Rule-Based Data Access: A Use Case in Agroecology

Elie Najm, Jean-François Baget and Marie-Laure Mugnier

LIRMM, Inria, University of Montpellier, CNRS, Montpellier, France

Abstract

There is a crucial need for tools to help designing sustainable agrosystems. In this paper, we consider the issue of selecting plant species according to the ecosystem services they are likely to deliver. For that, we rely on the one hand on recent scientific results in agronomy linking functional traits (i.e., measurable characterics of plant species) to ecosystem services, and on the other hand on data collected by the research community in ecology. The architecture of our prototype is inspired by the ontology-based data access paradigm, which clearly distinguishes between the data level and the knowledge representation level, with mappings linking the two levels. Knowledge is represented in a rule langage that extends plain Datalog with computed functions and stratified negation. We detail the construction of a knowledge base devoted to vine grassing, i.e., installing herbaceous service plants in vineyards, and briefly report on the experimental evaluation of the system's results on this use case.

Keywords

Agroecology, Vine grassing, Ontology-Based Data Access, Datalog

1. Introduction

Sustainable agrosystems should not only produce goods but also ecosystem services, like, e.g., pollinisation, nitrogen production for crops, soil fertility perservation, etc. It is widely acknowledged that this requirement involves increasing biodiversity on agricultural plots [1]. As these systems become much more complex, there is a crucial need for tools to help their design [2].

In this paper, we consider the issue of helping to select *service plants*, i.e., plants associated with crops, according to the ecosystem services they are likely to deliver. We propose to rely on two pillars. On the one hand, recent research in agronomy makes it possible to associate some measurable characteristics of plant species (called *functional traits*) with some *functions* of the agrosystem that contribute to the production of ecosystem *services*. For instance, several functional traits of the root system of a plant contribute to the function of soil structural stability, which supports the service of maintenance of soil quality [3]. On the other hand, rich data on functional traits has been collected by the international research community in ecology. In particular, the TRY initiative [4, 5] has built a very large dataset providing plant functional trait values measured in a wide range of environmental conditions (www.try-db.org). TRY currently integrates more than 400 datasets and contains experimental observations on 4 millions individual plants concerning 2100 different traits and about 160k plant taxa (mostly species). We hypothesized that if we could associate functional traits with functions and services, this

RuleML+RR'22: 16th International Rule Challenge and 6th Doctoral Consortium, September 26–28, 2022, Virtual

enajm@lirmm.fr (E. Najm); baget@lirmm.fr (J. Baget); mugnier@lirmm.fr (M. Mugnier)

database, and possibly others, would allow us to identify species that support these functions and services. To study the feasibility of our approach, we implemented it on the use case of *vine grassing*, i.e., installing service plants in vineyards [6].

Exploiting data on functional traits in the design of agrosystems is indeed a new approach. Existing decision helping tools rely on field experiments, farmers' know-how and workshops between agronomists from various domains. As this is time and budget demanding, these tools are typically restricted to a small set of plant species. Moreover, the decision is a "black box", in the sense that the computation of a recommendation is hardly explainable. On the other hand, these tools, which are intended for farmers and agricultural consultants, give a very accurate recommendation adapted to a specific cultivation context. As examples, let us cite SIMSERV [7], a tool for the selection of service plants (in a predefined list) to be associated with banana and yam crops, or a tool in agroforestry [8] to select shade tree species in coffee and cocoa agrosystems, based on an inventory of local practices. In contrast, our objective is to support the design activity of researchers and technicians in agroecology, with the aim of "opening the space of possibilities"; in particular, the tool should be able to suggest species that may not have been considered yet, while being able to explain why these species are likely to provide a desired package of services.

To sum up, our starting question was the following: can we exploit currently available data on plant functional traits and combine it with a suitable representation of scientific knowledge on the trait-function-service relationships, to assess the potential contribution of any plant species to some ecosystem service?

In this paper, we first present our system architecture (Section 2), the formal framework (Section 3) and the methodology to acquire expert knowledge from data sources (Section 4). Then, we detail the construction of a knowledge base devoted to the vine grassing case study (Section 5). Finally, we briefly report on the experimental evaluation of the system's results on this use case and discuss the lessons learnt.

2. System Architecture

To integrate data and knowledge in a principled manner, we decided to rely on the paradigm of *ontology-based data access* (OBDA) [9, 10]. OBDA systems are structured in three layers: the *conceptual level*, organized around a domain ontology ; the *data level*, composed of one or several data sources ; and *mappings* from the data level to the conceptual level, which allow to select relevant data and translate it into facts using the ontological vocabulary. Queries to the global system are expressed at the conceptual level.

The architecture of our system is outlined in Figure 1. A global *working database* is obtained by integrating several data sources. This integration step selects and aggregates relevant data, and translates it according to the global database schema, while keeping track of the data source provenance. The conceptual level is made of a *knowledge base* (KB), which comprises facts and rules. We further distinguish between two kinds of facts: *data facts* obtained from the working database (e.g., the fact that some functional trait for a given species has some normalized value according to a certain data source) and *expert facts* obtained from expert knowledge (e.g., the fact that a given ecosystem service is supported by some ecosystem functions, which

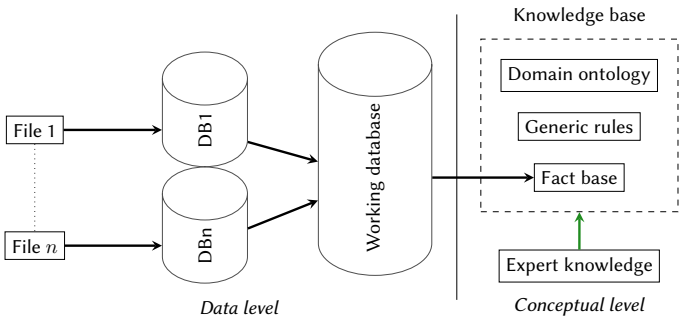

Figure 1: Overview of the global architecture. Black arrows depict mappings and the green arrow the formalization of expert knowledge.

