arrhythmia_suspicion
regular_rhythm_auscultation	
very_low_HR (heart rate)	
normal_HR	
high_HR	
justified_high_HR	
regular_rhythm_auscultation	
collapse	

Table 2. Subsystem I (first level input data and conclusions/ask for more data).

New Variables in Antecedents	New Variables in Consequents
hyperkinetic_pulse	clinically_irrelevant
hypotension	likely_clinically_relevant
ventral_oedema	clinically_relevant
jugular pulse	suspicion_of_HF
lethargy	likely_unsafe_to_handle_horse
tachycardia	

Table 3. Subsystem II (second level input data and conclusions/ask for more data).

New Variables in Antecedents	New Variables in Consequents
PP_above_60 (pulse pressure)	requires_cardiac_assessment
regurgitation_more_than_two_thirds	requires_echocardiography
LA_enlargement (left atrial)	requires_tensiometry
LV_enlargement (left ventricular)	requires_resting_ECG (echocardiography)
PA_dilation (pulmonary artery)	severe_AoI
large_VSD	severe_MI
HF (heart failure)	severe_TI
requires_cardiac_assessment	severe_VSD
	risk_collapse_at_rest

Table 4. Subsystem III (diagnoses): IIIa: arrhythmias (third level input data and conclusions/ask for more data).

New Variables in Antecedents	New Variables in Consequents
SR (sinus rhythm)	exercise_ecg_contraindicated
AVB_2o_ GMbI (second degree atrioventricular block Mobitz I)	risk_for_collapse_during_exercise
AVB_2o_GMbII (second degree atrioventricular block Mobitz II)	high_risk_collapse_during_exercise
AVB_3o_G (third degree atrioventricular block)	current_clinically_irrelevant
SVT (supraventricular tachycardia)	risk_for_poor_performance
VT (ventricular tachycardia)	requires_exercise_ecg
PVC (premature ventricular complex)	
PSVC (premature supraventricular complex)	
AF (atrial fibrillation)	
sinus_rhythm_sustained	
AVB_2o_GMbI_sustained	
AVB_2o_GMbII_sustained	

Table 5. Subsystem III (diagnoses): IIIb: murmurs (third level input data and conclusions/ask for more data).

New Variables in Antecedents	New Variables in Consequents
	riding_horse
	arrhythmia_during_exercise
	risk_ventricular_arrhythmia_during_exercise
	risk_supraventricular_arrhythmia_during_exercise
	HR_above_220

Table 6. Subsystem IV (management). Conclusions about horse management.

Variables in Antecedents	Variables in Consequents
unsafe_to_handle_horse	requires_follow-up dangerous_horse_during_exercise horse_should_be_retired handled_by_an_informed_adult_only ridden_by_an_informed_adult_only requires_cardiac_assessment

3.2.2. Knowledge Extraction Process

Let us detail the rules in the Previous Definitions Subsystem. The rules of the other subsystems are similar and are not included for the sake of space. Nevertheless, the complete *Maple 2021* code as well as all the rules of the DSS (in Spanish) can be found in the worksheet available at [32].

The 11 rules in the previous definitions subsystem are as follows (the corresponding *Maple* code can be found in Section 4):

R1: IF systolic_murmur AND left_side AND PMI_aorta AND grade_1_to_3 THEN physiologic_murmur
R2: IF systolic_murmur AND left_side AND PMI_aorta AND grade_4_to_6 THEN pathologic_murmur
R3: IF systolic_murmur AND left_side AND PMI_mitral THEN (pathologic_murmur AND MI)
R4: IF systolic_murmur AND right_side THEN (pathologic_murmur AND (TI OR VSD))
R5: IF diastolic_murmur AND left_side AND PMI_aorta THEN AoI
R6: IF continuous_murmur AND right_side THEN AoCF
R7: IF regular_rhythm_auscultation AND normal_HR THEN sinus_rhythm
R8: IF regular_rhythm_auscultation AND very_low_HR THEN arrhythmia_suspicion
R9: IF regular_rhythm_auscultation AND high_HR AND NOT justified_high_HR THEN arrhythmia_suspicion
R10: IF regular_rhythm_auscultation AND high_HR AND collapse THEN arrhythmia_suspicion
R11: IF NOT regular_rhythm_auscultation THEN arrhythmia_suspicion

Knowledge extraction is performed in the following way: for each variable in the consequents of the rules of each subsystem it is checked if it follows from the conjunction of the variables in the antecedents that were stated as true or were obtained by forward firing. It is based on the use of `Implies` command from *Maple*'s *Logic* package (see Section 4).

