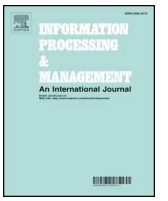
Information Processing and Management 57 (2020) 102214

Contents lists available at ScienceDirect

Information Processing and Management

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



Practical non-monotonic knowledge-base system for un-regimented domains: A Case-study in digital humanities

Luis A. Pineda[*,a], Noé Hernández[a], Iván Torres[b], Gibrán Fuentes[a], Nydia Pineda De Avila[c]

[a] *Instituto de Investigaciones en Matemáticas Aplicadas y en Sistemas, Universidad Nacional Autónoma de México, Ciudad Universitaria, Coyoacán, Ciudad de México, México*

[b] *Posgrado en Ciencia e Ingeniería de la Computación, Universidad Nacional Autónoma de México, Ciudad Universitaria, Coyoacán, Ciudad de México, México*

[c] *Posdoctoral fellow, Instituto de Investigaciones Estéticas, Universidad Nacional Autónoma de México, Ciudad Universitaria, Coyoacán, Ciudad de México, México*

ARTICLE INFO

Keywords:
Non-monotonic knowledge-base
Un-regimented knowledge domains
Information systems for un-regimented
Knowledge Domains
Digital humanities

ABSTRACT

Information systems for *un-regimented domains* such as museums, art and book collections, face representational and usability challenges that surpass the demands of traditional information systems for *regimented domains*. While the former require complex conceptual models supporting a set of dynamic and evolving qualitative properties of a small number of objects, the latter focus on the quantitative aspects of a possibly very large number of objects but with a relatively small and stable set of properties. In this paper we study the use of a non-monotonic knowledge-base system for the development of information systems for un-regimented domains. We discuss the ontological assumptions of the formalism, its structure and its inferential mechanisms through a simple example. Then we present an information system for a highly un-regimented domain in the digital humanities with promising results. The present study shows that the so-called extensible, flexible, dynamic or evolving information systems need the expressive power of non-monotonic knowledge-base systems, and that such phenomena should be addressed explicitly.

1. Information systems for un-regimented information domains

Regimented Information Domains admit standardization and can be modeled with the normal assumptions of spread-sheets and relational data-bases. These domains may include a huge set of objects but with a relatively small and stable number of mostly quantitative properties of a moderate number of types. These oppose to *Un-regimented Information Domains* that have a large number of types with an open ended set of mostly qualitative properties, although the number of individuals of each type may be very small, and resist standardization.

Un-regimented domains include objects that have a unique character, such as art objects, books in historical collections and museum pieces, to mention a few. These kinds of objects have unique properties that may be discovered by researchers or artists that work from different perspectives at particular places and times, and are subject to continuous research and curation. Through their work they may provide novel information at a given state of the system. At the same time, they might also change and enrich the

* Corresponding author.
E-mail addresses: lpineda@unam.mx (L.A. Pineda), nohernan@turing.iimas.unam.mx (N. Hernández), ivantr18@hotmail.com (I. Torres), gibranfp@unam.mx (G. Fuentes), nydia.pineda@gmail.com (N. Pineda De Avila).

https://doi.org/10.1016/j.ipm.2020.102214
Received 26 August 2019; Received in revised form 15 January 2020; Accepted 22 January 2020

conceptual model, which should remain coherent to other researchers and final users. Descriptions in the knowledge-base can also index multimodal information, and queries and updates can involve photographs of objects, sounds, annotated diagrams, etc.

The main objective of this research is to show that many problems with the conceptualization and development of complex information systems are due to the lack of an explicit distinction between regimented and un-regimented domains and, consequently, to the application of tools conceived for the former kind to the development of information systems of the latter.

In practice, when the domain is truly un-regimented, the system developing process is an ongoing activity plagued with new findings that involve modifying the conceptual model of the data-base, at great cost, time and effort. These kinds of systems are commonly labeled as *flexible, dynamic, evolving*, etc., but the problems are addressed mostly through management strategies employed during the system developing and testing processes.

The main claim of this proposal is that such problems can be addressed explicitly by recognizing first that the domain is un-regimented, and by using appropriate knowledge-base tools for the definition and developing effort. In particular, we claim that the issue of extending and refining the conceptual model is, ultimately, a problem of handling information that may stand in potential conflict with other pieces of information at the same time in a coherent and parsimonious fashion, which is the problem of non-monotonicity.

The proposal presupposes that information and knowledge-base systems can be seen as a continuum where standard information systems are just the basic case where the inferential capabilities are limited to direct inspection of the data-base or look-up operations, that are used under the unique-name and closed-world assumptions, as explained below. However, we do not suggest using declarative logical systems, either monotonic or non-monotonic, due to their usability, complexity and efficiency. The proposal is rather to use well-defined ontologies or proper hierarchies to the highest possible extent, and introduce practical heuristics to address the non-monotonic issues.

The second main objective of this research is to apply and extend a practical non-monotonic knowledge-base system, developed in our previous work on intelligent service robots (Pineda et al., 2018; Pineda, Rodríguez, Fuentes, Rascón, & Meza, 2017; Torres, Hernández, Rodríguez, Fuentes, & Pineda, 2019), to information systems for un-regimented domains. This service allows the specification of well-defined ontologies but does not enforce the unique-name assumption and supports the expression of incomplete information. Hence the system can be extended with new knowledge that may stand in conflict with previously stated knowledge, providing nevertheless a reliable answer any time. Furthermore, extending the system does not require redefining the conceptual model, which is never stated explicitly, and the distinction between system developer and final user is greatly reduced and even vanished, with the consequent economising of cost, time and effort.

