Reasoning about Ingredient Substitutions with Ontologies and Rules

Weronika T. Adrian[1,*], Maciej Kutyła[2,†], Jędrzej Potoniec[2] and Agnieszka Ławrynowicz[2]

[1]*AGH University of Krakow*
[2]*Poznan University of Technology*

Abstract
Preparing healthy and delicious meals requires knowledge about the process, nutritional properties of the ingredients, and their combinations. When we face the need for substitution in a dish, e.g., for allergic or dietary reasons, we should know how removing one element and adding another influences the resulting experience. The paper proposes an ontology-based model for reasoning about ingredient substitution in food recipes, enhanced with a logic programming component for constraint satisfaction and optimization. This opens up possibilities for computer-aided decision support in the area of substitute recommendation.

Keywords
ontologies, logic programming, answer set programming, Internet of Food, computational food, knowledge representation

1. Introduction

Food is a common human experience. We eat to survive, but also to enjoy the experience, the flavors, and other sensations, such as the crispness or softness of the meal. Preparing courses and working with recipes requires knowledge about the process, combinations of ingredients, properties of single ingredients, and the resulting dish. This is often contained in the recipe, and any modifications may change the overall result of the process. However, sometimes we have to alter the original recipe to meet the goals or limitations of the eating person.

The emerging field of knowledge engineering [1] in computational food tackles the problems of knowledge acquisition, modeling, and processing in the domain of food. This knowledge is often implicit, contextual, and culture-dependent. Making at least parts of it explicit, with some sort of formalization, opens up possibilities to develop intelligent knowledge-based solutions to assist humans in preparing and optimizing food recipes.

Searching for substitutions in food recipes is motivated by various constraints and objectives of a person, including allergies, diets, etc. What ingredient to substitute with what and how

RuleML+RR'23: 17th International Rule Challenge and 7th Doctoral Consortium, September 18–20, 2023, Oslo, Norway
*Corresponding author.
†This research was carried out when Maciej Kutyła was employed for the project TAISTI at the Poznan University of Technology

wta@agh.edu.pl (W. T. Adrian); jedrzej.potoniec@cs.put.poznan.pl (J. Potoniec); alawrynowicz@cs.put.poznan.pl (A. Ławrynowicz)
0000-0002-1860-6989 (W. T. Adrian); 0000-0002-6115-6485 (J. Potoniec); 0000-0002-2442-345X (A. Ławrynowicz)

will it influence the resulting dish? These are just some of the questions that require the knowledge of a dietician, or a food technologist (and sometimes: both). While solutions based on computing co-occurrence scores between ingredients [2] or other black-box solutions [3] may produce proposals of ingredients that should be replaced, it is not always understandable, if (and why) certain ingredients are appropriate or not, and what are the properties of the proposed substitutes that satisfy the person's goals. Therefore, logic-based and hybrid solutions, such as employing similarities and reasoning in one system [4, 5], may be developed to provide transparent and explainable answers to the questions outlined above.

Within TAISTI research project[1], an interdisciplinary team, composed of dieticians, food technologists, and computer scientists has been working on modeling solutions for proper and healthy substitutions of recipe ingredients. In particular, we analyzed the existing ontologies and thesauri, in order to design a knowledge base and develop a logic-based method that would point out ingredients of a given recipe that do not satisfy the constraints defined by a user, and propose a method availing of logical reasoning to select candidate substitutes based on constraints. Such methods inherently can be a base for explainable substitute recommendations.

In this paper, we show how substitution can be modeled and reasoned about with ontologies and rules. We discuss the elements of knowledge useful for the substitution and propose a general reasoning framework for identifying unacceptable ingredients and pruning wrong substitute recommendations. The paper is organized as follows: in Section 2, we introduce selected problems that motivated our work; Section 3 introduces our knowledge-based approach to ingredient substitution. In Section 4 we show the general reasoning framework, and in Section 5, we instantiate it for particular cases. The paper is concluded in Section 6.

2. Motivating example cases

There are various reasons for substitution in recipes: allergies, diets, cultural requirements or constraints, etc. In this section, we focus on some of them. More scenarios can be found in [6].