The process of determining through knowledge extraction that more data have to be introduced to the system (for instance that a certain test has to be carried out is identical to the process just described).

The different subsystems have, respectively, 11, 8, 7, 8/11, and 7 rules, so the whole DSS has 52 rules. Please note that the consequents of many rules are not simple (they are not Horns clauses, as required by other approaches). Simplifying the rules would result in a much bigger number of rules, but we have tried to stay as close as possible to the way the knowledge was expressed by the expert.

3.2.3. Data Completion Process

Some extra variables have to be considered for the data completion process, that are used by the data completion procedure in order to ask the end user which input data to complete.

For instance, those corresponding to the Previous Definitions Subsystem are:

knowing_if_left_side_or_right_side_is_required
knowing_if_PMI_aorta_or_mitral_is_required

knowing_if_grade_1_to_3_or_4_to_6_is_required
knowing_if_normal_or_very_low_or_high_HR_is_required
knowing_if_very_low_or_high_HR_is_justified_or_not_is_required

The process of data completion is treated separately from knowledge extraction, and it is performed by a long procedure denoted `required_check()`, written in imperative programming style (not based on knowledge extraction). It checks the rules looking for incomplete declaration of the truth of the variables in the antecedents. Then it asks the end user to assign truth values to those variables.

An example is the following: suppose that the end user had only introduced diastolic_murmur as input datum. According to rule R5 (already detailed above):

R5: IF diastolic_murmur AND left_side AND PMI_aorta THEN AoI

the user should have informed the system whether left_side holds or not as well as whether PMI_aorta holds or not. Then the system would ask the end user that:

knowing_if_left_side_or_right_side_is_required
knowing_if_PMI_aorta_or_mitral_is_required

This is achieved by many lines of trivial imperative code based on checking set membership conditions in simple conditional statements and grouped in a single procedure. The programming is simple although laborious. See Section 4 for details.

4. Maple Implementation

The system is going to be tested firstly in Spain so the variables in this first version of the implementation are in Spanish. Nevertheless, the terms are close, except *soplo* (*murmur*). It will be translated to English once tested by a team of equine veterinarians and equine cardiologists.

4.1. Maple Implementation-Process (i)

The *Maple* implementation is straightforward. A global variable `IFP` is initialized as the empty set

```
IFP := {}:
```

Rules are written almost in natural language. For instance, the rules of the Previous Definitions Subsystem (RP) are:

```
RP1:=soplo_sistólico &and lado_izdo &and PMI_aorta &and
     grado_1_a_3 &implies soplo_fisiológico:
RP2:=soplo_sistólico &and lado_izdo &and PMI_aorta &and
     grado_4_a_6 &implies soplo_patológico:
RP3:=soplo_sistólico &and lado_izdo &and PMI_mitral &implies
     (soplo_patológico &and IM):
RP4:=soplo_sistólico &and lado_dcho &implies
     (soplo_patológico &and (IT &or CIV)):
RP5:=soplo_diastólico &and lado_izdo &and PMI_aorta &implies IAO:
RP6:=soplo_continuo &and lado_dcho &implies FAOC:
RP7:=auscultación_ritmo_regular &and FC_normal &implies ritmo_sinusal:
RP8:=auscultación_ritmo_regular &and FC_muy_baja &implies
     sospecha_arritmia:
RP9:=auscultación_ritmo_regular &and FC_elevada &and
     &not justificación_para_FC_elevada &implies sospecha_arritmia:
RP10:=auscultación_ritmo_regular &and FC_elevada &and colapso
      &implies sospecha_arritmia:
```

```
RP11:=(&not auscultación_ritmo_regular) &implies sospecha_arritmia:
```

and their conjunction is stored in variable RP:

```
RP:=RP1 &and RP2 &and RP3 &and RP4 &and RP5 &and RP6 &and RP7
    &and RP8 &and RP9 &and RP10 &and RP11:
```

Similarly, rules of subsystems I, II, IIIa, IIIb and IV are named `RSIn`, `RSIIn`, `RSIIIan`, `RSIIIbn` and `RSIVn`, where `n` is a positive integer. Their conjunctions are stored in variables `RSx` where `x` is `I`, `II`, `IIIa`, `IIIb` or `IV`.