The system supports the expression of strong negation, positive and negative defaults and exceptions, preferences or conditional defaults, that can be used from premises to conclusions, but also bidirectionally from conclusions to premises, yielding plausible explanations of the facts asserted in the knowledge-base. There is also a set of services for creating and updating all elements of the ontology dynamically, including utilities for class abstraction and class specialization, which are useful operations for stage changes along the development of the information system. The system is supplemented with declarative queries, built with the set of general knowledge-services, that permit to define, consult and update not only the content but also the structure of the representation. These queries are declarative Prolog programs similar to SQL queries for exploring the knowledge and can be defined by experts but also by general users on demand.

The third objective is to explore the feasibility of the approach through the development of an information system for a domain in the digital humanities that is clearly un-regimented. This is targeted to store, consult and refine information about astronomical images produced in the New Spain –today México– from the sixteenth to the eighteenth century. This enterprise involves registering images, the conceptual and actual authors, which may be astronomers and/or artists, the materials, techniques and tools used to produce them; etc. In addition, the system considers books, collections, libraries, curators, and particular properties and references that are relevant for the study of these images. This information can be collected and curated by individual researchers in libraries and museums, by several researchers working at different locations and even at different times, that may be involved with the same project. This information may as well be queried by a general user, such as visitors of libraries or museums, who may also input relevant observations about the objects and their relations. This exercise was carried out in a few weeks, and suggests that describing a knowledge-domain is an exploratory task involving an initial stage focused on creating and modifying the conceptual model or the structure of the knowledge base –by adding and modifying classes and individuals with their properties and relations– intertwined with queries to consult its current state, followed by a more stable phase oriented to final users, although the structure of the knowledge-base can be updated dynamically any time.

2. Expressive power and un-regimented information

Regimented information can be handled by representations with a limited expressive power. In the case of relational data-bases all objects must have unique identifiers to avoid ambiguities –the so-called *unique name assumption*– and the information expressed in the tables is assumed to be complete –the so-called *Closed-World Assumption* (CWA)) (Levesque & Brachman, 1985)). Conversely, if something is not expressed in the system it is assumed to be false. However, there are knowledge-domains where the CWA is a strong limitation, and interpretation should be valid as well under the assumption that there might be relevant information that is not included in the system, perhaps because it was not known when the data-base was created. In this case, systems are said to support or reason with incomplete information. Such assumption is more pressing in systems that have inferential capabilities in opposition to data-bases or spreadsheets that are limited to performing look-up operations upon tables.

The assumption that the information is incomplete requires the inclusion of the negation operator and the expression of positive

and negative atoms. This additional expressive power allows to distinguish when the answer is *no* because the relevant negative proposition is included in the knowledge-base (i.e., strong negation) or because such proposition is absent (i.e., weak negation). A system with this expressivity can answer *no* or *I don't know* respectively.

A further consideration is that the inclusion of negation allows the expression of contradictions and the knowledge-base may become inconsistent. This may arise when a general rule is stated and applied generally but counter-examples arise later. Modeling this kind of reasoning, which is known as *non-monotonic*, has a long history in Artificial Intelligence and Knowledge Representation that can be traced back to the so-called Truth-Maintenance Systems in the late seventies (Doyle, 1979) and to the default logics proposed by Ray Reiter in the early eighties of the last century (Reiter, 1980).

The expression of implications may allow other kinds of contradictions or incoherent propositions and a strategy to deal with conflicts of knowledge is to associate weights or priorities to propositions which here are called *preferences*. Implications can also be interpreted as having an abductive character relating an observation (i.e., the conclusion) with its plausible cause or explanation (i.e., the antecedent) and a mechanism to handle these "reverse" preferences should also be available. More generally, updating knowledge-bases including negation and material implication may produce contradictions and incoherence sentences. For these reasons, non-monotonic reasoning is one of the most important problems addressed in knowledge representation in AI.

The present overview shows that small increments of the expressive power of representational systems have strong effects on the structure of the knowledge-base and on the reasoning capabilities required to handle the information. Standard relational data-bases are the simplest type of representation scheme in which all or most of the information is expressed extensionally through concrete propositions and where the inferential capabilities that store and retrieve the information are direct look up operations. Hence, in such schemes –in addition to the limitations imposed by the unique name assumption and the CWA– all the propositions are expressed explicitly and there is no implicit information left to be rendered through inference.

These limitations can be addressed in part if the knowledge domain is monotonic using knowledge-base systems, such as the standard versions of the language OWL commonly used to express knowledge in the Semantic Web, that allow the expression of some propositions explicitly but also of implicit knowledge that is rendered through the reasoning or inferential capabilities, providing a better compromise between the explicit and the implicit knowledge, but at the cost of the additional inferential machinery.

Non-Monotonic systems stand at the opposite extreme of data-bases in this trade-off. In the former the expressed knowledge consists of the explicit propositions and the set of implicit propositions that can be rendered through inference, which is a much larger set; however, the non-monotonic machinery needed to make inferences is much more complex than in relational data-bases or monotonic inference systems. Indeed, an additional major problem in knowledge representation in AI is to achieve the best possible trade-off between the expressive power of the system and the cost of the inference (Levesque & Brachman, 1985) and, hence, the complexity of the inferential machinery.

