Applied Mathematics and Computation 000 (2016) 1–10

Contents lists available at ScienceDirect

Applied Mathematics and Computation

journal homepage: www.elsevier.com/locate/amc

A prototype of a RBES for personalized menus generation

Eugenio Roanes-Lozano[a], José Luis Galán-García[b], Gabriel Aguilera-Venegas[b,*]

[a] Departamento de Algebra, Instituto de Matemática Interdisciplinar, Universidad Complutense de Madrid, Spain
[b] Departamento de Matemática Aplicada, Universidad de Málaga, Spain

ARTICLE INFO

Article history:
Available online xxx

Keywords:
Knowledge-based systems
Restaurant menus
Computer algebra
Groebner bases

ABSTRACT

People have many constraints concerning the food they eat. These constraints can be based on religious believes, be due to food allergies or illnesses, or be derived from personal preferences or dislikes. For instance, preparing the menus at a hospital can be really complex. Another special situation arises when traveling abroad or simply when eating at a foreign cuisine restaurant (it is not always enough to know the brief description in the restaurant's menu or the explanation of the waiter). Therefore, we consider that it would be very interesting to develop a knowledge-based system that automatically obtained a personalized menu for each customer, according to the precise recipes of the restaurant and taking into account the data given by the customer and the ingredients out of stock (if any). Although there are many knowledge-based systems devoted to diabetic's meals, diets, food supply chains, etc., we do not know of any comparable system. We have developed a rule based expert system that uses sets and lists for handling data and an algebraic inference engine. It has been implemented in the computer algebra system *Maple*.

1. Introduction

People have many constraints concerning the food they eat. These constraints can be based on religious believes, be due to food allergies or illnesses, or be derived from personal preferences or dislikes.

For instance, preparing the menus at a hospital can be really complex [1].

Another special situation arises when traveling abroad or simply when eating at a foreign cuisine restaurant: it is not always enough to know the brief description in the restaurant's menu or the explanation of the waiter.

For example, "calamares en su tinta" (squid in its own ink) is a delicious typical Spanish dish, not well-known abroad. Its brief description would be "squid with boiled rice in its own (black) ink". But an ingredient (included in a small amount, in order to thicken the sauce) is flour, a fact very important for a celiac.

Another example is meat, sometimes marinated in milk before cooking (to make it tenderer), what is not specified in the menu. This can be very important for people with allergy to milk and for Jewish.

Therefore, we consider that it would be very interesting to develop a knowledge-based system that addressed these problems: it should have information about allergies, other diet requirements (vegetarian, vegan,...) and religious constraints, know absolutely all the ingredients used by the chef when preparing the different dishes and check the situation for each customer. As the knowledge is fixed and the constraints are known, a rule based expert system (RBES) whose underlying logic is classic Boolean matches the task.

* Corresponding author.
E-mail addresses: eroanes@mat.ucm.es (E. Roanes-Lozano), jlgalan@uma.es (J.L. Galán-García), gabri@ctima.uma.es (G. Aguilera-Venegas).

http://dx.doi.org/10.1016/j.amc.2016.12.023
0096-3003/

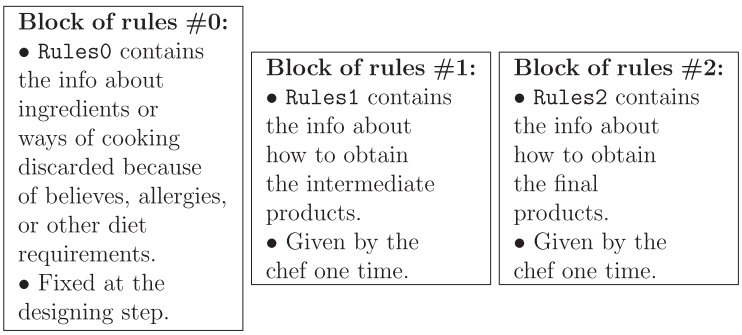

Fig. 1. The three blocks of rules of the package.

The key idea is to develop a RBES as comfortable as possible for both the restaurant and the customer. More precisely, we distinguish:

- Fixed data: ingredients or ways of cooking discarded because of believes, allergies or other diet requirements and recipes of the restaurant.
- Unfixed data: ingredients out of stock at a certain moment and customer's data.

The package designed constructs a "personalized restaurant menu" using set operations and knowledge extraction, thanks to an algebraic Groebner bases [2] based inference engine (although other approaches could be used instead).

It has been implemented in the computer algebra system *Maple 2016* (using its convenient "Embedded-Components") and can be run from computers, tablets and the iPad using *Maple*.

This paper extends talks presented at Applications of Computer Algebra 2015 (ACA 2015) conference [3] and the 5th European Seminar on Computing (ESCO 2016).

2. Generalities of the system: about the fixed data

Regarding food, we distinguish three kind of items:

- ingredients and ways of cooking (note that the ways of cooking the food is considered by the system at the same level as ingredients),
- intermediate products, that do not directly appear in the restaurant's menu (a sauce, like "mayonnaise", is neither an ingredient nor a final product), and
- final products (like "seafood cocktail"), that are the dishes listed in the restaurant's menu.