Variable `TR` stores the conjunction of all `RSj` and `Lconseq` is the list of lists of variables in the consequents of the rules of the different subsystems. Finally, procedure `exhaustive_check` checks one by one which of the consequents of the different subsystems can be deduced from the known data of the horse:

```
exhaustive_check:=proc(horse_data)
 local i,j,res;
 for i in Lconseq do
     for j in i do
         res := Implies( TR &and horse_data , j );
         if res=true then print(j) end if;
      end do;
  end do;
 end proc:
```

The approach is naive, but the number of rules is limited and at this first step we wanted to check the possibilities to develop a working prototype and to debug it.

A *Maple 2021* worksheet, including all the DSS code and the examples in Sections 4.3 and 4.4 (plus other examples) can be downloaded from the web page [32].

4.2. Maple Implementation-Process (ii)

It is performed by procedure `required_check`, a procedure written in imperative programming style, with several simple lines of code such as:

```
> if {soplo_sistólico} intersect Tuti <> {}
>    and {lado_izdo,lado_dcho} intersect IFP = {}
>    then res:=[op(res),requiere_saber_lado_izdo_o_dcho] end if:
```

where

- `intersect` is the usual set operation,
- `Tuti` is the union of the set of given data (variables sated as true by the user) and the set of variables obtained by forward firing of the rules, and
- `IFP` is the set of given data and their negations.

4.3. Example I

Please note that input lines are preceded by > symbols and output lines are centred. The computation times are expressed in seconds (the examples have been run on a standard portable computer).

In this first example, the end user just introduces one datum: `&not auscultacion_ritmo_regular`. The system gives a partial diagnosis: `sospecha_arritmia` and `sospecha_relevancia_clínica` and asks for more tests to be performed ("estudio cardiaco", "ecocardio", "tensiometría", "electro en reposo").

```
> FP := &not auscultacion_ritmo_regular:
> tiempo:=time():
> exhaustive_check(FP);
                       sospecha_arritmia
```

```
                   sospecha_relevancia_clínica
                    requiere_estudio_cardiaco
                        requiere_ecocardio
                      requiere_tensiometría
                     requiere_electro_reposo
```

Meanwhile, `exhaustive_check` asks for many data to be introduced (propositional variables that have to be stated as true or false).

```
> required_check(FP);
 Nota i: si el sistema "requiere_saber_x" añadir como cierto "x" o "&not x"
 Nota ii: si el sistema "requiere_saber_x_o_y" añadir como cierta "x" o "y"
                   ---- Prev. Subs. ----
         requiere_saber_FC_normal_o_muy_baja_o_elevada
                     ---- Subs. I ----
                     ---- Subs. II ----
                     requiere_saber_IC
                    ---- Subs. IIIa ----
                     requiere_saber_BS
                 requiere_saber_BAV_2o_GMbI
                  requiere_saber_FC_normal
                requiere_saber_BAV_2o_GMbII
                 requiere_saber_FC_muy_baja
                  requiere_saber_BAV_3o_G
                     requiere_saber_TSV
                     requiere_saber_TV
                     requiere_saber_CPV
                    requiere_saber_CPSV
                     requiere_saber_FA
                    ---- Subs. IIIb ----
> time()-tiempo;
                           0.859
```

After the new data were introduced new conclusions could be reached but also more data could be required (remember that the subsystems are arranged in a cascade and loops can appear in this cascade).

4.4. Example II

In this second example, the user introduces some data about the horse. The system gives a partial diagnosis (`IAO`). Again, `exhaustive_check` is run and asks for more data.

```
> FP := soplo_diastólico &and pulso_hipercinético &and lado_izdo &and
>      PMI_aorta:
> tiempo:=time():
> exhaustive_check(FP);
                            IAO
> required_check(FP);
 Nota i: si el sistema "requiere_saber_x" añadir como cierto "x" o "&not x"
 Nota ii: si el sistema "requiere_saber_x_o_y" añadir como cierta "x" o "y"
                       ---- Prev. Subs. ----
                 requiere_saber_grado_1_a_3_o_4_a_6
                        ---- Subs. II ----
                       ---- Subs. IIIa ----
                       ---- Subs. IIIb ----
                  requiere_saber_caballo_en_activo
                requiere_saber_arritmia_en_ejercicio
```

```
> time()-tiempo;
                                        0.750
```

After the new data were introduced new conclusions could be reached but also more data could be required (remember that the subsystems are arranged in a cascade and loops can appear in this cascade).