.

themselves rely in some way on some functional traits). Expert knowledge is acquired under the form of diagrams, from which facts can be automatically built. About rules, we distinguish between those defining the *domain ontology*, which provides the concept and relations that are meaningful to a user (an expert in agroecology who builds diagrams or an end-user who queries the system), and more complex rules used to process data facts and combine them with expert facts to estimate the contribution of species to ecosystem functions and services. Note that the latter rules are *generic* in the sense that they are independent from a specific use case (e.g., vine grassing).

Mappings allow to select and aggregate information from a structure (here, a formatted text file, typically a csv file, or a database) and to translate the resulting information into the vocabulary of another structure (here, a database or a fact base). Importantly, they are specified in a declarative way.

Query answering in OBDA usually follows a mediating (aka virtualization) approach, i.e., the fact base remains virtual, and user queries are first reformulated with the ontology, then rewritten with the mappings, to yield queries that are directly evaluated on the data [9, 11, 12, 13]. In contrast, we follow here a materialization approach: the fact base is first materialized by triggering the mappings, then saturated by rule applications; we finally store the part of the saturated fact base that is relevant to an end-user as a relational database, in order to benefit from the whole expressive power of SQL. There are several reasons for the choice of materialization: first, mediation has been mainly developed for simple queries (essentially unions of conjunctive queries), while our user queries are more complex (e.g., may involve aggregations); second, some features of our KR language (computed functions, default negation) do not allow to use off-the-shelf reformulation techniques; third, most queries of interest require to rank species (e.g., find the k-best species for some service) and materialization is more appropriate to answer such queries efficiently. Finally, the main advantage of virtualization is the independence with respect to the evolution of data sources, yet this does not seem to be an issue in the target applications.

3. Formal Foundations

Regarding the KR language, we did not make any *a priori* choice. We started by eliciting expert knowledge to identify the language that allowed us to express domain knowledge in a convenient way, while having a restricted expressivity in order to avoid needlessly costly inferences. Rule-based formalisms were natural candidates since expert knowledge is often expressed under the form of rules. Furthermore, compared to description logics (DL) [14], rules allow to express complex relationships between entities, whereas DL are essentially restricted to tree-shaped descriptions and binary predicates. Another important feature is the ability to incorporate computed functions into the logical formalism (the term function is used here in the sense of programming, i.e., a function outputs a value given a list of parameters). Such functions allow in particular to aggregate values of traits or ecosytemic functions, and can be arbitrarily complex. Finally, default negation allows us, for instance, to process missing values or priorities. In the current state of the modeling, our rule language is an extension of plain Datalog to computed functions and stratified default negation [15], as formally defined next. Note that the rules do not involve disjunction in the head, as this feature was not required by the modelling.

3.1. The Rule Language

We consider finite sets of predicates and functional symbols of any arity. Beside standard predicates, there are predefined binary predicates like $=$, $\neq$ and $<$. Each functional symbol is linked to a function defined in a programming language. Constants may be *objects* or *literals*. A *term* may be simple or complex. A simple term is a variable or a constant. A complex term is of the form $f(t_1, \ldots, t_n)$, $n > 0$, where f is a functional symbol and each t_i is a term. An *atom* is of the form $p(t_1, \ldots, t_n)$, where p is a predicate of arity n and each t_i is a term. A *filter* is of the form *not* $p(t_1, \ldots, t_n)$ (negated atom) or t_1<op>t_2, where <op> is a predefined binary predicate and t_1, t_2 are variables or literals. Given an atom or filter A, or set of these, we denote by *terms*(A) and *vars*(A) the terms and variables, respectively, that occur in A.

A *fact* is an atom whose terms are constants. A *query body* is a conjunction of atoms on simple terms and filters, such that each variable occurring in a filter of the form t_1<op>t_2 also occurs in an atom. A *rule* R has the form $R = \forall \overrightarrow{X} (B[\overrightarrow{X}] \rightarrow H[\overrightarrow{X'}])$, where $B[\overrightarrow{X}]$ (the body of R) is a query body with *vars*$(B) = \overrightarrow{X}$; and $H[\overrightarrow{X'}]$ (the head of R) is an atom such that *vars*$(H) = \overrightarrow{X'} \subseteq$*vars*$(B)$. Note that H may contain complex terms. In the examples, we omit quantifiers, $\wedge$ is replaced by a comma, words starting with a capital letter are variables, and function symbols are prefixed by *fct:*.