2.1. Recognizing ingredients with gluten

Following [7], we assume that *gluten* is one of the proteins of wheat, barley, rye, and oats (and their hybrid) and thus to recognize that an ingredient contains gluten, it is sufficient to check whether it is a derivative of any of these four cereals, or any of its ingredient is such a derivative. Moreover, we consider idealized ingredients in that we do not take into account gluten impurities from the manufacturing process.

Assuming that recipe ingredients are represented as classes in a knowledge graph build upon the FoodOn ontology [8], in a complete knowledge graph it would be sufficient to check whether a class representing an ingredient is a subclass of 'derives from' SOME ('wheat plant' OR 'oat plant' OR 'barley plant' OR 'triticale plant' OR 'rye plant') (see FoodOn Ontology [8]).

For example, consider the recipe *Air fried sunfish*, with its list of ingredients, and their possible mappings to FoodOn reported in Tab. 1. Observe, that it immediately questions the assumption about the completeness of the knowledge graph. While no mapping is incorrect, some of them

[1]See http://taisti.eu

Table 1
Air fried sunfish, manually mapped to FoodOn classes

ingredient	FoodOn class	FoodOn label
8 sun fish fillets	FOODON:03309729	'filefish (raw)'
1 cup buttermilk	FOODON:00001067	'buttermilk food product'
2 teaspoons california garlic salt	FOODON:03307739	'garlic salt'
paprika	FOODON:03301223	'paprika (ground)'
black pepper	FOODON:03306739	'black pepper (ground)'
panko breadcrumbs	FOODON:03301907	'wheat bread crumbs'
2 sprays butter-flavored cooking spray	FOODON:03309370	'cooking spray (nonstick)'

are imprecise, in particular, fillets of a concrete fish and a concrete type of breadcrumbs were mapped to respective generic classes. Apart from that, the problem is relatively simple in this case: the only gluten-containing ingredient is panko breadcrumbs, and 'wheat bread crumbs' are a subclass of the class expression specified above. For the remaining ingredients, a close-world assumption must be made: since it does not follow from the ontology that they contain gluten, it must be assumed that they do not. We remark that it would be useful should the knowledge graph contain the necessary axioms to avoid making close world assumptions "on the fly".

The situation gets more complicated with *Healthy sesame chicken*. The ingredients and their manual mappings are presented in Table 2. Apart from the soy sauce, it is straightforward and

Table 2
Healthy sesame chicken, manually mapped to FoodOn classes

ingredient	FoodOn class	FoodOn label
boneless skinless chicken breasts	FOODON:00003364	'chicken breast (skinless, boneless)'
light soy sauce	FOODON:03301115	'soy sauce'
all-purpose flour	FOODON:03304534	'wheat flour (all purpose)'
sesame seeds	FOODON:03310306	'sesame seed (whole)'
black pepper	FOODON:03306739	'black pepper (ground)'
dry chili flakes		

can be treated similarly to the previous case. Soy sauce, on the other hand, is an interesting problem: traditionally, it is the product of fermentation of soy and wheat, and thus contains gluten. There are, however, variants of soy sauce that are produced without sources of gluten, such as tamari sauce, or some industrially-produced replacements of soy sauce. The knowledge graph should thus not assert that the soy sauce contains gluten, as only some variants do.[2] This, in turn, indicates that the user should be informed that they must choose a gluten-free variety of the sauce. This poses an interesting modelling challenge. One must also note that there may be two variants of gluten-free diet: one followed by people for whom even the traces of gluten cause health problems, and another followed by people who limit the intake of gluten, but the results of ingesting a small amount are insignificant to them. In the description above we

[2]For example, Kikkoman, one of the leading manufacturers of soy sauce, states in the FAQ https://web.archive.org/web/20220904110036/https://www.kikkoman.eu/faq/: *The gluten in Kikkoman Naturally Brewed Soy Sauce is below the detection limit of 10 ppm (according to tests by independent institutes). We recommend Tamari Gluten-free Soy Sauce for people with gluten intolerance.*

concentrate on the first variant, where the aim is to eliminate all the gluten sources. The second variant is far more subtle, as it requires finding a balance between the amount of gluten in the recipe, and the added cost of the modified recipe, due to, e.g., harder-to-obtain ingredients. For example, the recipe for *Easy Coconut Macaroons*[3] yields 14 servings and uses only 2 tablespoons of wheat flour. For some people, this amount may be acceptable and the recipe may be left as-is. For others, it requires adjustments.