The rules derive directly from the "recipes" (that have to be specified by the chef) and only contain the information about the required ingredients, way(s) of cooking and names of the dishes. The rules can be grouped in three blocks:

- the ingredients or ways of cooking discarded because of believes, other diet requirements or allergies are fixed at the designing step,
- the information about how to obtain the intermediate products has to be manually introduced by the chef (as they can depend on the restaurant), but it has to be done only one time (unless he/she changes any recipe),
- the information about how to obtain the final products also has to be manually introduced by the chef (as they can depend on the restaurant), but, again, it has to be done only one time (unless he/she changes any recipe).

These blocks of rules are denoted `Rules0`, `Rules1` and `Rules2`, respectively (Fig. 1).
When the menu is going to be computed:

- the restaurant has to precise the ingredients out of stock at that moment (if any), and,
- the customer has to introduce the religion, allergies and other restrictions due to illnesses or personal preferences.

3. Fixed data introduction

The connectives than can be used in the rules are
`NEG, &AND, &OR, &IMP, &XOR`
(in our implementation the unary one is prefix and the binary ones are infix ones).
As said before, the data introduction step is performed just by:

- storing rules that describe the forbidden ingredients (for believers, allergic reasons or other diet requirements) in set `Rules0`,

Please cite this article as: E. Roanes-Lozano et al., A prototype of a RBES for personalized menus generation, Applied Mathematics and Computation (2016), http://dx.doi.org/10.1016/j.amc.2016.12.023

- storing the rules that describe the "recipes" of the intermediate products in set `Rules1`,
- storing the rules that describe the "recipes" of the final products in set `Rules2`.

These sets of rules are stored altogether in file `RULES.TXT`.

3.1. Data introduction at the designing step

The set of rules `Rules0` is fixed and defines the special requirements of allergies, believers and people with other diet requirements. This is introduced at the "designing step" and, initially, is not supposed to be changed by the chef.

In this first prototype the following types of diets based on religious believes, allergies or other specific diet requirements have been considered:

CELIAC, JEWISH, MUSLIM, VEGETARIAN, VEGAN

The rules in Rules0 must be of the form:

```
CELIAC &IMP bread
```

(indicating that "bread" is forbidden to a "celiac"), and should not include any other logic connectives.

Regarding how animals are slaughtered, we distinguish, for instance, three kinds of lamb: `lamb_j`, `lamb_m`, `lamb`, indicating that the Jewish ("kosher") ritual animal slaughter [4,5], or the Muslim ritual animal slaughter (so that the meat is "halal" [6]), or none special ritual animal slaughter has been followed (respectively). As the rules in set `Rules0` indicate prohibitions, we include rules such as:

```
MUSLIM &IMP lamb
MUSLIM &IMP lamb_j
```

informing the system that lamb slaughtered following no special ritual or following the Jewish ritual is forbidden for a Muslim.

Let us underline that `VEGAN` is more restrictive than `VEGETARIAN`. The system is able to avoid duplicating rules by adding the following one:

```
VEGAN &IMP VEGETARIAN
```

This way, all restrictions introduced for `VEGETARIAN` are inherited by `VEGAN` (obviously, some other restrictions have been included for `VEGAN`).

3.2. Data introduction by the chef – I

The rules in `Rules1` are of the form:

```
(oil &AND eggs &AND vinegar &AND salt &AND CRUDE) &IMP mayonnaise
```

(indicating how the intermediate product "mayonnaise" is prepared by the chef). They remain unchanged unless the chef decides to change any recipe.

The antecedents of a rule are ingredients and way(s) of cooking, and the consequent is an intermediate product. The ingredients could also include other intermediate products, as happens in the following example:

```
(mayonnaise &AND ketchup &AND orange_juice &AND lemon_juice &AND brandy &AND CRUDE)
&IMP pink_sauce
```

3.3. Data introduction by the chef – II

The rules in `Rules2` are of the form:

```
(king_prawn &AND shrimp &AND lettuce &AND pink_sauce
&AND CRUDE) &IMP seafood_cocktail
```

(indicating how the final product "seafood_cocktail" is prepared by the chef). They remain unchanged unless the chef decides to change any recipe.

The antecedents are ingredients, intermediate products and ways of cooking, and the consequent is a final product.

It is possible to use logical disjunctions in the rules detailing the recipes, for instance in order to include in a single rule the Muslim and Jewish options. An example can be found in the following rule:

```
((beef &OR beef_j &OR beef_m) &AND salt &AND pepper &AND oil &AND GRILLED) &IMP
grilled_beef_steak
```

4. Unfixed data introduction

4.1. Data introduction by the chef – III

The chef also has to introduce the ingredients out of stock at each moment, if any. This set of ingredients will be stored in variable `INS`.

Please cite this article as: E. Roanes-Lozano et al., A prototype of a RBES for personalized menus generation, Applied Mathematics and Computation (2016), http://dx.doi.org/10.1016/j.amc.2016.12.023

4.2. Data introduction by the customer

The customer can introduce, if desired:

- religion, allergies and other special diets (that will be stored in variable PCO),

as well as:

- forbidden ingredients/ways of cooking (that will be stored in variable FIR),
- forbidden intermediate products (that will be stored in variable FIP), and
- forbidden final products (that will be stored in variable FFP)

(to forbid an ingredient, ways of cooking or product can be based on a diet, a dislike, etc.).