Figure 2 illustrates facts and rules. The three first facts are data facts specifying a value for some trait and some species (e.g., the first fact says that the trait "specific root length" of species "dactylis glomerata" has value 0.72). Note that the values of traits are normalized and range on the interval $[0, 1]$. The next two facts are expert facts, specifying that "soil exploration and competition with vines" is an ecosystem function, which is linked to traits "specific root length", "root length density" and "relative growth rate", with "mean" as the method of aggregation of these trait values. The rule says that when an ecosystem function EcoSystemFunction is linked to traits Trait1, Trait2 and Trait3 with Aggregation as the aggregation method of these

```
% Facts
hasTraitValue("specific root length","dactylis glomerata",0.72).
hasTraitValue("root length density","dactylis glomerata", 0.38).
hasTraitValue("relative growth rate","dactylis glomerata", 0.54).
ecoSystemFunction("soil exploration and competition with vines").
isLinkedTo("soil exploration and competition with vines","specific root
length","root length density","relative growth rate",fct:mean).

% Rule
isLinkedTo(EcoSystFunction,Trait1,Trait2,Trait3,Aggregation),
hasTraitValue(Trait1,Species,V1),
hasTraitValue(Trait2,Species,V2),
hasTraitValue(Trait3,Species,V3)
→ hasValue(EcoSystFunction,Species,fct:aggreg3(Aggregation,V1,V2,V3)).
```

Figure 2: Five facts and a (positive) rule

trait values, and Trait1, Trait2 and Trait3 respectively have values V1, V2 and V3 for a species Species, then the score of Species for EcoSystemFunction is the aggregation of V1, V2 and V3 with method Aggregation. Here, *fct:mean* denotes a constant (5th fact), while *fct:aggreg3* is a functional symbol associated with a computed function whose first parameter is the name of the aggregation method. Note that this is a simplified example: the actual facts and rules have additional arguments to specify the data source and the plant growing conditions for a measured trait value, the weight of the link between a trait and an ecosystem function, as well as the reliability of the aggregated result.

A *knowledge base* (KB) $\mathcal{K} = (F, \mathcal{R})$ is composed of a finite set of facts F (the fact base) and a finite set of rules $\mathcal{R}$ (the rule set).

A *homomorphism* from a query body B to a set of facts F is a substitution h of *vars*(B) by *terms*(F) such that *(1)* for every atom $p(t_1, \ldots, t_k) \in B$, $h(p) = p(h(t_1), \ldots, h(t_k))$ is an atom of F, and *(2)* every filter in B is evaluated to true in the context of substitution h. In particular, a filter $\texttt{not}A$ is evaluated to true in the context of h if there is no homomorphism from $h(A)$ to F; note that when such a filter is evaluated some of its variables may not be instantiated by h. A rule $R : B \rightarrow H$ is *applicable* to a fact set F if there is a homomorphism h from B to F. The pair (B, h) is called a trigger for R on F. The application of a rule according to trigger (B, h) produces the atom $h(H)$, obtained by substituting each variable $X_i \in H$ by $h(X_i)$, then evaluating the complex terms; e.g., Figure 2: the application of the rule to the set of facts produces the fact `hasValue("soil exploration and competition with vines", "dactylis glomerata", 0.55)`.

Given a KB $\mathcal{K} = (F, \mathcal{R})$, a *derivation* (from F) is a sequence of fact sets $(F_0 = F), F_1, \ldots, F_n$ such that, for all $0 < i \leq n$, there is a rule $R : B \rightarrow H \in \mathcal{R}$ and a trigger (B, h) for R on F_{i-1}, $F_i = F_{i-1} \cup \{h(H)\}$ and $h(H) \not\subseteq F_{i-1}$. A derivation $(F_0 = F), \ldots, F_n$ is *complete* (aka fair) if it cannot be extended; then F_n is called the *saturation* of F by $\mathcal{R}$.

Finally, we consider stratified rule sets [15], which ensures that each KB has a well-defined semantics, based on its (unique) saturated fact base. A rule set $\mathcal{R}$ is stratifiable if there is a surjective mapping ρ from its set of intensional predicates P (i.e., predicates occurring in the rule heads) to a set of m integers, such that for all rule $R \in \mathcal{R}$ with head predicate p and all

$q \in P$ that occurs in the body of R: (1) if q occurs in an atom of R then $\rho(q) \leq \rho(p)$, and (2) if q occurs in a negated atom of R then $\rho(q) < \rho(p)$; then a *stratification* of $\mathcal{R}$ is a partition of $\mathcal{R}$ into subsets $\mathcal{R}_1 \ldots \mathcal{R}_m$ such that each rule with head predicate p belongs to the subset $\mathcal{R}_{\rho(p)}$. A derivation complies with a stratification $\mathcal{R}_1 \ldots \mathcal{R}_m$ if, for all $1 \leq i < j \leq m$, any application of a rule from $\mathcal{R}_i$ precedes any application of a rule from $\mathcal{R}_j$. It is well-known that, for a stratifiable rule set $\mathcal{R}$ and a KB $\mathcal{K} = (F, \mathcal{R})$, all the complete derivations that comply with a stratification of $\mathcal{R}$ lead to the same saturation. Semantically, stratified negation can be seen as a particular case of stable negation [16].

Answers to queries are defined with respect to the saturated fact base.

3.2. Mappings

Our architecture requires to manipulate data coming from different data sources, possibly stored in different systems (e.g., relational or NoSQL databases, spreadsheet tables, etc.). To be able to do that, we rely upon *mappings*: these objects are akin to rules and are composed of a query on some data source S_1 (the body, written in the query language associated with S_1) and an insertion query on some other data source S_2 (the head, written in the query language associated with S_2). Though the body of a query can be written in a fairly expressive language such as SQL, we also have to handle, for instance, the very simple queries on spreadsheet files that only return tuples corresponding to lines in a table. In that case, constraints can be added to the mapping body to ensure that no required value is missing, and more generally, that retrieved data satisfies some integrity conditions. Furthermore, functions in the mapping head allow transforming values from S_1 into values that fulfil the syntactic requirements of S_2 (e.g, type casting). Formally, a mapping from S_1 to S_2 has the following general form:

$$Q_1(\overrightarrow{X}), C_1(\overrightarrow{X_1}), \ldots, C_k(\overrightarrow{X_k}) \rightarrow Q_2(f_1(\overrightarrow{Y_1}), \ldots, f_p(\overrightarrow{Y_p}))$$

where Q_1 is a query over S_1 ; answers to Q_1 are substitutions of $\overrightarrow{X}$ by values from S_1; the C_i are constraints on $\overrightarrow{X_i} \subseteq \overrightarrow{X}$; and Q_2 is an insertion query over S_2, where each $\overrightarrow{Y_i} \subseteq \overrightarrow{X}$ and each f_i is a function symbol.