5. Results

The authors designed, developed and implemented a prototype of DSS for equine cardiovascular diseases diagnosis and management. This was an unexplored field, as far as we know. Therefore, an innovation of this work is the field of application of DSS techniques and knowledge.

The organization of the equine cardiology knowledge in the form of mathematical rules was a long work that required close collaboration of the two authors and a careful debugging.

The knowledge extraction part of the DSS was straightforward to implement as it uses well known techniques. It is performed by the `exhaustive_check()` procedure that exhaustively checks which diagnoses can be obtained by forward firing the rules in the DSS given the horse's data introduced by the end user.

Nevertheless, the first tests of the DSS showed that, in most cases, an unaware vetrenarian (one that was not an equine cardiologist), would not introduce much auxiliary data, which would prevent the DSS from obtaining all the appropriate conclusions. This happens because the potential facts can be stated as true or false or perhaps nothing is said about them. A rule is fired if and only if its antecedent is true, which requires all the variables in it to be adequately stated as true or false. Therefore we developed a second part of the DSS (that consists of a long procedure, denoted `required_check()`, written in imperative programming style, i.e., not based on forward firing rules) that checks rules with undefined variables in their antecedents and asks the end user to assign to those variables either the "true" or the "false" value. This data completion part of the DSS is another novelty of the work (motivated by the possible lack of specialization of the user –a veterinarian).

6. Conclusions and Further Work

We believe this could be a very useful DSS, because cardiac diseases are not very common in horses, so a small proportion of veterinarians have a background in equine cardiology. Let us underline that it does not try to substitute the specialist, but to help a general veterinarian or a veterinarian that is not an equine cardiologist to help in deciding, for instance, whether to refer the horse to a cardiologist or not or whether to retire a horse or not. The initial target end user (a general veterinarian) has to reply the questions asked by the DSS and to perform the tests required by the DSS (for instance, an ecocardiography). The possibility to carry out the latter obviously depends on their qualification and the available equipment.

Nevertheless, it can also help an equine cardiologist in obtaining a second opinion from the set of data of a real case.

This is a first design, development, and implementation of a prototype. It would have to be extensively tested and debugged, prior to real life use.

However, it also has strong educational potential. It can be used in its present state in the veterinarian cardiology classroom (in a "supervised work" approach), as well as for performing "autonomous homework". Moreover, the first approach could give the first feedback for the DSS debugging.

Although not very important as the implementation is fast, it would be more elegant if recomputing was avoided by storing the conclusions of the previous subsystems with completed data. That could be a future task.

From the computational point of view, other alternative approaches should be explored and tested. *Maple* was chosen because it is really friendly and development times are excellent, although it has the disadvantage that it is a commercial CAS, which can limit the use of the DSS.

Moreover, a comfortable GUI should be developed for its real use, as the target end user is not supposed to be a programmer. An advantage of using *Maple* is that its last versions have enhanced their possibilities to develop friendly windows-based applications.

Author Contributions: Conceptualization, M.V.-O. and E.R.-L.; methodology, M.V.-O. and E.R.-L.; software, E.R.-L.; writing—original draft preparation, M.V.-O. and E.R.-L.; writing—review and editing, M.V.-O. and E.R.-L. Both authors have read and agreed to the published version of the manuscript.

Funding: This research was Partially funded by the research project PGC2018-096509-B-I00 (Government of Spain).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

DSS	Decision Support System
KBS	Knowledge Based System
RBES	Rule Based Expert Systems
CAS	Computer Algebra System
GUI	Graphical User Interface
AF	atrial fibrillation
AoI	aortic insufficiency
AVB_2o_ GMbI	second degree atrioventricular block Mobitz I
AVB_2o_GMbII	second degree atrioventricular block Mobitz II
AVB_3o_G	third degree atrioventricular block
AoCF	aortocardiac fistula
ECG	echocardiography
HF	heart failure
HR	heart rate
MI	mitral insufficiency
PA	pulmonary artery
PP	pulse pressure
PMI	point of maximal intensity
PSVC	premature supraventricular complex
PVC	premature ventricular complex
SVT	supraventricular tachycardia
TI	tricuspid insufficiency
VSD	ventricular septal defect
VT	ventricular tachycardia.