3. Related work

Knowledge representation and reasoning (KRR) have traditionally been one of the main fields of study in Artificial Intelligence. Throughout more than 60 years of research, different logical formalisms have been exploited for this purpose, most commonly first-order logic and description logic (DL). Multiple languages and inference systems to represent and reason about knowledge have been derived from these formalisms, including KL-ONE (Brachman & Schmolze, 1985), LOGIN (Aït-Kaci, Nasr, & Seo, 1990), CycL (Guha & Lenat, 1991) and, more recently, the Web Ontology Language (OWL) (Bechhofer et al., 2004). KRR systems are typically evaluated in terms of expressiveness, consistency, completeness, efficiency, among other characteristics (Bingi, Khazanchi, & Yadav, 1995; Niwa, Sasaki, & Ihara, 1984).

Since the early days of the World Wide Web (WWW), a great deal of effort has been devoted to the development of systems that can handle ever increasing, constantly changing and highly complex knowledge from a wide variety of open domains (e.g. (Domingue, Motta, & Garcia, 1999; Motta, 1998)). Standard frameworks and systems have been established to facilitate knowledge exchange and reuse between different applications and domains, in particular, in the Semantic Web (Shadbolt, Berners-Lee, & Hall, 2006). These semantic technologies have become an essential part of the digital humanities.

One of the most important standards for the Semantic Web is the Resource Description Framework (RDF) (Brickley & Guha, 2014), which includes a data model for specifying resources in the Web. There are different ways to specify ontologies from RDF data such as RDF Schemas (RDFS) (Brickley & Guha, 2014), the Simple Knowledge Organization System (SKOS) (Miles & Bechhofer, 2009), the Rule Interchange Format (RIF) (Kifer & Boley, 2013) and the Web Ontology Language (OWL) (Bechhofer et al., 2004). Among these, OWL has been one of the most widely used, not only for WWW knowledge but also for other domains, such as robotics (e.g. (Becker et al., 2011; Fan, Tosello, Palmia, & Pagello, 2014; Pangercic, Tenorth, Jain, & Beetz, 2010; Tenorth, Kunze, Jain, & Beetz, 2010)). To the best of our knowledge these systems are monotonic since OWL's underlying formalism is Description Logic (DL), although non-monotonic OWL reasoners such as Pellet (Sirin, Parsia, Grau, Kalyanpur, & Katz, 2007) have been developed. Specifically, Pellet incorporates a non-monotonic epistemic operator to reason non-monotonically about knowledge expressed in OWL-DL (an OWL sublanguage). On the other hand, Klinov (2008) proposed a non-monotonic probabilistic DL reasoner built on top of Pellet to enable probabilistic reasoning within the Semantic Web.

Many of the KB systems mentioned above have been used as the building blocks of semantic technologies for different disciplines of the humanities, including library science, cultural heritage, history and philosophy. One early KB system for the digital humanities was BABEL, a library information system which uses LOGIN as its knowledge representation formalism. The humanities have proved to be particularly challenging domains because the knowledge can be highly context-dependent, frequently changing and subject to different interpretations. In particular, managing knowledge that can change constantly to reflect its evolution has recently gained

attention in the Knowledge Representation community. This trend has been partly driven by the limitations of current systems to properly handle un-regimented domains, such as those of the humanities.

Hitherto, the Web Ontology Language has been the most popular way to handle knowledge in different domains. Unfortunately, this system poses several limitations when knowledge evolves over time or needs to be changed constantly. To cope with these problems, Avery and Yearwood (2003) extended OWL with versioning mechanisms to account for evolving ontologies. Niepert, Buckner, and Allen (2007) studied the use of information retrieval and information extraction methods to maintain a dynamic ontology for reference work in the *Stanford Encyclopedia of Philosophy*. In some of the disciplines of the humanities, there have been huge efforts to incorporate the semantic technologies to their practice, which has resulted in prominent reference models, standards and frameworks such as the CIDOC Conceptual Reference Model (Aalberg et al., 2018; Bruseker, Carboni, & Guillem, 2017; Doerr, 2003; Görz, Günther, Schiemann, & Oischinger, 2008; Guillem & Bruseker, 2017) for cultural heritage and museum documentation, the Dublin Core (Weibel & Koch, 2000) for Library Science and InPho for Philosophy (Buckner, Niepert, & Allen, 2011).

In general, knowledge bases are designed, populated and maintained manually by humans. This process can be expensive and time-consuming, especially when dealing with large and complex knowledge resources. For this reason, a lot of effort has been devoted to automatically extract and/or infer knowledge taxonomies or ontologies from text documents. For example, the network, document and relation structure of Wikipedia articles has been exploited to extract knowledge and answer open questions using a set of matching rules (Ryu, Jang, & Kim, 2014). In the same application domain, Answer Set Programming was used to populate and infer such kinds of ontologies from text documents (Niepert, Buckner, & Allen, 2008). More recently, K and Thilagam (2019) proposed Crime Base, a system to automatically extract and integrate crime entities and their relationships from text and image information in online newspapers. A multi-source knowledge representation learning model was proposed by Tang, Chen, Cui, and Wei (2019), which considers entity descriptions, hierarchical types and textual relations to learn a knowledge graph.

In this work, we introduce a system to represent and reason about knowledge of un-regimented domains, which can take advantage of existing ontology learning approaches such as those mentioned above. The proposed KB system resembles description logic in that it maintains an explicit ontology while supporting conceptual inferences with clear regimented classes, individuals, properties and relations. However, as opposed to description logic, it also enables non-monotonic reasoning and can handle un-regimented domains by incorporating a simple and efficient computational mechanism based on the principle of specificity, as explained below. We believe this KB system offers a good trade-off between efficiency and expressiveness, allowing the expression of complete and incomplete knowledge, supporting queries and updates, and solving conflicts through preferences among properties and relations. Therefore, it offers an efficient non-monotonic alternative to widely used KB systems in the digital humanities, such as OWL and RDFS.