5. Description of the processes carried out

We can distinguish three processes within the package, that we have denoted:

- Preprocess
- Main Process
- Postprocess.

5.1. Preprocess

The package is intended to be as comfortable as possible. Therefore, the system obtains all the information included in the rules, and the following data don't have to be typed by the chef. This is carried out at this "Preprocess" step.

The set of possible intermediate products (those usable in the best case) are obtained from the consequents of the rules in Rules1 and are stored in variable IP.

The set of possible final products (those offerable in the best case) are obtained from the consequents of the rules in Rules2 and are stored in variable FP.

The set of ingredients and ways of cooking is obtained from the variables in Rules1 ∪ Rules2, excluding those variables in IP ∪ FP. They are stored in variable IR.

The set of all the religious, allergies and other special diets considered are stored in set RAC, meanwhile all the ingredients that could be forbidden for these reasons are stored in set CBFI.

Finally, the variables involved in the rules in Rules0 are stored in variable COV.

These processes take advantage of the possibilities of *Maple* for handling sets (the usual set operations are *Maple* commands).

5.2. Main Process

5.2.1. Step i of the Main Process

The process begins by converting the set of variables involved, with their complete names in English (ingredients, intermediate and final products, and variables involved in the rules in Rules0), IR ∪ IP ∪ FP ∪ COV, into a list, denoted LSV_.

The package also builds a list of variables of the form x[j], of the same length as list LSV_, that will be used as internal nicknames of the variables with complete names in English (in LSV_), for the sake of brevity of the algebraic computations to be carried out. They are stored in list LSV.

It also constructs the ideal iI, that introduces idempotency in the polynomial ring where the inference engine works (see Section 6).

5.2.2. Step ii of the Main Process: obtaining the usable ingredients

Forward firing is applied to the religious, allergies and special diet data of the customer (stored in PCO) and the rules in Rules0 (the latter describing the consequences of his/her believes, allergies or special diets).

To achieve this, we consider (see Section 6):

- ideal iK, generated by the algebraic translation of (the negation of) the variables in PCO,
- ideal iJ, generated by the algebraic translation of (the negation of) the rules in Rules0,
- and compute the "reduced Groebner basis" of ideal iI+iJ+iK, denoted BO.

Now, of the products that could be forbidden, CBFI, we check, using command NormalForm, which are really forbidden for this particular customer (and they are stored in variable INR).

Finally, of all the ingredients (IR) we exclude

- those not in stock (in INS),
- those forbidden by the customer (in FIR)
- and those not usable because of religious reasons/allergies/special diets (in INR),

obtaining this way the set of usable ingredients (UI).

Please cite this article as: E. Roanes-Lozano et al., A prototype of a RBES for personalized menus generation, Applied Mathematics and Computation (2016), http://dx.doi.org/10.1016/j.amc.2016.12.023

5.2.3. Step iii of the Main Process: obtaining the usable intermediate products

Forward firing is applied to the usable ingredients (in `UI`) and the rules in `Rules1`.

To achieve this, we consider:

- ideal `iK`, generated by the algebraic translation of (the negation of) the variables in `UI`,
- ideal `iJ`, generated by the algebraic translation of (the negation of) the variables in `Rules1`,

and compute the "reduced Groebner basis" of `iI+iJ+iK`, denoted `B1`.

Now, of the possible intermediate products (stored in `IP`) that are not forbidden (i.e., not in `FIP`), we check, using command "NormalForm", which can be prepared for this particular customer, starting with the allowed ingredients, and we store the result in the set `UIP` (of usable intermediate products).

5.2.4. Step iv of the Main Process: obtaining the offerable final products

Forward firing is applied to the usable ingredients (stored in `UI`) and usable intermediate products (stored in `UIP`), and the rules in `Rules2`.

To achieve this, we consider:

- ideal `iK`, generated by the algebraic translation of (the negation of) the variables in `UI` $\cup$ `UIP`,
- ideal `iJ`, generated by the algebraic translation of (the negation of) the variables in `Rules2`,

and compute the "reduced Groebner basis" of `iI+iJ+iK`, denoted `B2`.

Now, of the possible final products (stored in `FP`) that are not forbidden (`FFP`), we check, using command "NormalForm", which can be prepared for this particular customer, by starting with the allowed ingredients and intermediate products, and store the result in set `UFP` (of usable final products).

5.3. Postprocess

After unassigning the names of the variables `x[j]`, the postprocess consists of applying an auxiliary procedure that, for each `x[j]` in `UFP`, looks for the jth element in the list of variables `LSV_`, recovering this way the "usual" names of the products in the set of usable final products (`UFP`).