2.2. Allergies and diets

A gluten-free diet is but one example of a wider family: allergies. While there are officially recognized lists of allergens, they vary from state to state in their comprehensiveness. For example, currently, the EU recognizes 14 major allergens, among others lupin and mustard [4]; at the same time in the US, only 9 allergens are recognized. However, it must be underlined that neither is nor aims to be comprehensive: their aim is to inform the majority of consumers with allergies, not all of them. This strongly hints that a system aiming at allergies should either allow the user to specify an arbitrary list of ingredients as a list of allergens or should follow specific regulations and target major allergies according to the law in selected countries.

3. Knowledge-based substitution in food recipes

Substitution *per se* has been mostly studied in recommender systems [2], mostly availing of the appearance of ingredients in similar contexts.

Conceptualizing the substitution, we have identified three aspects of it:

1. Goals – in fact, any substitution is motivated by the stating some objectives;
2. Constraints – a valid substitution is usually subject to meeting certain requirements;
3. Similarity – to avoid substituting *anything with anything*, there should be some similarity measure between the original and the resulting dish.

The similarity is an interesting challenge here. On the one hand, we do not want to change the original recipe *completely*, so we want the substitute to be somehow similar to the original ingredient, in terms of taste, crispiness, etc. On the other hand, the ingredient to be substituted has some features that *do not meet* the user's requirements or goals, so the new substitute must significantly *differ* from the original one. In our approach in TAISTI, we have solved this challenge by applying a two-phase process: first, we identify *candidate substitutes* with machine learning models, based on the similarity of their contexts (recipes in which they appear), and then we *prune* the substitutes with logic. The machine learning-based recommendation algorithms are beyond the scope of this paper; here we focus on the knowledge-based model for reasoning about the original recipes and the proposed substitutes.

[3]See https://www.yummly.com/recipe/Easy-Coconut-Macaroons-484241
[4]See https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32011R1169&from=EN

3.1. Knowledge model

Using the ontology design pattern for ingredient substitution proposed in [6], and after researching available ontologies and data sources, we have designed a knowledge model we describe next. Our proposed knowledge base consists of integrated parts of selected ontologies and is populated with instances that describe particular recipes (in particular, the required ingredient sets of the recipes). Based on this model, we implemented a (minimal viable) knowledge graph [5] on which we can demonstrate the reasoning about ingredients and substitutes that are not appropriate in a given context.

The knowledge we need for reasoning about substitution falls into the following three groups:

1. Constraint-related knowledge, in particular the knowledge about diets (from Ontology of Nutritional Studies (ONS) [9]) and allergies (based on EU regulations),
2. Food-related knowledge, mostly based on FoodOn [8],
3. Recipe-related knowledge, based on the ontologies proposed in [10] and [6].

To identify the "forbidden" ingredients in a considered recipe or to "prune" the list of proposed ingredient substitutes (which may be pre-computed with other methods, e.g. based on machine learning), one can check the inconsistency between the *user recipe* and the *user constraints*, both described formally, with the use of ontological reasoning or logical rules (in particular hard or weak constraints). A general sketch of knowledge portions used in the reasoning process and how they influence each other is depicted in Figure 1. For clarity, this picture omits the