4. Non-monotonic taxonomies

The basic ontological assumption of the present knowledge-base system is that there is a set of individual objects constituting the universe or domain of discourse. The domain is divided into a set of mutually exclusive partitions. We define *a class* as a partition abstracting away their individual objects. Each partition can be further divided to conform subclasses down to the basic partitions representing the basic classes. This is illustrated in Fig. 1 where each individual object is represented by a bold dot and classes are demarcated by lines, whose thickness represents the level of class (i.e, the thickest the highest). The classes are labeled with their names, such that the size of the font denotes the class level too.

Properties and relations, in turn, are thought of as regions over the domain that can overlap across different classes –there may be an arbitrary number of regions for the same property or relation– such that individuals have the properties and relations represented by the regions in which they are included. Properties and relations can have a negative character and individuals in such regions do not have the corresponding properties or relations, and inclusion in a region labeled with a negative atom stands for strong negation. The extension of properties and relations is illustrated in Fig. 1 with the regions labeled p_i and r_i –which may be negative– respectively.

Common knowledge engineering tasks proceed in discrete stages that correspond to particular states of the ontology. At each stage, properties and relations of classes and individual objects can be added, updated or deleted, in a *horizontal dimension*. These

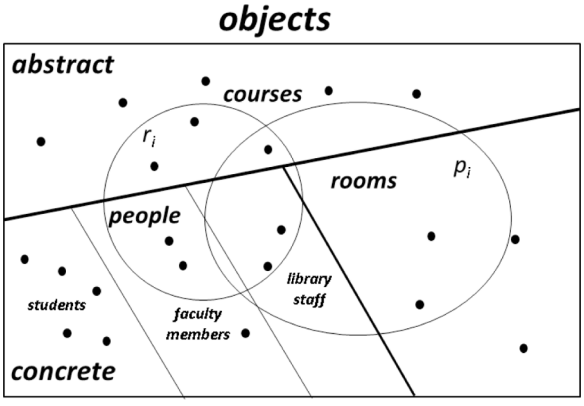


Fig. 1. The Ontology.

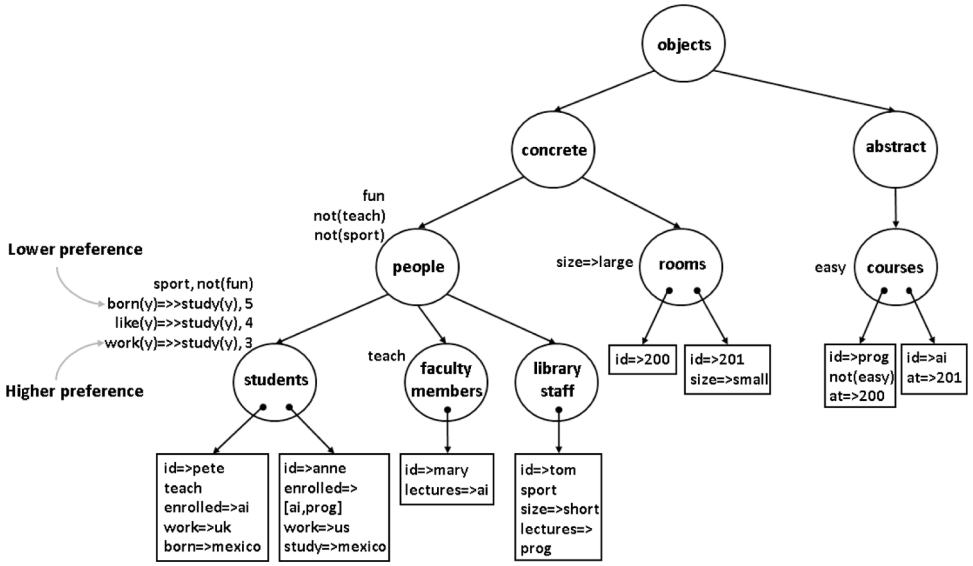


Fig. 2. Non-Monotonic Ontology.

stages correspond to particular conceptual models of data-base systems which support queries and updates in stable states. If the ontology is thought of as the belief state of the system, these kind of operations implement the belief revisions of the knowledge base.

However, the development of an ontology involves also operations that modify the class structure in a *vertical dimension*. Two main operations of this latter kind are class abstraction and class specialization. In the former, two or more classes are coalesced or combined in a more general one, and in the latter one class is split off into two or more specific or concrete classes. These kinds of operations change the conceptual stage of the system, correspond to modifications of the conceptual model of data-base, and cannot be considered as standard belief revision operations.

The graphical notation illustrates that the operation of class abstraction can be thought of as deleting the line demarcating two sub-classes and the operation of class specialization as drawing a line splitting a class into to mutually excluding regions. These operations do not affect the individuals in the ontology and only provide respectively a more general and a more specific view of the represented world.

The availability of these latter operations makes the knowledge-engineering process more transparent, reduces the gap between developers and final users, provides a more parsimonious framework, and reduces the cost, time and effort of the development and maintenance effort.

4.1. Notation

The recurrent division of the universe in mutually exclusive partitions corresponds to a strict hierarchy or taxonomy. This is illustrated for the current example in Fig. 2. Classes are represented by circles and individuals by boxes. The class subsumption or inclusion relation is represented by arrows top-down from the subsuming to the subsumed class. The membership relation is represented by a bold dot within the circle representing its class and an arrow pointing to the box representing the individual member. The labels naming classes and individuals are placed within the corresponding circles and boxes. Labels representing properties and relations of classes are placed next to the corresponding figures, and those of individuals are placed within the corresponding box.