5.4. About "kosher" restrictions

According to the Jewish "kosher" prohibitions [4], meat and dairy products cannot be mixed in the same meal. Therefore, the program has to be run twice in case the customer is Jewish, one time including "dairy" (a variable that is a set of products, defined within the file `Preprocess.mpl`) in the forbidden ingredients (`FIR`) and the other including "meat" (another set already defined). Other ingredients can be also included as forbidden.

Let us remark that the system only analyzes the ingredients and ways of cooking. The restaurant is responsible for observing the correct rules regarding set of utensils, dishes, etc.

6. The algebraic approach to RBES

(This section can be skipped by readers aware of this approach.)

Let $(\mathcal{C}, \vee, \wedge, \neg, \rightarrow)$ be the Boolean algebra of the propositions that can be constructed using a finite number of propositional variables $P, Q, \ldots, R$. Let us consider the Boolean algebra $(\mathcal{A}, \tilde{+}, \cdot, 1+,$ "is a multiple"), where

$$\mathcal{A} = \mathbb{Z}_2[p, q, \ldots, r]/\langle p^2 - p, q^2 - q, \ldots, r^2 - r\rangle$$

and $p\tilde{+}q = p + q - p \cdot q$. Let us define:

φ: $(\mathcal{C}, \vee, \wedge, \neg, \rightarrow) \longrightarrow (\mathcal{A}, \tilde{+}, \cdot, 1+,$"is a multiple") the following way; for propositional variables:

$$\begin{array}{c} P \longrightarrow p \\ Q \longrightarrow q \\ \ldots \\ R \longrightarrow r \end{array}$$

and for any $A, B \in \mathcal{C}$:

$$\begin{array}{c} A \vee B \longrightarrow a\tilde{+}b \\ \neg A \longrightarrow 1 + a \end{array}$$

Then, as an immediate consequence of the De Morgan laws:

$$A \wedge B \longrightarrow a \cdot b$$

and the correspondence φ turns out to be a Boolean algebra isomorphism.

Please cite this article as: E. Roanes-Lozano et al., A prototype of a RBES for personalized menus generation, Applied Mathematics and Computation (2016), http://dx.doi.org/10.1016/j.amc.2016.12.023

Moreover, if $\vee$ is substituted by *XOR* in the left hand side and $\tilde{+}$ is substituted by $+$ in the right hand side, a Boolean ring isomorphism is obtained.

The main result translates "checking a tautological consequence" into a "polynomial ideal membership":

Theorem 1. *A propositional formula α is a tautological consequence of a set of formulae $\{\beta_1, \ldots, \beta_m\}$, if and only if*

$$\varphi(\neg\alpha) \in \langle \varphi(\neg\beta_1), \ldots, \varphi(\neg\beta_m) \rangle$$

Therefore "knowledge extraction" in a RBES can be performed in the polynomial model by checking whether the polynomial that is the translation of the negation of the logic formula belongs to the ideal J, an ideal generated by the polynomial translation of the negation of certain formulae (facts, rules and integrity constraints) in the polynomial ring $\mathcal{A}$.

This can be checked with a computer algebra system by computing whether the normal form of the polynomial modulo the ideal J is 0 or not in $\mathcal{A}$.

Regarding RBES "logical inconsistency" (what happens when a statement turns out to be true and false at the same time), it is translated in the polynomial model in the degeneracy of the quotient ring $\mathcal{A}/J$ into a ring with only one element (a ring where $0 = 1$).

This can be checked with a computer algebra system by calculating whether the Groebner Basis of J in $\mathcal{A}$ is $\{1\}$ or not.

Details of this model of Boolean and modal many-valued logics and of RBES whose underlying logic is one of these logics can be found in [7–9]. These works extend previous works about performing effective computations in Boolean [10,11] and modal many-valued logics [12,13]. We have used this approach in different expert systems, mainly devoted to diagnosis in medicine [14–17].

We have chosen this approach because of our experience with it [18] and *Maple*'s possibility to easily deal with lists and sets. Nevertheless, other approaches such as, for instance, logic programing (using *Prolog*) or answer set programing (using *Smodels*) could have been used instead.

The algebraic approach has the advantage that, if during an advanced step it is decided to move to a modal many-valued logic instead, few changes have to be made.

7. Examples

The prototype `Rules0` consists of 140 rules, `Rules1` consists of 4 rules and `Rules2` consists of 22 rules.

Two examples are included below as illustration. They are carried out interactively in a *Maple Worksheet* firstly. The execution of both examples through a simple GUI is shown afterwards.