Such mappings can be seen as a generalization of the GAV relational mappings classically defined in OBDA to mappings over heterogeneous data sources, hence the addition of constraints to supplement the lack of expressivity of some query languages. The application of a mapping m is as follows: an answer to the body of m is a substitution σ that is an answer to Q_1 such that $C_i(\sigma(\overrightarrow{X_i}))$ evaluates to true for all $1 \leq i \leq k$; given an answer σ to the body of m, each $f_i(\sigma(\overrightarrow{Y_i}))$ $(1 \leq i \leq p)$ is evaluated, which yields a value v_i, and the tuple $(v_i)_{1 \leq i \leq p}$ is inserted into S_2 by the insertion query Q_2. Given a set of mappings from S_1 to S_2, the construction of S_2 is obtained by performing all the applications of these mappings on S_1.

In our current prototype, we rely upon data sources available in spreadsheet tables. *Cleaning mappings* select and transform data from each source to store it into a relational database. In those mappings, constraints are used to discard irrelevant, doubtful or unusable lines, and functions for normalization purposes. Then, *database mappings* from the obtained databases lead to a single relational database (the working database): those mappings are used to select

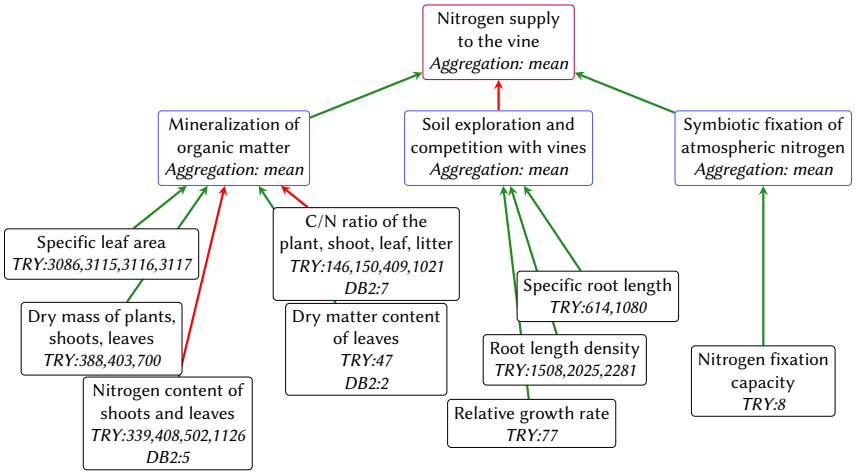

Figure 3: Traits-Functions-Service for Nitrogen supply to the vine. Green and red arrows indicate positive and negative impact, respectively.

data relevant to a specific use case (here, vine grassing) and to aggregate values. Then, *data-to-knowledge mappings* normalize and aggregate values from the working database to produce the higher-level facts (so-called data facts). Rules are applied to saturate the fact base, which also contains so-called expert facts. Finally, *storage mappings* from the saturated fact base to a relational database allow to select the part available for querying, while benefiting from SQL expressive power.

4. Acquisition of Expert Knowledge

In this section, we present the acquisition of expert knowledge by means of diagrams, as well as the construction of correspondences between traits identified by experts and database trait IDs.

Expert diagrams. The experts provide diagrams structured in 3 levels: *traits*, *functions* and *services*, with links between elements of successive levels. Figure 3 depicts the diagram describing the service *nitrogen supply to the vine*.[1] A green arrow denotes a positive impact of the trait (resp. function) value for the function (resp. service) rendered. A red arrow denotes a negative impact.

To define services relevant to agricultural ecosystems, the experts relied on the reference study EFESE[2]. Links from functional traits to ecosystem functions are based on scientific papers as well as grey literature. The experts were asked to specify *methods of aggregation* to pass from traits to functions (resp. from functions to services). Since no precise criteria could be derived from state-of-the-art domain knowledge, they decided to consider the mean of the normalized trait values, effectively giving the same importance to all traits (resp. functions).

[1]Diagrams for two other services can be found in file Appendix.pdf at https://gitlab.inria.fr/enajm/related-doc
[2]EFESE is a French national initiative to assess ecosystems and ecosystem services: https://www.inrae.fr/en/news/assessing-services-provided-agricultural-ecosystems-improve-their-management.

Correspondences with databases. The next step consists of associating each trait in a diagram with one or several trait IDs in available databases.

In its current state, the prototype is mainly based on the large Plant Trait database TRY presented in the introduction. It also uses a second database built by a French study on the interests of intermediate crops[3], which gives records for 58 species according to 7 traits. Although it is very small, this database has the advantage of being devoted to herbaceous species (relevant to vine grassing) and most trait values it provides for these species are not filled in TRY. It allows us to implement the principle of a multi-database setting. Next, this database is called DB2 as it is still confidential. Both databases use the same standard names for plant species, defined in *The Plant List*[4] [17] in relationship with the *Taxonomic Name Resolution Service*[5] [18].