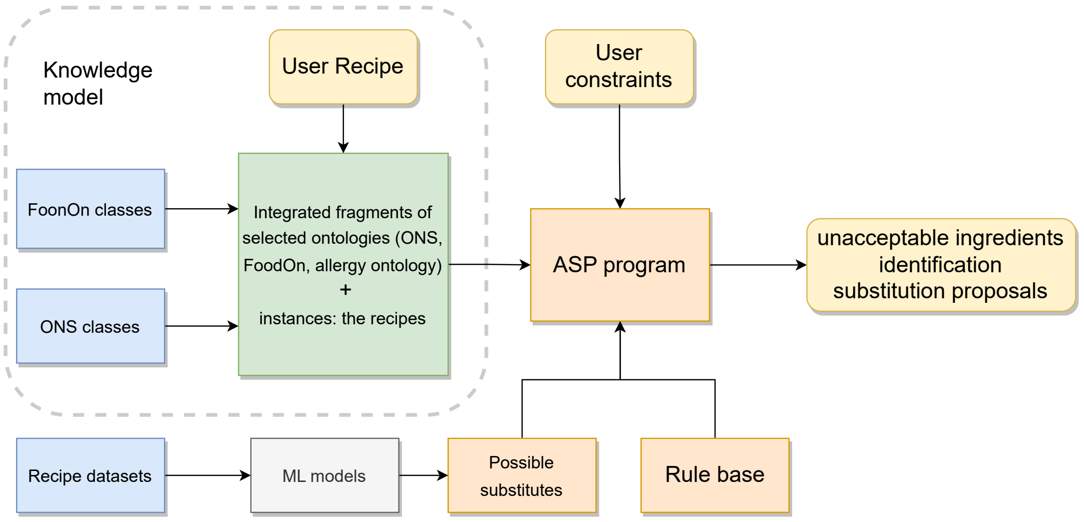
Knowledge model
User Recipe
User constraints
FoonOn classes
ONS classes
Integrated fragments of selected ontologies (ONS, FoodOn, allergy ontology) + instances: the recipes
ASP program
unacceptable ingredients identification substitution proposals
Recipe datasets
ML models
Possible substitutes
Rule base

Figure 1: Knowledge model of the proposed approach and the logic programming component

interconnections between food products, recipes, and constraint ontologies (the "integrated fragments of selected ontologies") that we discuss next.

[5]See https://github.com/taisti/SubstitutionRules

3.2. Main classes and properties in the integrated ontology model

Following good practices in ontology engineering, we have reused existing knowledge for the following subclass hierarchies:

- diets – diet classes have been imported from the Ontology of Nutritional Studies
- food products – wherever possible, food product classes have been imported from FoodOn
- recipe – following the recently added 'recipe' class from FoodOn, we imported the class and its related classes, such as ingredient set, etc.

For allergies, we defined a set of classes, based on the EU regulations, and re-use the property 'has allergic trigger' (RO:0001022) from the OBO Relations Ontology.

3.3. Modeling the recipes

For modeling the recipes, we have adopted the pattern and the class 'food recipe' first proposed in [10]. In this pattern, the 'food recipe' class is linked to the class 'ingredient set' through the 'has component' relation. The 'ingredient set' class is linked to the 'ingredient specification' class via 'has member' relation. An inverse relationship is 'is member of.' All classes of ingredients are linked to their corresponding classes in FoodOn (which are subclasses of the 'food product' class hierarchy) through the 'is about' relation. Each instance of the 'ingredient set' class is created using the schema: recipe name + 'ingredient set'. A similar naming convention is used for instances of 'ingredient specification', where the first part is the ingredient name.

3.4. Linking food to diets and allergies

The main classes that take part in reasoning are: 'recipe', 'food product', 'diet', and 'allergy'. Relations that we added to the above-mentioned ontologies are: 'acceptableIn' and 'unacceptableIn' (linking recipes with diets), and 'isNeededFor' (linking food products with recipes).

There are numerous diets that can be followed for a variety of reasons, including health and ideological ones. ONS includes a model that categorizes diets based on several characteristics, such as 'diet by nutritional composition'. The subclasses 'diet by kind of food' (ONS:1000003), 'diet by food organism' (ONS:1000019) were given special attention in this work. It comprises the classes 'vegetarian diet' (ONS:1000020) and 'vegan diet' (ONS:1000021). In addition, for the purposes of the model, the class 'meat diet' has been introduced. These straightforward examples will be used to demonstrate inference about whether a certain recipe is suitable for a specific diet. There are two relations linking a specific cuisine to diets:

- 'acceptable in' - connects instances from the food recipe class to diets;
- 'unacceptable in' - same domain and range as the 'acceptable in' property.