7.1. Example 1: the customer is a celiac Muslim and the restaurant declares two ingredients as out of stock

After introducing the data, the system computes the intermediate products that could be prepared in the restaurant that day (according to the stock):

```
> read(‘Preprocess.mpl‘):
> FIR := {}:                        #forbidden ingredients
> INS := {lamb_m, oranges};        #out of stock
> PCO := {CELIAC, MUSLIM};         #religions, allergies, diets
> IP;
          {lemon_juice, mayonnaise, orange_juice, pink_sauce}
```

The customer can now reject the intermediate products that he/she dislikes (in `FIP`) and the system then computes all the final products that could be prepared in the restaurant that day (according to the stock and the intermediate products rejected by the customer):

```
> FIP := {}:                  #forbidden intermediate products
> FP;

{ajoblanco, cheese_cake, creme_caramel, fried_calamari,
gazpacho, gazpachuelo, grilled_hake, mixed_paella,
russian_salad, seafood_paella, sponged_cake,
asparragus_gazpachuelo, eggplant_with_honey,
grilled_beef_steak, grilled_chicken_steak, grilled_pork_steak,
lemon_butter_baked_cod, seafood_cocktail, spanish_omelette,
squid_in_its_own_ink, vegetarian_paella,
lamb_in_veg_sauce_w_rosemary}
```

Please cite this article as: E. Roanes-Lozano et al., A prototype of a RBES for personalized menus generation, Applied Mathematics and Computation (2016), http://dx.doi.org/10.1016/j.amc.2016.12.023

Finally, the system computes the final products appropriate for that particular customer (after reading the possible final products, the customer can still reject some of them, assigning variable FFP accordingly):

```
> FFP := {};                    #forbidden final products
> read ‘Process.mpl‘:
> read ‘Postprocess.mpl‘;

[creme_caramel, gazpachuelo, grilled_hake, russian_salad,
seafood_paella, asparragus_gazpachuelo, grilled_beef_steak,
spanish_omelette, vegetarian_paella]
```

Let us underline that the final product `lamb_in_veg_sauce_w_rosemary` is not obtained since it contains the ingredient `lamb_m`, that is declared out of stock. Note that, although the ingredients `lamb` and `lamb_j` are available, the final product is not offered since the customer is Muslim.

The preceding example is carried out in a standard computer with an i5 processor in 16.24 s.

7.2. Example 2: the customer is vegetarian and informs that he/she does not like artichoke

In this case, the data introduction and knowledge extraction processes would be:

```
> read(‘Preprocess.mpl‘):
> FIR := {artichoke}:           #forbidden ingredients
> INS := {};                    #out of stock
> PCO := {VEGETARIAN};          #religions, allergies, diets
> IP;
          {lemon_juice, mayonnaise, orange_juice, pink_sauce}
> FIP := {}:                    #forbidden intermediate products
> FP;
{ajoblanco, cheese_cake, creme_caramel, fried_calamari,
gazpacho, gazpachuelo, grilled_hake, mixed_paella, russian_salad,
seafood_paella, sponged_cake, asparragus_gazpachuelo,
eggplant_with_honey, grilled_beef_steak, grilled_chicken_steak,
grilled_pork_steak, lemon_butter_baked_cod, seafood_cocktail,
spanish_omelette, squid_in_its_own_ink, vegetarian_paella,
lamb_in_veg_sauce_w_rosemary}
> FFP := {};                    #forbidden final products
> read ‘Process.mpl‘:
> read ‘Postprocess.mpl‘;
[ajoblanco, cheese_cake, creme_caramel, gazpacho, sponged_cake,
asparragus_gazpachuelo, eggplant_with_honey, spanish_omelette]
```

Let us underline that the final product `vegetarian_paella` is not obtained since it contains the ingredient `artichoke`, that the customer has declared to dislike.

The preceding example is carried out in a standard computer with an i5 processor in 14.836 s.

7.3. Graphic user interface

For this prototype, a GUI has been implemented in *Maple 2016* using its “Embedded-Components”. Figs. 2 and 3 show Examples 1 and 2 (respectively) using this simple GUI.

The user only has to fill the spaces (if desired) and click the buttons (from top to bottom). At some steps the system returns information (for example the intermediate products available).

8. Related works

There are very many works describing computer applications to food and nutrition. For instance [19] focuses on computations regarding nutrition.

Please cite this article as: E. Roanes-Lozano et al., A prototype of a RBES for personalized menus generation, Applied Mathematics and Computation (2016), http://dx.doi.org/10.1016/j.amc.2016.12.023