Properties and relations of classes hold for all individuals of the class and are named with positive or negative atoms (e.g., `fun`, `not(fun)`). The reading is indeterminate and the answers to *do people have fun* or *do teachers teach?* should be *yes* without specifying who. Properties and relations of individuals, on the other hand, have a specific reading. The labels in boxes and next to circles are bound within the scope of their corresponding objects, that is, the individuals having such properties or the subjects of such relations. Boolean properties are represented directly by atoms; otherwise the operator `=>` relates the property with its value (e.g., `size=>short`).

The operator `=>` is also used to represent a relation of the bound subject with its corresponding object. For instance, `lectures=>prog` in the box with id `tom` states that Tom lectures programming. The operator `=>>` in turn stands for an implication such that the consequent holds only if the antecedent holds; the antecedent can have a sequence of positive or negative atoms, interpreted as the corresponding conjunction, but the consequent is a single positive or negative atom. Implications are also bound to the individual that has them in its box or to the class that they are placed next to, and the argument variable ranges over the individuals that can stand in such relation. For instance, `work(y)=>>study(y)` next to `students` states that if a student works in `y`, then she studies in `y`. The integers next to the implications stipulate a weight or priority of such proposition, as will be explained below. The information stated through all these labels and operators is the explicit information provided by the human user who plays the role of both the developer and the final user.

A knowledge-base consists of a list of Prolog's predicates each representing a class. The arguments are the name of the class, the mother class, and the lists of properties, relations and individuals of the class. The three lists may be empty. Individuals are in turn specified as a list including the individual's id, the list of its properties and the list of its relations. Properties and relations are also specified as a list of two elements: the property or relation proper and its associated weight, which for standard properties and relations is zero. The conditional expressing preferences can be included both in the list of properties and in that of relations, suggesting that the consequent is a property or relation that holds in case the antecedent holds. This is a mere notational convention and other alternatives could be defined as well. The actual specification of the knowledge expressed in Fig. 2 is shown in Listing 1.

```
[
class(top,none,[],[],[]),
class(objects,top,[],[],[]),
class(concrete,objects,[],[],[]),
class(abstract,objects,[],[],[]),
class(people,concrete,[[fun,0],[not(teach),0],[not(sport),0]],[],[]),
class(rooms,concrete,[[size=>large,0]],[],
 [
  [id=>200,[],[]],
  [id=>201,[[size=>small,0]],[]]
 ]),
class(courses,abstract,[[easy,0]],[],
 [
  [id=>prog,[[not(easy),0]],[[at=>200,0]]],
  [id=>ai,[],[[at=>201,0]]]
 ]),
class(students,people,
     [[sport,0],[not(fun),0],
      [(born=>'-')=>>(study=>'-'),5],
      [(like=>'-')=>>(study=>'-'),4],
      [(work=>'-')=>>(study=>'-'),3]],
     [],
 [
  [id=>pete,[[teach,0],[work=>uk,0],[born=>mexico,0]],
    [[enrolled=>ai,0]]],
  [id=>anne,[[work=>us,0],[study=>mexico,0]],
    [[enrolled=>[ai,prog],0]]]
 ]),
class('faculty members',people,[[teach,0]],[],
 [
  [id=>mary,[],[[lectures=>ai,0]]]
 ]),
class('library staff',people,[],[],
 [
  [id=>tom,[[sport,0],[size=>short,0]],[[lectures=>prog,0]]]
 ])
]
```

Listing 1. Full code of the example non-monotonic taxonomy.

4.2. Inference

Individuals have, in addition to their specific properties and relations, the properties and relations of the classes that they belong to, and subsumed classes have the properties and relations of their subsuming classes. These can be seen directly in the diagrammatic representation in Fig. 1 where the set membership and inclusion relations correspond to the basic *modus ponens* inference scheme. For instance, in Fig. 2, library staff have fun, do not teach and do not practice sports, and Tom, a member of the staff, inherited such properties in addition to his own. Properties and relations of classes can be thought of as defaults that hold for all individuals of the class and its subsumed classes.

However, negation allows the specification of direct contradictions. For instance, students do and do not have fun, and Pete, a student, does and does not teach and practice sports. In addition, there are also incoherent specifications such as the size of rooms, which is large by default, but room 201 is small, and such room is large and small. The example shows several instances where there is a conflict between positive defaults and negative exceptions and vice versa.

An arbitrary set of consistent propositions can be extended in one way with one proposition, say the default, and in another with the conflicting proposition, say the exception. The two paths are mutually inconsistent but each is a consistent extension of the original set and the knowledge-base has *multiple extensions*. The reasoning problem is then reduced to choose the consistent extension that is relevant for the current query.

To handle this problem we use the principle of specificity stating that in case of knowledge conflicts the more specific propositions should be preferred. This principle is already a general interpretation preference that provides the preferred consistent extension over the whole of the knowledge-base. Defaults and exceptions are handled with this simple principle as the exception will be preferred always. Exceptions can even have exceptions so the default may nevertheless hold, and this is handled by the same principle.

The interpreter assumes that knowledge may be incomplete; specific rules are included for handling negation and the system can answer *no* in case the proposition is denied explicitly or through inference, or *I don't know* if neither the positive nor the negative atom is included in the closure of the inheritance relation (Pineda et al., 2017).