Hence, each trait in a diagram is associated with one or several trait IDs in the databases TRY and DB2. As shown in Figure 3, some traits are associated with a single database trait ID, like "Relative growth rate" associated with ID 77 in TRY, or with a single ID in each database, like "Dry matter content of leaves" associated with ID 47 in TRY and ID 2 in DB2. However, most traits have more than one match in TRY. The reason for that is that traits can be measured according to different techniques, which are reported in TRY. Hence, the same trait for the same observed individual plant has different values according to the measurement techniques. Since we will normalize traits values (on a [0,1] interval), it will still be relevant to aggregate different values of a trait for a species, regardless of the measurement technique. Trait IDs associated with the same expert trait are called *exchangeable*. As there is no universally preferred measurement technique, they are very useful to mitigate the impact of missing values.

Finally, our modeling includes preferences between databases: there is a global default order, which can be overwritten for specific traits (here, TRY is globally preferred to DB2). This allows to give a higher priority to a more reliable source. The value of a trait for a species is given by the highest priority data source that provides one.

To summarize, our methodology for the acquisition of expert knowledge consists of the following main steps:

1. Build traits-functions-service diagrams, based on scientific sources and the grey literature.
2. Identify relevant databases and associate diagram's traits with relevant IDs in these databases, together with the choice of an aggregation technique.
3. Define priority among databases, globally and possibly for specific traits.

Back and forth between the steps 1 and 2 are necessary, as traits in the diagrams have to find counterparts in the databases.

Formalization. Expert knowledge is formalized into two types of knowledge:

- Generic rules handling the passage from database values to functional trait, function and service values for species. These rules are generic in the sense that they do not consider specific traits, functions or services, nor specific aggregation methods, hence they are not tied to specific diagrams, nor to specific data sources.
- Expert facts that describe specific diagrams, including their links to database trait IDs.

[3] https://methode-merci.fr
[4] http://www.theplantlist.org.
[5] https://tnrs.biendata.org/.

Importantly, rules are independent from a use case, hence the evolution of diagrams, including the introduction of new diagrams, only implies to generate again expert facts. Whereas rule-based expert systems often rely on carefully crafted rules that are tailored for a specific use case, our rules could in principle be applicable to any use case that follows the trait-function-service approach.

5. The Knowledge Base

The domain ontology provides the vocabulary meaningful to a user and is described by simple rules expressing concept subsumption and relation signatures. Additional predicates are used at intermediate steps in the computation of trait, function and service values.

5.1. From Expert Diagrams to (Expert) Facts

Each entity of a diagram is typed by the according concept (a unary predicate) and relationships between traits and functions (resp. functions and services) are captured by predicates that may have a high arity. E.g., the upper part of the diagram from Figure 3 (from the functions to the service) is described as follows:

```
biophysicalRegulationService("nitrogen supply to the vine").
function("mineralization of organic matter").
function("soil exploration and competition with vines").
function("symbiotic fixation of atmospheric nitrogen").
isLinkedTo3Functions("nitrogen supply to the vine",
   "mineralization of organic matter",1,
   "soil exploration and competition with vines",-1,
   "symbiotic fixation of atmospheric nitrogen",1, fct:mean).
```

The last fact, with predicate `isLinkedTo3Functions`, links the service to three functions, each followed by a number weighting their participation to the service (here, 1 for a positive participation, and -1 for a negative one), and specifies the aggregation method (here, mean). Other facts translate correspondences between traits and data IDs. E.g., the following facts express that: the trait "nitrogen content of shoots and leaves" has 4 matches in TRY (ID: 339, 408, 502 and 1126) and one match in DB2 (ID 5); furthermore, the aggregation of these exchangeable IDs is made by taking their *max* value (from the highest priority data source).

```
numberOfMatches("nitrogen content of shoots and leaves",4,try,max).
numberOfMatches("nitrogen content of shoots and leaves",1,db2,max).
hasTraitID("nitrogen content of shoots and leaves","339",try).
hasTraitID("nitrogen content of shoots and leaves","408",try).
hasTraitID("nitrogen content of shoots and leaves","502",try).
hasTraitID("nitrogen content of shoots and leaves","1126",try).
hasTraitID("nitrogen content of shoots and leaves","5",db2).
```

5.2. From Data to (Data) Facts

Using cleaning mappings. TRY data is available on request for specific traits (we asked for 52 traits, which yielded about 151k species) and comes as a formatted text file (similar to csv). This file contains so-called observations (about 1.6M); each observation (identified by an ID) corresponds to measurements of a trait (identified by an ID) on an individual plant (associated with a species or genus ID). An observation is described by several lines, which briefly provides measurements, as well as contextual information about the observation, original dataset provenance and estimation of the (un)reliability of the values (called error risk). Actually, the content of the cells is very heterogeneous in terms of units of measure, values taken by a non-numeric field, kind of contextual information, etc. To illustrate, the trait Plant Life Span takes string values among ["Bisannual", "Annual", "Biennial", "Perennial"] but one also finds at lot of other values like "perennial < 20 years", "biasannual", "pere", "nope", "from few decades to more than 60 years", "1", "2", "3", "winter annual", "shrub", "woody", etc. Hence, a step of cleaning is absolutely necessary before this rich source of information can be exploited in an automated way. This step, mainly based on string search, discards irrelevant, doubtful (cf. error risk) or unusable information, and transforms values for the retained fields. We also observed that the growing conditions of the observed plants may lead to very different trait values, hence we made the distinction between natural conditions and experimental conditions. At the end of this step, each triple (observation ID, species ID , trait ID, growing conditions) occurring in the obtained database has a single trait value expressed in a standardized unit (i.e., we chose a unit of measure for each trait). [6]

Although cleaning could have comprised the whole data and be independent of the specific vine grassing use case, we performed some selective cleaning for time reasons. In particular, only herbaceous species are relevant for the use case, while this is not a well defined category from an ecology viewpoint; we constructed this category from specific values of trait IDs, in order to distinguish species that are herbaceous from the others. Finer categories could be built for other use cases. The data source DB2 considers solely herbaceous with natural growing conditions and required no cleaning.