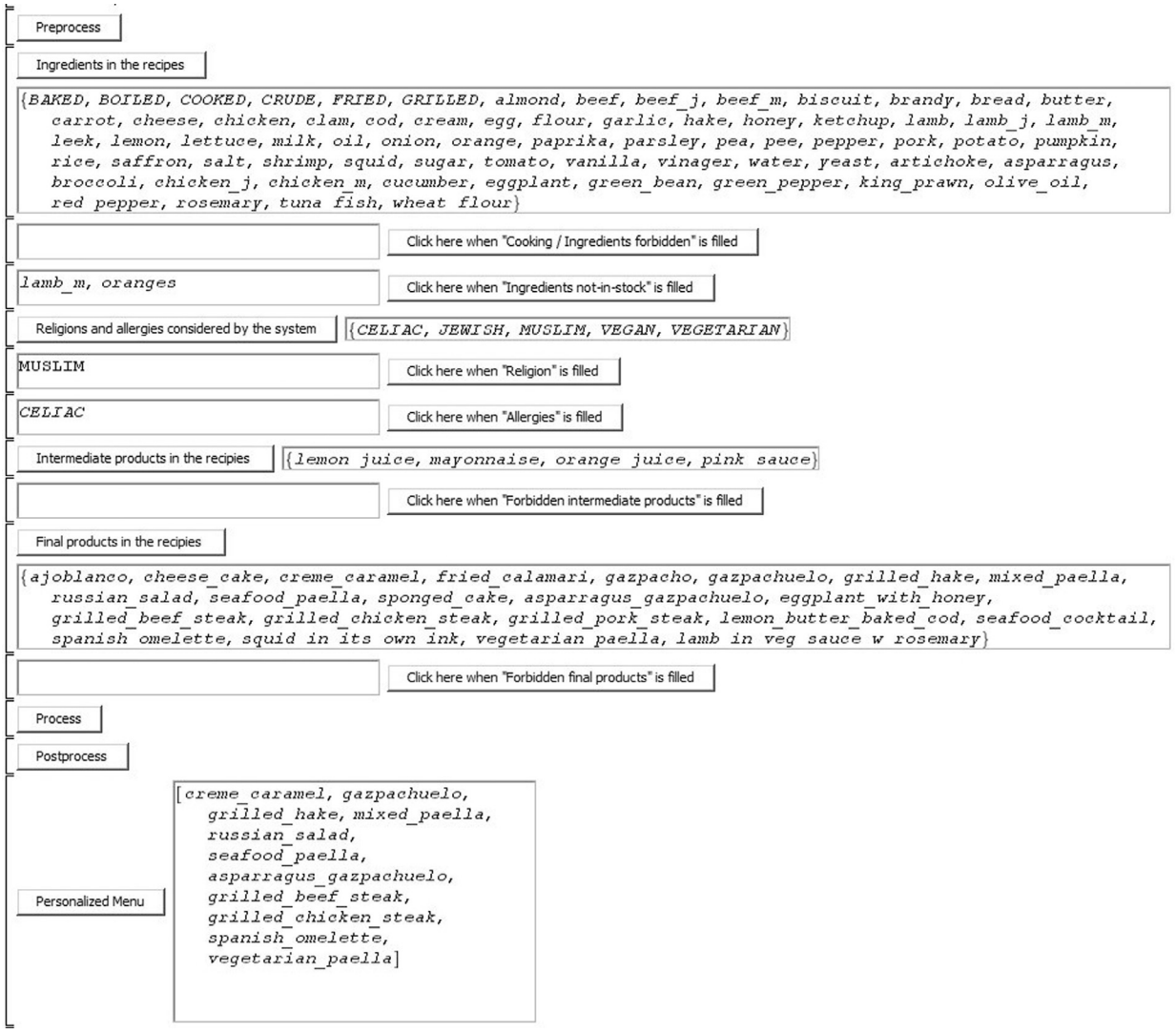

Fig. 2. Example 1 using the GUI.

In [20] a menu is created using rough sets and the knowledge (information about preferences) stored in large databases available from the Web.

Another example of applications is the location of the restaurant that fits best to the potential customer's characteristics [21,22].

A closer work to the one presented here is [23], an ontology-driven system for menu planning considering ingredients, nutrition facts, prices, etc.

But we know of no other work with our approach: obtaining a taylor-made for each customer according to his/her characteristics and preferences and the updated stock of the restaurant.

9. Conclusions

We have presented a prototype of what is, as far as we know, a new and useful application of RBES.

We do not pretend to have implemented a complete and detailed KBS on the topic, but a possible design and a simple example of development of what could be done (as illustration). More recipes should be included and experts on allergies, and on other diet requirements (vegetarian, vegan,...) as well as on Islamic and Jewish dietary laws should be consulted (this is a prototype proposing an architecture, and only elementary bibliography on these topics has been consulted).

We would like to underline that many steps are completely automatized and, for instance ingredients or ways of cooking do not need to be declared as they are obtained directly from the rules (recipes).

Please cite this article as: E. Roanes-Lozano et al., A prototype of a RBES for personalized menus generation, Applied Mathematics and Computation (2016), http://dx.doi.org/10.1016/j.amc.2016.12.023