References

1. Reef, V.B.; Bonagura, J.; Buhl, R.; McGurrin, M.K.J.; Schwarzwald, C.C.; van Loon, G.; Young, L.E. Recommendations for management of equine athletes with cardiovascular abnormalities. *J. Vet. Intern. Med.* **2014**, *28*, 749–761. [CrossRef] [PubMed]
2. Bonagura, J.D. Overview of Equine Cardiac Disease. *Vet. Clin. North Am.-Equine Pract.* **2019**, *35*, 1–22. [CrossRef] [PubMed]
3. Keen, J.A.A. Examination of Horses with Cardiac Disease. *Vet. Clin. North Am.-Equine Pract.* **2019**, *35*, 23–42. [CrossRef] [PubMed]
4. Reef, V.B. Assessment of the Cardiovascular System in Horses During Prepurchase and Insurance Examinations. *Vet. Clin. N. Am.-Equine Pract.* **2019**, *35*, 191–204. [CrossRef] [PubMed]
5. de Solis, C.N.; Althaus, F.; Basieux, N.; Burger, D. Sudden death in sport and riding horses during and immediately after exercise: A case series. *Equine Vet. J.* **2018**, *50*, 644–648. [CrossRef] [PubMed]

6. Gao, H.; Jiang, G.; Gao, X.; Xiao, J.; Wang, H. An equine disease diagnosis expert system based on improved reasoning of evidence credibility. *Inf. Process. Agric.* **2019**, *6/3*, 414–423. [CrossRef]
7. Burstein, F.; Holsapple, C. (Eds.) *Handbook on Decision Support Systems 1. Basic Themes*; Springer: Berlin/Heidelberg, Germany, 2008.
8. Payne, T.H. Computer Decision Support Systems. *Chest* **2000**, *118* (Suppl. 2), 47S–52S. [CrossRef] [PubMed]
9. Roanes-Lozano, E.; Laita, L.M.; Hernando, A.; Roanes-Macías, E. An algebraic approach to rule based expert systems. *Rev. R. Acad. Cienc. Exactas Fis. Nat. Ser. A Mat. RACSAM* **2010**, *104*, 19–40. [CrossRef]
10. Piury, J.; Laita, L.M.; Roanes-Lozano, E.; Hernando, A.; Piury-Alonso, F.J.; Gómez-Argüelles, J.M.; Laita, L. A Gröbner bases-based rule based expert system for fibromyalgia diagnosis. *Rev. R. Acad. Cienc. Exactas Fis. Nat. Ser. A Mat. RACSAM* **2012**, *106*, 443–456. [CrossRef]
11. Roanes-Lozano, E.; Roanes-Macías, E.; Laita, L.M. Railway Interlocking Systems and Groebner Bases. *Math. Comput. Simul.* **2000**, *51*, 473–481. [CrossRef]
12. Roanes-Lozano, E.; Hernando, A.; Alonso, J.A.; Laita, L.M. A logic approach to decision taking in a railway interlocking system using Maple. *Math. Comput. Simul.* **2011**, *82*, 15–28. [CrossRef]
13. Gelfond, M. Answer sets. In *Handbook of Knowledge Representation*; van Harmelen, F., Lifschitz, V., Porter, B., Eds.; Elsevier: Amsterdam, The Netherlands, 2008; pp. 285–316.
14. Gelfond, M.; Lifschitz, V. The stable model semantics for logic programming. In Proceedings of the Fifth International Conference on Logic Programming, ICLP-88, Seattle, WA, USA, 15–19 August 1988; MIT Press: Cambridge, MA, USA, 1988; pp. 1070–1080.
15. Niemelä, I. Logic programs with stable model semantics as a constraint programming paradigm. *Ann. Math. Artif. Intell.* **1999**, *25*, 241–273. [CrossRef]
16. Niemelä, I.; Simons, P. Smodels–an implementation of the stable model and well-founded semantics for normal logic programs. In Proceedings of the Logic Programming and Nonmonotonic Reasoning: Fourth International Conference, LPNMR'97, Dagstuhl Castle, Germany, 28–31 July 1997; Lecture Notes in Artificial Intelligence; Springer: Berlin/Heidelberg, Germany, 1997; Volume 1265, pp. 420–429.