Using database mappings. The database mappings build the working database by retrieving only herbaceous species (about half of the species in TRY) and aggregating all the trait values coming from different observations for a given tuple (species ID, trait ID, growing conditions, source database). Hence, at the end of this step, the working database contains a single trait value for each tuple (species ID, trait ID, growing condition, source database).

Some additional queries compute views, in order to simplify the expression of data-to-knowledge mappings. In particular, for each trait ID, one computes its minimal and maximal values among all the values it takes in the working database with the same growing condition.

Using data-to-knowledge mappings. The computation of species' score for ecosystem functions and services, which is performed at the conceptual level, requires to aggregate values of different traits. To do so, we turn each value associated with a trait ID into a normalized

[6] Actually, the trait (ID) occurring in an observation has itself several associated "subtraits", whose values need to be aggregated, but for the sake of simplicity we do not detail this aspect in this paper.

```
(R1):  numberOfMatches(Trait,2,DB,Aggregation),
hasTraitID(Trait,TraitID1,DB), hasTraitID(Trait,TraitID2,DB),
TraitID1<TraitID2, hasSpeciesID(Species,SpeciesID,DB),
hasInitialValue(TraitID1,SpeciesID,GrowCond,V1,DB),
hasInitialValue(TraitID2,SpeciesID,GrowCond,V2,DB)
→ hasSingleDBValue(Trait,Species,GrowCond,fct:aggreg2(Aggregation,V1,V2),DB)

(R2):  numberOfMatches(Trait,2,BD,Aggregation),
hasTraitID(Trait,TraitID1,DB), hasTraitID(Trait,TraitID2,DB),
TraitID1!=TraitID2, hasSpeciesID(Species,SpeciesID,DB),
hasInitialValue(TraitID1,SpeciesID,GrowCond,V1,DB),
not hasInitialValue(TraitID2,SpeciesID,GrowCond,V2,DB)
→ hasSingleDBValue(Trait,Species,GrowCond,V1,DB)
```

Figure 4: Rules to aggregate trait values coming from a single database

value ranging over the interval $[0, 1]$. This is done by the formula $\frac{v - min}{max - min}$, where v is the value of the trait to be normalized for one species and min (resp. max) the minimum (resp. maximum) value of the trait for all (herbaceous) species in the working database.

Finally, the data-to-knowledge mappings yield facts of the following shape:

```
hasInitialValue(Trait ID,Species ID,GrowCond,NormalizedValue,DB).
hasSpeciesID(Species, Species ID, Database).
```

Note that the facts with predicate `hasInitialValue` still consider trait IDs and not the traits defined by the agronomists. It will be the role of rules to associate values to expert traits.

5.3. Rules to Consolidate Trait Values

As seen in Section 4, a trait (from an expert diagram) can be associated with several exchangeable trait IDs in each database. The aggregation of their values for a species S yields the trait value for S according to the considered database. This aggregation is meaningful as all trait values have been normalized in the interval $[0, 1]$ by the mappings. Its interest is that it yields more observations for a species, hence leads to retrieve more species. As an example, the trait "specific leaf area" is associated with four IDs in TRY, which are filled in for respectively 403, 7485, 6705 and 12584 species, which at the end yields 16006 species with a value for this trait.

To illustrate, Figure 4 depicts rules dealing with the case where a trait has two matching IDs in a database. For a given trait Trait that matches two IDs (TraitID1 and TraitID2), a species SpeciesID, a growing condition GrowCond, all from the same database DB, two cases are considered: either both database traits have values, and the computed value is the aggregation of these values, Rule (R1); or only one of the database traits has a value, which is then retained, Rule (R2). This step produces facts of the following form:

```
hasSingleDBValue(Trait,Species,GrowCond,NormalizedValue,DB).
```

Finally, we exploit the preference order on databases to retain the first available value (this is again expressed by rules) and compute facts of the following form:

```
hasValue(Trait,Species,GrowCond,NormalizedValue).
```

```
 % (R1): both traits filled, 100% reliability
isLinkedTo2Traits(Function,Trait1,Link1,Trait2,Link2,Aggregation),
hasTraitValue(Trait1,Species,V1,GrowCond),
hasTraitValue(Trait2,Species,V2,GrowCond)
→ hasCalculatedValueOfFunction(Function,Species,
      fct:aggreg2Links(Aggregation,V1,Link1,V2,Link2),100,GrowCond)

 % (R2): first trait filled, 50% reliability
isLinkedTo2Traits(Function,Trait1,Link1,Trait2,Link2,Aggregation),
hasTraitValue(Trait1,Species,V1,GrowCond),
not hasTraitValue(Trait2,Species,V2,GrowCond)
→ hasCalculatedValueOfFunction(Function,Species,V1,50,GrowCond)

 % (R3): second trait filled, 50% reliability: similar to (R2)
```

Figure 5: Rules to compute the value of an ecosystemic function

5.4. Rules to Compute Ecosystemic Function and Service Values

The aim of these rules is to express that the score of a species for an ecosystemic function is the aggregation of its values for all traits participating to the function. And similarly to go from ecosystemic functions to services. However, we face again here the problem of missing values: it often happen that not all the values of traits participating in a function are filled in for some species.