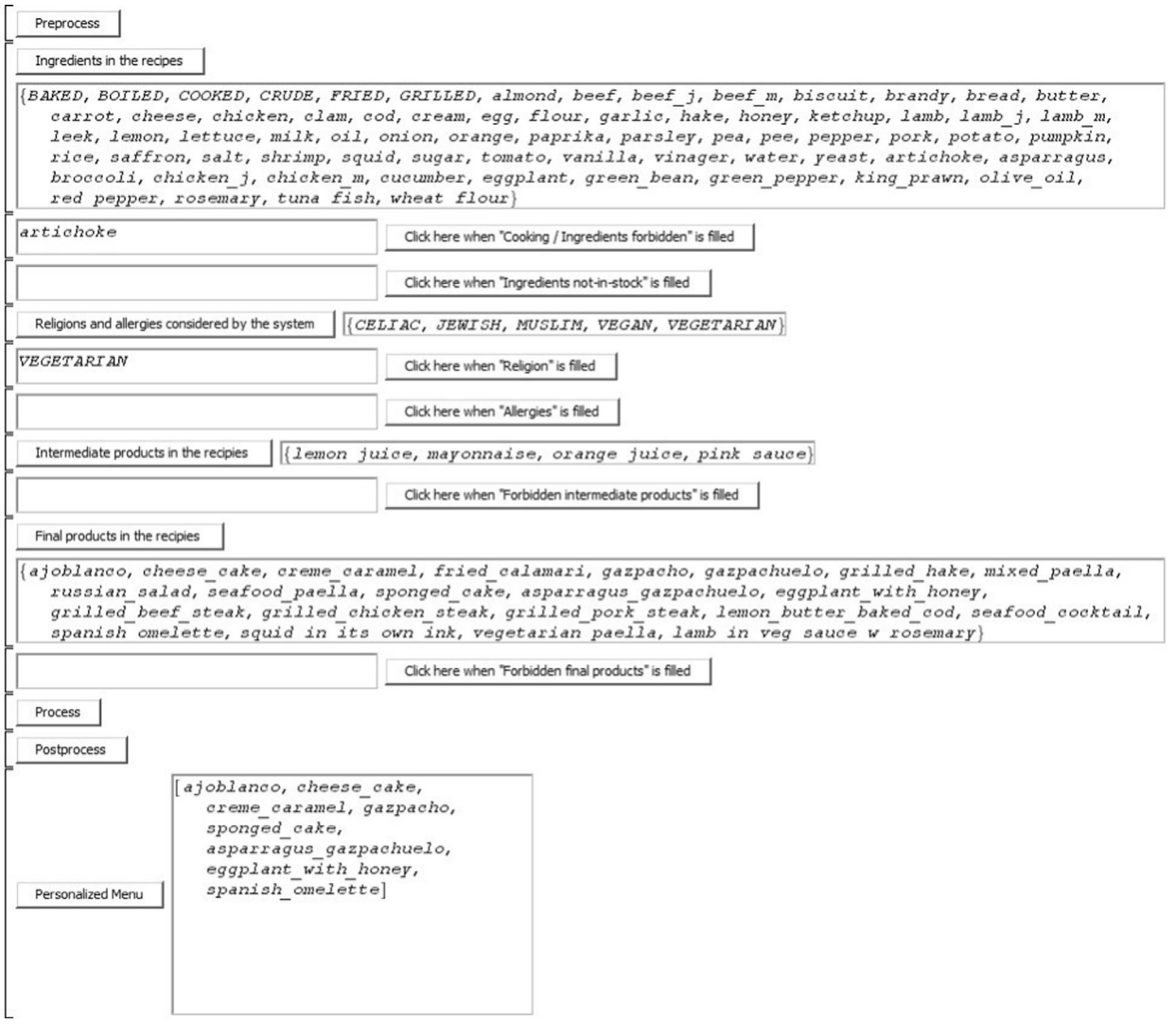

Fig. 3. Example 2 using the GUI.

The prototype uses an algebraic inference engine and is implemented in *Maple*, but neither using the algebraic approach nor implementing it in *Maple* are necessary, they have been chosen because of the experience of the authors in this approach and *Maple*'s friendliness.

In order the system to be used in restaurants, it could be advisable to port it to a free computational language. Using a free CAS like *CoCoA* would be rather straightforward. Other related approaches such as, for instance, logic programing (using *Prolog*) or answer set programing (using *Smodels*) could also be used. It would be advisable that the restaurants offered tablets or small laptops so that the customers could introduce their data in order the computer of the restaurant to provide the personalized menus (a waiter could alternatively introduce the data and bring back a printed personalized menu).

Acknowledgments

This work was partially supported by the research projects TIN2015-66471-P (Government of Spain) and CASI-CAM S2013/ICE-2845 (Comunidad Autónoma de Madrid); the Research Group ACEIA and by Grant no. TIN2011-28084 of the Science and Innovation Ministry of Spain, co-funded by the European Regional Development Fund (ERDF).

Finally, we thank the anonymous reviewers for their suggestions and comments, which have improved the quality of the paper.

Please cite this article as: E. Roanes-Lozano et al., A prototype of a RBES for personalized menus generation, Applied Mathematics and Computation (2016), http://dx.doi.org/10.1016/j.amc.2016.12.023