17. Smodels Web Page. Available online: http://www.tcs.hut.fi/Software/smodels/ (accessed on 20 July 2021).
18. Roanes-Lozano, E.; Alonso, J.A.; Hernando, A. An approach from answer set programming to decision making in a railway interlocking system. *Rev. R. Acad. Cienc. Exactas Fis. Nat. Ser. A Mat. RACSAM* **2014**, *108*, 973–987. [CrossRef]
19. Alonso-Jiménez, J.A.; Aranda-Corral, G.A.; Borrego-Díaz, J.; Fernández-Lebrón, M.M.; Hidalgo-Doblado, M.J. A logic-algebraic tool for reasoning with Knowledge-Based Systems. *J. Log. Algebr. Methods Program.* **2018**, *101*, 88–109. [CrossRef]
20. Aranda-Corral, G.A.; Borrego-Díaz, J.; Galán-Páez, J.; Rodríguez-Chavarría, D. Towards a Notion of Basis for Knowledge-Based Systems-Applications. *Mathematics* **2021**, *9*, 252. [CrossRef]
21. Peña-Cadahia, C.; Manso-Díaz, G.; Santiago-Llorente, I.; Villalba-Orero, M. Accelerated Idioventricular rhythm associated with Isoflurane Administration in a Foal: A case report. *J. Equine Vet. Sci.* **2019**, *80*, 64–68. [CrossRef] [PubMed]
22. Padrón-Barthe, L.; Villalba-Orero, M.; Gómez-Salinero, J.M.; Domínguez, F.; Román, M.; Larrasa-Alonso, J.; Ortiz-Sánchez, P.; Martínez, F.; López-Olañeta, M.; Bonzón-Kulichenko, E.; et al. Severe cardiac dysfunction and death caused by Arrhythmogenic right ventricular cardiomyopathy type 5 are improved by inhibition of glycogen synthase kinase-3β. *Circulation* **2019**, *140*, 1188–1204. [CrossRef] [PubMed]
23. Padrón-Barthe, L.; Villalba-Orero, M.; Gómez-Salinero, J.M.; Acín-Pérez, R.; Cogliati, S.; López-Olañeta, M.; Ortiz-Sánchez, P.; Bonzón-Kulichenko, E.; Vázquez, J.; García-Pavía, P.; et al. Activation of serine one-carbon metabolism by calcineurin $A\beta1$ reduces myocardial hypertrophy and improves ventricular function. *J. Am. Coll. Cardiol.* **2018**, *71*, 654–667. [CrossRef] [PubMed]
24. Wester, M.J. *Computer Algebra Systems: A Practical Guide*; John Wiley & Sons: Chichester, UK, 1999.
25. Bernardin, L.; Chin, P.; DeMarco, P.; Geddes, K.O.; Hare, D.E.G.; Heal, K.M.; Labahn, G.; May, J.P.; McCarron, J.; Monagan, M.B.; Ohashi, D.; Vorkoetter, S.M. *Maple Programming Guide*; Maplesoft, Waterloo Maple Inc.: Waterloo, ON, Canada, 2020. Available online: https://www.maplesoft.com/documentation_center/maple2020/ProgrammingGuide.pdf (accessed on 20 July 2021).
26. Burkhardt, W. *First Steps in Maple*; Springer: London, UK, 1994.
27. Corless, R. *Essential Maple. An Introduction for Scientific Programmers*; Springer: New York, NY, USA, 1995.
28. Heck, A. *Introduction to Maple*; Springer: New York, NY, USA, 2003.
29. Maplesoft. *Maple User Manual*; Maplesoft, Waterloo Maple Inc.: Waterloo, ON, Canada, 2021. Available online: https://www.maplesoft.com/documentation_center/maple2021/UserManual.pdf (accessed on 20 July 2021).
30. Sendra, J. R.; Pérez-Díaz, S.; Sendra, J.; Villarino, C. *Introducción a la Computación Simbólica y Facilidades Maple*; Ra-Ma: Madrid, Spain, 2012.
31. Recio, T.; Vélez, M.P. Automatic Discovery of Theorems in Elementary Geometry. *J. Autom. Reas.* **1999**, *23*, 63–82. [CrossRef]
32. RBES Cardio Equino. Available online: https://www.ucm.es/info/secdealg/Ejemplos_Maple_CardioEquino.mw (accessed on 8 October 2021).