The natural way of doing, first recommended by the experts, was to discard species for which a required trait value was missing. However, this made almost all the species disappear when coming to ecosystemic services. Consider for instance the traits implied in the service "nitrogen supply to the vine": the percentage of species for which a given trait is filled in goes form $8,8\%$ of all the species (and $6,4\%$ of herbaceous species) to only 2 species. Note that herbaceous plants are not a particularly disadvantaged category in general.

Therefore, we decided to add a parameter indicating the *reliability* of the computed value of a function (respectively of a service). This allows one to return more species, which is in line with the objective of "opening the field of possibilities". The reliability parameter indicates the proportion of valued traits among all the traits associated with the function, i.e., $\frac{n_r}{n_t}$ where n_r is the number of traits with a value and n_t the total number of traits attached to the function.

To illustrate, Figure 5 depicts the rules to compute the value of an ecosystemic function linked to 2 traits: (R1) considers the case where both traits are filled (then the reliability of the result is 100%) and (R2-R3) the case where only one trait is filled (then the reliability is 50%).

5.5. Rules to Account for User Feedback

Finally, user feedback may palliate missing values for traits or invalidate the value assigned to an ecosystemic function (for a specific species). In the case of traits, the user is simply considered as a data source that has the highest priority, associated with trivial mappings. Facts coming from the user have the following shape: `hasSingleDBValue(Trait,Species,GrowCond,Value,User)`. Then, facts with shape `hasValue(Trait,Species,GrowCond,Value)` are produced as previously.

In the case of functions, facts coming from the user have the following shape: `hasUserValueOfFunction(Function,Species,Value,Reliability)`.
Rules that compute the value of a function give priority to facts with predicate `hasUserValueOfFunction` over those with `hasCalculatedValueOfFunction`.

6. Evaluation and Discussion

The use case described in this paper has been implemented in java using the Graal tool [19] (which is devoted to reasoning with existential rules), recently extended to stratified negation and computed functions/predicates[7]. Since mappings in Graal are still under development, we simulated the complete workflow by executing the corresponding queries.

6.1. Evaluation on the Vine Grassing Use Case

In order to assess the applicative validity of the approach, we carried out an evaluation[8] on a specific ecosystemic service for a specific set of species, and compared the scores provided on the one hand by the tool (tool score) and on the other hand by the domain literature (expert score).We chose the service of "nitrogen supply to the vine" (Figure 3), which is one of the most relevant for vine grassing. The experts selected a set of 19 herbaceous species, which were sufficiently well-documented in the literature, including scientific and grey literature. They retained 16 papers in the literature, each one providing a comparison between some of the selected species for the selected service. Note that each paper covers only some of the selected species (between 2 and 8 species).

From each document, the experts built a total order on the species covered in the document: $s_i > s_j$ means that s_i is deemed better than s_j for the service. To aggregate these different orders, we built a directed graph, whose nodes are the species s_i and there is an edge (s_i, s_j) if at least one document says that $s_i > s_j$. We found that the results in the literature were remarkably consistent, as the graph was circuit-free. Hence, the graph provided a partial order on the whole set of species. We then considered all the total orders compatible with this partial order, and assigned to each species a score equal to the average of its ranks in the total orders.

Finally, we studied the correlation between the tool and expert scores, according to Pearson correlation coefficient r. In our case, a perfect correlation is reflected by $r = -1$ (as the tool and the expert rank species in opposite orders), a perfect inverse correlation by $r = 1$, and an absence of correlation by $r = 0$. Globally, we found a good correlation ($r = -0.67$). Interestingly, the reliability of the tool's score appeared to be a crucial parameter: the value of r drops to -0.74 when we consider only species whose computed service value has a reliability at least 50%, and to -0.81 for reliability at least 60% (i.e., for these species, at least 6 of the 9 traits actually have a value by the predicate `hasTraitValue`). This highlights the crucial issue of missing values in the data sources.

[7] https://gitlab.inria.fr/rules/graal-v2
[8] We give here a brief outline of the evaluation, for more detail see the file Appendix.pdf at https://gitlab.inria.fr/enajm/related-doc

6.2. Concluding Remarks

In this paper, we studied the feasibility of a new approach for the selection of plant species in agriculture according to ecosystem services. This approach exploits on the one hand scientific results on the relationships between functional traits, ecosystem functions and services, and on the other hand data collected by the research community in ecology. Our modeling relies on generic rules, which allows for a smooth evolution of expert knowledge: expert facts can be automatically generated from the diagrams, without impact on the rules.

To put this approach in practice, we first had to undertake significant efforts to clean the data from TRY (which integrates almost all data sets about functional traits). This was a mandatory step before any automated exploitation. The evaluation carried out with agronomists confirms that very satisfactory results can be obtained as long as the proportion of missing values is not to high. As the TRY initiative is developing, in both volume and standardization of the data, and there is a growing effort to produce trait data in agronomy, the approach is indeed promising.

We expect the accuracy of the results to increase if contextual information (like climate and soil in particular) is taken into account. We did not exploit this kind of information because it would have further reduced available data and worsen the problem of missing values. As future work, we plan to also consider agricultural practices, as they are decisive for the realisation or neutralisation of a potential ecosystemic function. Finally, on a more practical side, our agenda includes a dedicated user interface, with explanation of query results in close relationship with expert diagrams.