These relations are disjoint, i.e., we cannot connect a pair of instances using both of them. For example, 'chicken curry' cannot be both a meat and a non-meat dish. On the other hand, one dish can be linked several times to different diets, e.g., a vegan dish can also be explicitly linked to a vegan diet.

4. General reasoning framework

As outlined in Figure 1, the ontological knowledge model is a part of a broader reasoning framework. In this section, we present its elements, first discussing some ontological reasoning (see Section 4.1), and then the logic programming component (the ASP model, c.f., Section 4.2) whose primary responsibility is to identify the unacceptable ingredients in a recipe and prune inappropriate substitute recommendations.

4.1. Ontological queries and reasoning

In order to identify ingredients that do not meet constraints defined by the user, we can pose DL queries, within the ontology editor or via an API, or use ASP rules and constraints that can be interpreted by a logic programming engine. Given a recipe with its set of ingredients `recipe:Recipe` and constraints that can model a diet or an allergy etc., the following queries will identify the ingredient a that violates the required constraints:

1. For an instance `MY_DIET` of a class: `diet`:

   ```
   foodProduct(A) AND recipe(RECIPE) AND diet(MY_DIET)
   AND memberOf(A, RECIPE) AND unacceptableIn(A, MY_DIET)
   ```

2. For an instance `MY_ALLERGY` of a class `allergy`:

   ```
   foodProduct(A) AND recipe(RECIPE) AND allergy(MY_ALLERGY)
   AND memberOf(A, RECIPE) AND hasAllergicTrigger(MY_ALLERGY, A)
   ```

These high-level queries are based on the classification of the classes present in the ontology. New cases for allergies or diet-based restrictions can be added by providing appropriate logical rules or ontological axioms. For instance, one can express the diet constraints for a specific diet as follows:

```
unacceptableIn(A, vegan_diet) IF isSubclassOf(A, meat)
unacceptableIn(A, glutenfree_diet) IF derivesFrom(A, wheat)
...
```

and define specific food products to be allergy triggers:

```
hasAllergicTrigger(seafood_allergy, A) IF isSubclassOf(A, seafood)
hasAllergicTrigger(sesame_allergy, sesame)
...
```

They can be added to the ontology, as subclass axioms, or extend the logic programming rule base that we discuss in the next section.

4.2. Reasoning with logical rules and constraints

In this section, we provide a logic programming model that allows us to reason about substitution in a given context. The model is written in Answer Set Programming [11], a declarative paradigm for expressive knowledge representation and reasoning based on stable model semantics. Given a recipe (a unary predicate `recipe/1`) with a set of ingredients linked to the recipe with `isNeededFor/2` relation (binary predicate), and using the requirements given by a user (in terms of stating their `diet/1` or `allergy/1`), the model identifies which ingredient in the given `recipe/1` must or should be replaced, and provides a (set of) suggestions for substitution:

```
%%%% Rules for processing user information and suggesting substitutes
% R1: Identifying original ingredients that are wrong
must_replace(X) :- recipe(Recipe), allergy(Allergy), isNeededFor(X, Recipe),
hasAllergicTrigger(Allergy, X).

% R2: Identifying original ingredients that are wrong
should_replace(X) :- recipe(Recipe), diet(Diet), isNeededFor(X, Recipe),
unacceptableIn(X, Diet).

% R3, R4: Projection rules for cleaner output
suggestion(I,S):- must_replace(I), goodSubstitute(I,S).
suggestion(I,S):- should_replace(I), goodSubstitute(I,S).
```

Let us now discuss how the conclusions (`suggestion/2` predicates) are reached. As introduced in the previous section, user-defined constraints can be easily defined as rules that conclude that some ingredients are unacceptable in a diet or can trigger an allergy:

```
%% Rules for diets and allergies - an extendable rule base
% D1: vegan diet
unacceptableIn(X, vegan_diet) :- isSubclassOf(X, meat).
% D2: gluten-free diet
unacceptableIn(X, glutenfree_diet) :- derivesFrom(X,wheat).
%...
% A1: seafood allergy
hasAllergicTrigger(seafood_allergy, X) :- isSubclassOf(X, seafood).
% A2: sesame allergy
hasAllergicTrigger(sesame_allergy, sesame_seeds).
%...
```