Implications hold also for all individuals of the class at which these are specified and hence for all individuals in the subsumed classes. Their conclusions can be thought of as conditional defaults that hold in case their antecedents hold. For instance, if a student works at *y* then he studies at *y*; as Pete works at UK then he studies in UK. However, it is also stated that if someone was born at *y* then he also studies at *y*; this is the case for Pete, who was born in Mexico and then he studies in Mexico. Finally, among the conditionals for students it is known that if a student likes place *y* then he studies at *y*, but it does not apply to Pete since the place he likes is not provided. The conditional defaults are specified at the same level and inherited all the way down to the individual entity; and the conflicting facts, Pete studying both in the UK and Mexico, cannot be dealt with specificity alone. For this an additional preference or weight is assigned to the conditional. Assuming the convention that the lower the weight the higher the priority, in the present example the preferred conditional is that students study where they work. Consequently, the system will answer that Pete studies in the UK.

Implications can be seen as abductive or diagnosis rules too. For instance, by taking from the taxonomy that Anne studies in Mexico and considering the conditional rules for students a hypothesis can be drawn that she does so because she works, likes or was born in Mexico. She works in the US, so the conditional with the highest priority `work(y)=>>study(y)` is dismissed. Next in priority is `like(y)=>>study(y)`, by instantiating such implication with `y=Mexico` no contradiction occurs, therefore the system will answer that Anne studies in Mexico because she likes it, ignoring thus the conditional in third place `born(y)=>>study(y)`.

For the implementation, the hierarchy can be examined either top-down or bottom up placing all atoms in a list, such that the most specific information is placed at the front. Similarly, the propositions with the highest priority or lowest weight are also placed at the front, and the resulting list contains the preferred propositions overall.

The extension of the knowledge base consists of the sets of classes and individuals with their properties and relations expressed explicitly (e. g., Listing 1), in addition to the implicit knowledge that can be derived through inference, which corresponds to the closure of the inheritance relation. The knowledge-base system provides eight basic services for retrieving such knowledge (Pineda et al., 2017), as follows:[1]

1. `class_extension(Class, Extension)`: provides the set of individuals in the argument class.
2. `property_extension(Property, Extension)`: provides the set of individuals that have the argument property.
3. `relation_extension(Relation, Extension)`: provides the set of individuals that stand as subjects in the argument relation.
4. `explanation_extension(Property/Relation, Extension)`: provides the set of individuals with an explanation supporting why such individuals have the argument property/relation.
5. `classes_of_individual(Argument, Extension)`: provides the set of mother classes of the argument individual.
6. `properties_of_individual(Argument,Extension)`: provides the set of properties of the argument individual.
7. `relations_of_individual(Argument, Extension)`: provides the set of relations in which the argument individual stands as subject.
8. `explanation_of_individual(Argument, Extension)`: provides the supporting explanations of the conditional properties and relations that the individual has.

[1] These eight basic services also need the KB of interest to be passed as an argument. An illustration of the use of each of these predicates in relation to the KB specified in Fig. 2 and Listing 1 is presented in Appendix A.

These KB-services are used as normal Prolog predicates within specialized queries to retrieve specific information, in a manner similar to standard SQL queries, but expressed in Prolog's notation. Thus the KB system allows the definition of tailored and custom queries that meet the requirements encountered during any particular management of a knowledge resource.

The KB system has also a set of utilities to add, update or remove properties and relations of classes and individuals. The eight basic services, the specialized procedures and the basic maintenance services work in the horizontal dimension at any given stage of the ontology, and can be thought of as the utilities used for belief revision.

In addition, the system supports services to add and delete classes to create a new stage of the ontology in the vertical dimension. These operations add or remove a class, with their general properties and relations, but the individual objects involved are preserved, as these are the basic objects of the ontology.

The class abstraction operation simply creates a new class subsuming the classes involved, and collects the common properties of the subsumed classes. The class specialization, in turn, partitions a class into two or more specific classes, keeps the common properties, and inherits the corresponding individuals to their respective sub-classes. Adding a new class is the result of applying either the class abstraction or the class specialization operation, and deleting a class simply includes its individuals into the more specific subsuming class. These operations are performed by standard system utilities although its effect is to produce a new stage of the KB.

An example of an interactive session with the system, assuming the specification of the KB at a particular stage in Listing 1 is shown in Appendix A. There we show a number of queries with their corresponding answers in the horizontal dimensions, as well as a number of services for updating the content and structure of the KB in the vertical dimension.[2]

5. A case-study in digital humanities

5.1. Introduction

This case-study focuses on broadly-defined astronomical images in New Spain from the sixteenth to the eighteenth century. These are graphic representations (both manuscript and printed) related to recorded observations of celestial phenomena, diagrams of cosmological hypothesis, instruments, astrological figures, allegorical scenes, etc., which remain hitherto neglected in the historiography of the cultural history of science and art. Through a wide survey of instances in international repositories, the aim of this study is to assist a better understanding of how astronomical images were produced, circulated, and used by different social actors, as well as their place in contemporary repositories. Since no comprehensive survey of these representations has yet been conducted, the information that enriches this study increases according to unexpected findings of primary materials which, in turn, triggers processes of re-conceptualization. This research also relies on contingent conditions of access to archives. For this reason, the information gathered needs to be set in a dynamic and evolving framework that allows for radical change. For instance, as the research advances in the study of the image in Fig. 3, the image itself as well as its authors and readers could be related to instruments, practices, and other productions. The KB needs to be able to adapt to changing perspectives and questions.