References

[1] Manual de Utilización del Código de Dietas del Hospital Regional Universitario Carlos Haya, Servicio Andaluz de Salud, Conserjería de Salud, Málaga, 2012.
[2] B. Buchberger, Bruno Buchberger's Ph.D. thesis 1965: an algorithm for finding the basis elementals of the residue class ring of a zero dimensional polynomial ideal, J. Symb. Comput. 41 (3–4) (2006) 475–511, doi:10.1016/j.jsc.2005.09.007.
[3] E. Roanes-Lozano, J.L. Galán-García, G. Aguilera-Venegas, Computer Algebra-based RBES personalized menu generator (Abstract) http://math.unm.edu/~aca/ACA/2015/Nonstandard/RoanesLozano.pdf (accessed 3.12.16).
[4] Wikipedia, Kashrut https://en.wikipedia.org/wiki/Kashrut (accessed 3.12.16).
[5] Wikipedia, Jewish cuisine https://en.wikipedia.org/wiki/Jewish_cuisine (accessed 3.12.16).
[6] Wikipedia, Islamic dietary laws https://en.wikipedia.org/wiki/Islamic_dietary_laws#Slaughter (accessed 3.12.16).
[7] L.M. Laita, E. Roanes-Lozano, L. de Ledesma, J.A. Alonso, A computer algebra approach to verification and deduction in many-valued knowledge systems, Soft Comput. 3 (1999) 7–19, doi:10.1007/s005000050086.
[8] E. Roanes-Lozano, L.M. Laita, E. Roanes-Macías, A polynomial model for multivalued logics with a touch of algebraic geometry and computer algebra, Math. Comput. Simul. 45 (1) (1998) 83–99, doi:10.1016/S0378-4754(97)00088-8.
[9] E. Roanes-Lozano, L.M. Laita, A. Hernando, E. Roanes-Macías, An algebraic approach to rule based expert systems, RACSAM. Rev. R. Acad. Cien. Serie A. Mat. 104 (1) (2011) 19–40, doi:10.5052/RACSAM.2010.04.
[10] J. Hsiang, Refutational theorem proving using term-rewriting systems, Artif. Intell. 25 (1985) 255–300.
[11] D. Kapur, P. Narendran, An equational approach to theorem proving in first-order predicate calculus, in: A.K. Joshi (Ed.), Proceedings of the International Joint Conference on Artificial Intelligence, IJCAI–85, Morgan Kaufmann, San Francisco, CA, 1985, pp. 1146–1153.
[12] J.A. Alonso, E. Briales, Lógicas polivalentes y bases de gröbner, in: M. Vide (Ed.), Proceedings of the Fifth Congress on Natural Languages and Formal Languages, Barcelona Univ. Press, Barcelona, 1989, pp. 307–315.
[13] J. Chazarain, A. Riscos, J.A. Alonso, E. Briales, Many-valued logic and Gröbner bases with applications to modal logic, J. Symb. Comput. 11 (1991) 181–194, doi:10.1016/S0747-7171(08)80043-0.
[14] L.M. Laita, E. Roanes-Lozano, V. Maojo, E. Roanes-Macías, L. de Ledesma, L. Laita, An expert system for managing medical appropriateness criteria based on computer algebra techniques, Comput. Math. Appl. 42 (12) (2001) 1505–1522, doi:10.1016/S0898-1221(01)00258-9.
[15] C. Pérez-Carretero, L.M. Laita, E. Roanes-Lozano, L. Lázaro, J. González-Cajal, L. Laita, A logic and computer algebra-based expert system for diagnosis of anorexia, Math. Comput. Simul. 58 (3) (2002) 183–202, doi:10.1016/S0378-4754(01)00370-6.
[16] C. Rodríguez-Solano, L.M. Laita, E. Roanes-Lozano, L. López-Corral, L. Laita, A computational system for diagnosis of depressive situations, Exp. Syst. Appl. 31 (2006) 47–55, doi:10.1016/j.eswa.2005.09.011.
[17] J. Piury, L.M. Laita, E. Roanes-Lozano, A. Hernando, F.J. Piury-Alonso, J.M.G. Argüelles, L. Laita, A Gröbner bases-based rule based expert system for fibromyalgia diagnosis, RACSAM. Rev. R. Acad. Cien. Serie A. Mat. 106 (2) (2012) 443–456, doi:10.1007/s13398-012-0064-8.
[18] E. Roanes-Lozano, J.L. Galán-García, G. Aguilera-Venegas, A portable knowledge-based system for car breakdown evaluation, Appl. Math. Comput. 267 (2015) 758–770, doi:10.1016/j.amc.2014.12.001.
[19] S.-M. Hong, J.-Y. Cho, J.-H. Lee, G. Kim, M.-C. Kim, Nutrisonic web expert system for meal management and nutrition counseling with nutrient time-series analysis, e-food exchange and easy data transition, Nutr. Res. Pract. 2 (2) (2008) 121–129, doi:10.4162/nrp.2008.2.2.121.
[20] T. Kashima, S. Matsumoto, H. Ishii, Decision support system for menu recommendations using rough sets, Int. J. Innov. Comput. Inf. Control 7 (5B) (2011) 2799–2808.
[21] M.-H. Park, J.-H. Hong, S.-B. Cho, Ubiquitous Intelligence and Computing, in: J. Indulska, et al. (Eds.), 4th International Conference, UIC 2007, Hong Kong, China, July 11-13, 2007, Proceedings, LNCS, 4611, Springer-Verlag, Berlin Heidelberg, 2007, pp. 1130–1139.
[22] R. Burke, Knowledge-based recommender systems, in: Encyclopedia of Library and Information Science, 69, 2000, pp. 180–200.
[23] C. Snae, B. M, FOODS: a food-oriented ontology-driven system, in: Proceedings of the 2008 Second IEEE International Conference on Digital Ecosystems and Technologies (IEEE DEST 2008), 2008, pp. 168–176. IEEE.

Please cite this article as: E. Roanes-Lozano et al., A prototype of a RBES for personalized menus generation, Applied Mathematics and Computation (2016), http://dx.doi.org/10.1016/j.amc.2016.12.023