Acknowledgments

We thank the experts in agronomy and agroecology who took part in this study: Christian Gary, Raphaël Métral, Léo Garcia and Aurélie Métay. We are also grateful to Sébastien Minette for providing us with the second traits database. This work was supported by a governmental grant managed by the Agence Nationale de la Recherche (ANR) within the framework of the "Investissements d'Avenir" program under the reference ANR-16-CONV- 0004 (#DigitAg).

References

[1] M. Duru, O. Therond, G. Martin, R. Martin-Clouaire, M.-A. Magne, E. Justes, E.-P. Journet, J.-N. Aubertot, S. Savary, J.-E. Bergez, J.-P. Sarthou, How to implement biodiversity-based agriculture to enhance ecosystem services: a review, Agronomy for Sustainable Development 35 (2015). doi:10.1007/s13593-015-0306-1.

[2] F. Lescourret, T. Dutoit, F. Rey, F. Côte, M. Hamelin, E. Lichtfouse, Agroecological engineering, Agron. Sustain. Dev. 35 (2015) 1191–1198. doi:10.1007/s13593-015-0335-9.

[3] L. Garcia, G. Damour, C. Gary, S. Follain, Y. Le Bissonnais, A. Metay, Trait-based approach for agroecology: contribution of service crop root traits to explain soil aggregate stability in vineyards, Plant and Soil 435 (2019). doi:10.1007/s11104-018-3874-4.

[4] J. Kattge, G. Bönisch, S. Díaz, S. Lavorel, I. Prentice, P. Leadley, S. Tautenhahn, G. Werner,

T. Aakala, M. Abedi, A. Acosta, et al., Try plant trait database – enhanced coverage and open access, Global Change Biology 26 (2020) 119–188. doi:10.1111/gcb.14904.

[5] J. Kattge, S. Diaz, S. Lavorel, I. C. Prentice, P. Leadley, G. Bönisch, E. Garnier, M. Westoby, P. B. Reich, I. J. Wright, et al., Try–a global database of plant traits, Global change biology 17 (2011) 2905–2935.

[6] L. Garcia, F. Celette, C. Gary, A. Ripoche, H. Valdés-Gómez, A. Metay, Management of service crops for the provision of ecosystem services in vineyards: A review, Agriculture, Ecosystems & Environment 251 (2018) 158–170. URL: https://www.sciencedirect.com/science/article/pii/S0167880917304309. doi:https://doi.org/10.1016/j.agee.2017.09.030.

[7] H. Ozier-Lafontaine, J.-M. Blazy, M. Publicol, C. Melfort, SIMSERV - Expert system of selection assistance of service plants, 2010. URL: https://hal.inrae.fr/hal-02820845.

[8] J. Van der Wolf, L. Jassogne, G. Gram, P. Vaast, Turning local knowledge on agroforestry into an online decision-support tool for tree selection in smallholders' farms, Experimental Agriculture 55 (2019) 50–66.

[9] A. Poggi, D. Lembo, D. Calvanese, G. De Giacomo, M. Lenzerini, R. Rosati, Linking data to ontologies, in: S. Spaccapietra (Ed.), Journal on Data Semantics X, Springer Berlin Heidelberg, Berlin, Heidelberg, 2008, pp. 133–173.

[10] G. Xiao, D. Calvanese, R. Kontchakov, D. Lembo, A. Poggi, R. Rosati, M. Zakharyaschev, Ontology-based data access: A survey, in: IJCAI, ijcai.org, 2018, pp. 5511–5519.

[11] D. Calvanese, B. Cogrel, E. G. Kalayci, S. Komla-Ebri, R. Kontchakov, D. Lanti, M. Rezk, M. Rodriguez-Muro, G. Xiao, Obda with the ontop framework., in: SEBD, Citeseer, 2015, pp. 296–303.

[12] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, A. Poggi, M. Rodriguez-Muro, R. Rosati, M. Ruzzi, D. F. Savo, The mastro system for ontology-based data access, Semantic Web 2 (2011) 43–53.

[13] M. Buron, F. Goasdoué, I. Manolescu, M. Mugnier, Ontology-based RDF integration of heterogeneous data, in: Proceedings of EDBT 2020, Copenhagen, Denmark, March 30 - April 02, 2020, 2020, pp. 299–310.

[14] F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, P. F. Patel-Schneider (Eds.), The Description Logic Handbook: Theory, Implementation, and Applications, Cambridge University Press, 2003.

[15] S. Abiteboul, R. Hull, V. Vianu, Foundations of Databases, Addison Wesley, 1994.

[16] V. Lifschitz, Answer set programming, Springer Heidelberg, 2019.

[17] J. M. Kalwij, Review of 'the plant list, a working list of all plant species', Journal of Vegetation Science 23 (2012) 998–1002.

[18] B. Boyle, N. Hopkins, Z. Lu, J. A. Raygoza Garay, D. Mozzherin, T. Rees, N. Matasci, M. L. Narro, W. H. Piel, S. J. Mckay, et al., The taxonomic name resolution service: an online tool for automated standardization of plant names, BMC bioinformatics 14 (2013) 1–15.

[19] J. Baget, M. Leclère, M. Mugnier, S. Rocher, C. Sipieter, Graal: A toolkit for query answering with existential rules, in: N. Bassiliades, G. Gottlob, F. Sadri, A. Paschke, D. Roman (Eds.), Proceedings of RuleML 2015, Berlin, Germany, 2015, volume 9202 of *Lecture Notes in Computer Science*, Springer, 2015, pp. 328–344.