In this study, the properties of objects (such as images, printed books, manuscripts, instruments), people and places, and the relations between them cannot easily be expressed in a closed-world system with fixed categories. Issues of authorship, for instance, can be ambiguous: the person who signs a certain engraving may be the same person that physically crafted the work, but an engraver is rarely the sole author of an image. In some cases, the printer of the image is also its intellectual author, but this cannot be taken for a rule. Moreover, this research defies fixed taxonomies because the images themselves were produced in contexts of great fluidity. It would be anachronistic, for instance, to say that an instance was either mathematical, astronomical or astrological, since these disciplines were not understood as separated fields of knowledge. Furthermore, copies of the same image in a variety of media and formats were circulated in a wide array of places and were thus read by different people. Readers of images, identified through provenance marks, might also be related to authors, printers and collectors, and can be linked by connecting instances found in a variety of repositories. The KB therefore needs to be able to account for all this entangled knowledge and to be open to new relations and findings.

In its current state, the KB considers images, books, people, places and repositories as objects with unique properties that can be inherited down to individual instances. Images, as a class, are described by name, dimension, support, technique, colouring, inscriptions, author and engraver, place of publication and time of publication. The description also includes whether the images are embedded or not in text and if so, in which language (considering that the most frequent language is Latin). This general framework then gives space to identify the characteristics of specific copies in different repositories, and to include information, such as specific inscriptions or provenance marks, without restricting the inclusion of a specific category. This information can shed light on the context of production of an image and allows for the establishment of relations to other relevant objects (i.e., instruments) and to people (i.e., printers, authors, artists, patrons). It is relevant that the KB considers books as a separate class from images because engravings can be found in books but can also circulate independently: the description of a book may or may not give information about a specific instance and therefore this KB avoids hierarchical relationships between books and images.

Moreover, people, as a class, are fundamental aspects of this KB, for they refer to authors, artists, publishers, readers and

[2] The new state of the KB after the example session is shown in the GitHub repository found in https://github.com/KBS-Lab-IIMAS-UNAM/non-monotonic-KBS-for-DH. Such a repository holds all the source code and examples of this paper and will be referenced often in Section 5, where an actual case-study in digital humanities is presented.

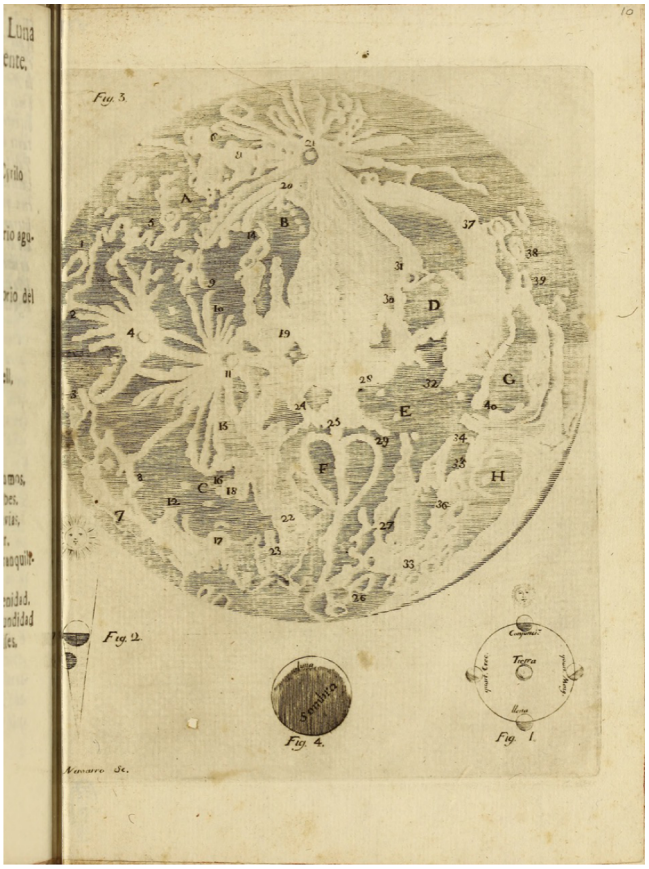


Fig. 3. Selenografía in Alzate, *Eclypse de luna del doce de diciembre de mil setecientos sesenta y nueve años: observado en la imperial ciudad de Mexico*, Mexico, 1770 (John Carter Brown Library).

collectors, who are key elements for understanding the production, circulation, interpretation and value given to the images. People are broadly described by name, occupation, place of birth and place of work. Finally, places and repositories are represented as classes in the KB. Places described can be both spaces of production, such as printing houses, observatories; and of circulation, such as universities, historic private libraries, intellectual and artistic salons. Since these places have not hitherto been documented, they were not initially considered in the ontology. However, their study provided enough information to consider them as classes in their own right instead of properties of images, books or people. The properties of individual places can now be enriched accordingly in the course of this research. On the other hand, repositories, whether libraries or archives, allow the researcher to keep account of the afterlife of images in the making of collections in contemporary cultural institutions. Repositories, whether archives or public and private libraries, can help the researcher locate the place of an image or a set of images in contexts of readership and collectors. Unexpected information found in different types repositories might supply evidence of the reception of each work studied and thus must be recorded to enrich the system.

The KB requires a strong amount of hand-labour for the description of each documented image, person, book, repository and place. However, this work allows the researcher to record findings according to how the information appears in the course of library and archival research, unrestricted by fixed categories. The expert is aware that the class hierarchy does not determine the structure of knowledge representation because their properties and relations can be adapted as knowledge of material studies evolves.

5.2. Design of the KB

The ontology for the present exercise is depicted in part in Figs. 4 and 5.[3] The main class `objects` has five daughters which are `people` and `images` –see Fig. 4– and `repositories`, `books` and `places` –see Fig. 5. The classes `people` and `places` have no daughters and have three and two particular individual instances respectively, with their corresponding identifiers, properties and relations. The classes `images`, `repositories` and `books` do have sub-classes with the corresponding individuals at the bottom. In

[3] The full code of this ontology is in the file `ains_taxonomy/kb_ains_final.txt` at https://github.com/KBS-Lab-IIMAS-UNAM/non-monotonic-KBS-for-DH.