To define candidate substitutes, we use a predicate `isSubstituteFor/2`. This information can be stated explicitly, as a fact, or inferred from the function a food product plays in a recipe:

```
% S1: inferring substitutes from functions
isSubstituteFor(A,B) :- foodProduct(A), foodProduct(B), hasFunction(A,F),
                        hasFunction(B,F).
```

Moreover, the substitute proposals can be obtained from the machine learning models (suggested as "candidate substitutes" with a certain level of similarity that we can asses):

```
% S2: inferring substitutes from ML-based recommendations
isSubstituteFor(A,B) :- foodProduct(A), foodProduct(B), ml_similar(A,B,S),
                        threshold(T), S>T.
```

To express that a certain ingredient (or a proposed substitute) must absolutely *not* appear in the recipe one can use *hard constraints* (or integrity constraints) expressed as: `:- conjunction of logical terms.` If an ingredient or a proposed substitute *should not* be allowed in the recipe, but in some cases can be accepted, one can define a *weak constraint* expressed in ASP as: `:~ logical terms.` (see [11]).

In the model, we follow the *Guess-Check-Optimize* methodology for ASP programming: We first state that a candidate substitute can be good or bad in the given context (to allow the reasoner to search for possible models), and then prune the bad substitute recommendations saying it is not a `goodSubstitute/2` if it violates the hard constraints of an allergy, and ask to minimize the `badSubstitute/2` proposals (suggest its avoidance because of dietary patterns):

```
%%%% Model to reason about substitution
%% I: Guess - In the given context, a substitute can be either good or bad
goodSubstitute(A,S) | badSubstitute(A,S) :- isSubstituteFor(A,S).

%% II: Check ( Constraints)
% C1: exclude substitutes that are allergic triggers
:- goodSubstitute(X,S), hasAllergicTrigger(Allergy, S).

%% III: Optimize (Weak constraints)
% W1: if possible, exclude substitutes unacceptable in the given diet(s)
:~ badSubstitute(X,S), unacceptableIn(S, Diet). [1]
```

Weak constraints can also be used to ensure minimizing or maximizing the intake of selected nutritional substances, e.g., in a low-fat or high-protein diet. This requires extending the ontology with data properties and working with numerical constraints which is currently our work-in-progress and is beyond the scope of this paper.

5. Use case examples

Let us consider selected possible questions we can pose to our model.

Case 1: A not vegan dish with gluten Assume we are looking for a dish with at least a single ingredient that contains gluten (implicitly - it is derived from wheat) and is not suitable for a vegan diet. This is possible with the following DL query:

```
'food recipe' and 'has component' some ('ingredient set' and 'has member'
some ('food product' and  'derives from' some 'wheat plant' )) and
'unacceptable in' some 'vegan diet'.
```

This query employs the ontology's recipe scheme: we check to see if there is any product in the 'ingredient set' that is linked to the 'wheat plant' class via the 'derives from' relationship. Finally, a second criterion is introduced that excludes all the recipes that are not vegan-friendly. As a result, we get the 'air fried sunfish recipe' and 'healthy sesame chicken recipe'.

Case 2: A meat dish with sesame allergen In this scenario, we seek a dish that has a sesame allergen and is part of a meat-based diet. This is achievable with the following DL query:

```
'food recipe' and 'acceptable in' some 'meat diet' and 'has allergic trigger'
some 'sesame allergy'.
```

This query is simpler - we have two restrictions that directly filter our collection of recipes. Only one recipe from our model meets these requirements: 'healthy sesame chicken recipe'.

Case 3: Searching for a gluten-containing product in specific recipe In the next example, we want to discover a gluten-containing product. Assume we know a recipe contains gluten (we can confirm this using the hasAllergicTrigger relation on the recipe), but we do not know which item contains it. We can verify this by running the following query:

```
'food product'  and 'is about' some('ingredient specification'
and ('member of' value 'air fried sunfish ingredient set'
and 'has allergic trigger' some 'gluten allergy'))
```

We find the 'air fried sunfish recipe' ingredients. Then, using the relation 'is about', we determine which food product the ingredient is associated with. 'wheat bread crumbs' is the result of this query and the problematic ingredient.

Case 4: Applying dietary and allergy-related constraints Let us now consider a more complicated case. A vegetarian person who allows some seafood occasionally but prefers not to eat it if possible and who is allergic to soy and sesame would like to prepare the 'Healthy sesame chicken' recipe from Case 2.

```
recipe(healthy_sesame_chicken). diet(avoid_seafood). diet(vegetarian_diet).
allergy(soy_allergy). allergy(sesame_allergy). diet(avoid_nuts).
foodProduct(chicken). isNeededFor(chicken, healthy_sesame_chicken).
foodProduct(soy_sauce). isNeededFor(soy_sauce, healthy_sesame_chicken).
foodProduct(flour). isNeededFor(flour, healthy_sesame_chicken).
foodProduct(sesame_seeds). isNeededFor(sesame_seeds, healthy_sesame_chicken).
foodProduct(black_pepper). isNeededFor(black_pepper, healthy_sesame_chicken).
foodProduct(chili_flakes). isNeededFor(chili_flakes, healthy_sesame_chicken).
isSubclassOf(chicken, meat). contains_soybeans(tofu). foodProduct(salmon).
isSubclassOf(salmon, seafood). isSubclassOf(squid, seafood).
```

Possible substitutes are listed below:

```
isSubstituteFor(chicken, seitan). isSubstituteFor(chicken, tofu).
isSubstituteFor(chicken, oyster_mushroom). isSubstituteFor(chicken, salmon).
isSubstituteFor(chicken, squid). isSubstituteFor(soy_sauce, coconut_aminos).
isSubstituteFor(sesame_seeds, panko_breadcrumbs).
isSubstituteFor(sesame_seeds,almonds). isSubstituteFor(sesame_seeds,corn_flakes).
```

If we extend the model to know that

```
% A3: soy allergy
hasAllergicTrigger(soy_allergy, soy_sauce).
hasAllergicTrigger(soy_allergy, X):- contains_soybeans(X).
% D3: avoid seafood diet (if possible)
unacceptableIn(X,avoid_seafood) :- isSubclassOf(X, seafood).
% D4: vegetarian_diet
unacceptableIn(X, vegetarian_diet) :- isSubclassOf(X, meat).
```

the model immediately gives us:

```
must_replace(soy_sauce).
must_replace(sesame_seeds).
should_replace(chicken).
```

Hard constraints filter out tofu (contains soybeans), while soft constraints filter out salmon, and squid, leaving them with oyster mushrooms and seitan:

```
suggestion(chicken,seitan). suggestion(chicken,oyster_mushroom).
suggestion(soy_sauce,coconut_aminos). suggestion(sesame_seeds,corn_flakes).
suggestion(sesame_seeds,panko_breadcrumbs). suggestion(sesame_seeds,almonds).
```

Since they are allergic to sesame, they sometimes react to other nuts as well (due to the allergens cross-reactivity [6]) and thus prefer to avoid them altogether if possible. To take it into consideration, one must add the following constraint:

```
% D5: avoid nuts
unacceptableIn(X, avoid_nuts) :- isSubclassOf(X, nuts).
```

and thus the model will filter out almonds preferring panko breadcrumbs or cornflakes.

6. Conclusion

In this paper, we have presented an approach to knowledge-based substitution of recipe ingredients that uses ontological modeling and logic programming. We have designed and implemented an ontology that integrates parts of existing knowledge bases to support the logic-based substitution of ingredients. We populated the model with several recipes. We have also designed a logic

[6]See, e.g., https://www.allergopharma.com/fileadmin/user_upload/allergopharma-com/Patients_information_download_Cross_reaktive_USA.pdf.

program in ASP that allows to reason about recipes and substitutions based on user-defined constraints regarding allergies and diets. Using description logic reasoners (e.g. HermiT[7]), we can reason about the classification of recipe ingredients, and thanks to an ASP engine we can identify wrong ingredients with hard and weak constraints, and reason about possible and appropriate substitutes.

In the future, we will consider more variants, and possibilities when it comes to ingredient substitution. From the user perspective, we will analyze a broader spectrum of cases and motivations for substitution. From the technical one, we plan to include constraints and optimization based on numerical values (e.g., to minimize or maximize intake of specific substances in the given diet). Finally, it is interesting to research how the tripartite flavor model [12] can be applied to look for substitutes similar to the original ingredients present in the recipe.

Acknowledgments

Research funded by the Applied Research Programme under the EEA and Norway Grants 2014-2021. Project registration number: NOR/SGS/TAISTI/0323/2020.

References

[1] A. Lawrynowicz, A knowledge engineering primer, CoRR abs/2305.17196 (2023). URL: https://doi.org/10.48550/arXiv.2305.17196. doi:10.48550/arXiv.2305.17196. arXiv:2305.17196.

[2] D. Lawo, L. Böhm, G. Stevens, Veganaizer: AI-assisted Ingredient Substitution (2020). URL: http://rgdoi.net/10.13140/RG.2.2.18736.17922. doi:10.13140/RG.2.2.18736.17922.

[3] S. Li, Y. Li, J. Ni, J. J. McAuley, SHARE: a system for hierarchical assistive recipe editing, in: Y. Goldberg, Z. Kozareva, Y. Zhang (Eds.), Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022, Association for Computational Linguistics, 2022, pp. 11077–11090. URL: https://aclanthology.org/2022.emnlp-main.761.

[4] K. Skjold, M. Øynes, K. Bach, A. Aamodt, IntelliMeal - Enhancing Creativity by Reusing Domain Knowledge in the Adaptation Process, in: ICCBR, 2017. URL: https://folk.idi.ntnu.no/kerstinb/paper/2017-ICCBR-Skjold-etal.pdf.

[5] S. S. Shirai, O. Seneviratne, M. E. Gordon, C.-H. Chen, D. L. McGuinness, Identifying Ingredient Substitutions Using a Knowledge Graph of Food, Frontiers in Artificial Intelligence 3 (2021) 111. URL: https://www.frontiersin.org/article/10.3389/frai.2020.621766. doi:10.3389/frai.2020.621766.

[6] A. Ławrynowicz, A. Wróblewska, W. T. Adrian, B. Kulczyński, A. Gramza-Michałowska, Food recipe ingredient substitution ontology design pattern, Sensors 22 (2022) 1095.

[7] J. R. Biesiekierski, What is gluten?, Journal of Gastroenterology and Hepatology 32 (2017) 78–81. URL: https://doi.org/10.1111/jgh.13703. doi:10.1111/jgh.13703.

[8] D. M. Dooley, E. J. Griffiths, G. S. Gosal, P. L. Buttigieg, R. Hoehndorf, M. C. Lange, L. M. Schriml, F. S. Brinkman, W. W. Hsiao, FoodOn: a harmonized food ontology to increase

[7]http://www.hermit-reasoner.com

global food traceability, quality control and data integration, npj Science of Food 2 (2018) 1–10.

[9] F. Vitali, R. Lombardo, D. Rivero, F. Mattivi, P. Franceschi, A. Bordoni, A. Trimigno, F. Capozzi, G. Felici, F. Taglino, et al., ONS: an ontology for a standardized description of interventions and observational studies in nutrition, Genes & nutrition 13 (2018) 1–9.

[10] D. Dooley, M. Weber, L. Ibanescu, M. Lange, L. Chan, L. Soldatova, C. Yang, R. Warren, C. Shimizu, H. K. McGinty, et al., Food process ontology requirements, in: IFOW 2021 integrated food ontology workshop, September 15-18, Bolzano Italy, 2021.

[11] T. Eiter, G. Ianni, T. Krennwallner, Answer set programming: A primer, Springer, 2009.

[12] T. Naravane, M. Lange, Tripartite flavour model: Food phenotype, sensory and interpretative matrices., in: ISWC (P&D/Industry/BlueSky), 2018.