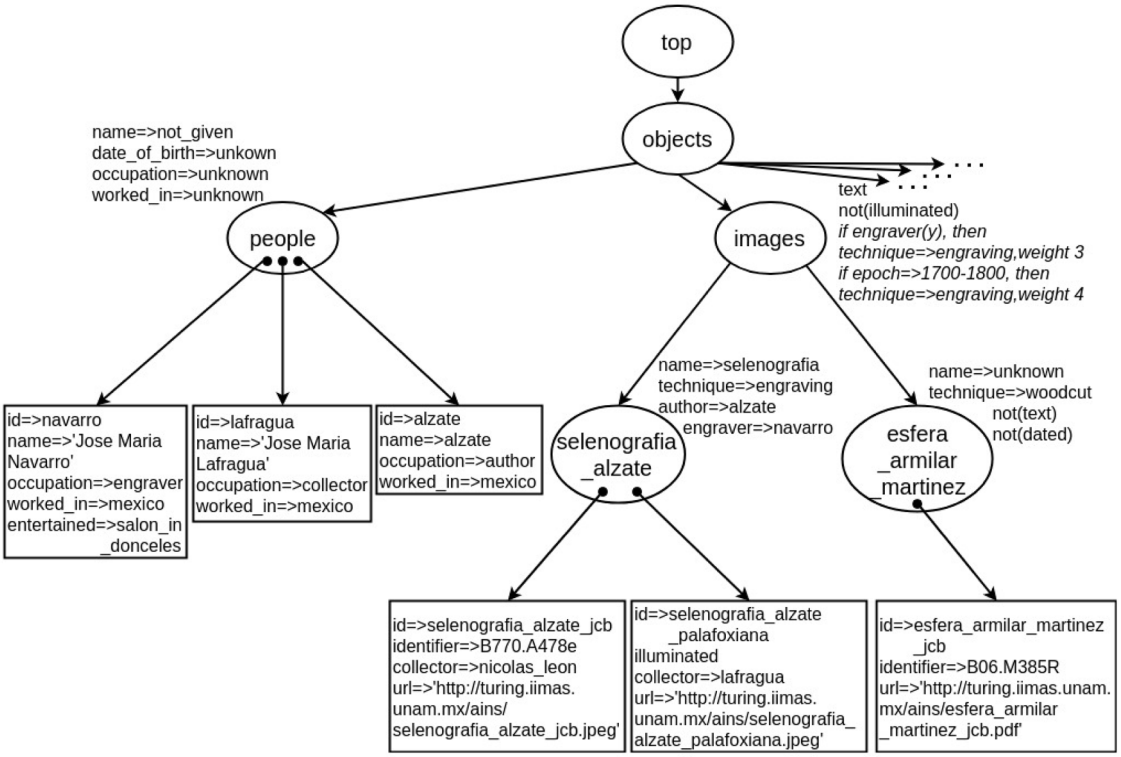


Fig. 4. Non-monotonic hierarchy in digital humanities (Part one).

the present study particular images, repositories and even books were modeled as classes, and their instances –which are material objects with a particular history and actual location– as individuals.

The research showed that there are properties and relations that hold for all members of the higher classes, and exceptions that are specified at the level of sub-classes or particular individuals. For instance, `images` have texts and are not illuminated as default properties, but also have exceptions in sub-classes or concrete individuals, such as the `selenografia_alzate_palafoxiana`, which is illuminated, and the `esfera_armilar_martinez` which does not have text. There are also properties and relations that are specific to individual objects, such as the `reportorio_martinez_jcb` that has `stamped_by`, `inscribed_by` and `includes`.

The research also showed that classes have preferences or conditional defaults that hold only whenever a particular condition is present; for instance, if the time of production of an image was the eighteenth century, it is most likely that the technique used to produce it was engraving; but this preference can be overridden by more specific information, such that the technique used for the `esfera_armilar_martinez` was woodcut.

The research also indicates that some observed facts can have explanations that are not explicit but follow possibly from other facts, profiting from the abductive capabilities of the representational scheme. For instance, according to this case study, if the technique used to make an image was engraving it is likely that the engraver was Navarro.

The taxonomy also features URL links to other resources, such as the picture of the Selenografía Alzate in the John Carter Brown library.

The construction of the ontology is a laborious process which starts from a basic hypothesis and is enriched and refined incrementally. The design and specification process may involve radical changes in the ontology, but these can be easily implemented using the utilities provided by our system.

Next, we briefly review the design process. The initial specification of the ontology is shown in Listing 2 and its taxonomy is depicted in Fig. 6. As can be seen, the class `objects` has only four daughters (i.e., `people`, `images`, `repositories` and `books`) and `images` has only three instances.

Suppose that in the course of the research a new instance of the Selenografía Alzate is discovered; hence, this is added as an individual instance to the class `images` with the id `selenografia_alzate_palafoxiana`. This is achieved through the KB-Service `add_object` that includes individuals into classes that have already been defined.

In the first stages of this research, it became relevant that places of production and circulation play an important role in the study of astronomical images in New Spain, such as `salon_in_donceles` that was used as a `salon` located in `'calle donceles'`. Hence, as an application of the class specialization operation, the class `places` has to be added into the ontology. Such addition of a new class involves a change of the conceptual model of the knowledge resource and changes significantly its structure. Refining the classes of an ontology to make finer distinctions is an important process in the evolution of knowledge, which in the present system is performed through a standard KB-Service as well (i.e., `add_class`).

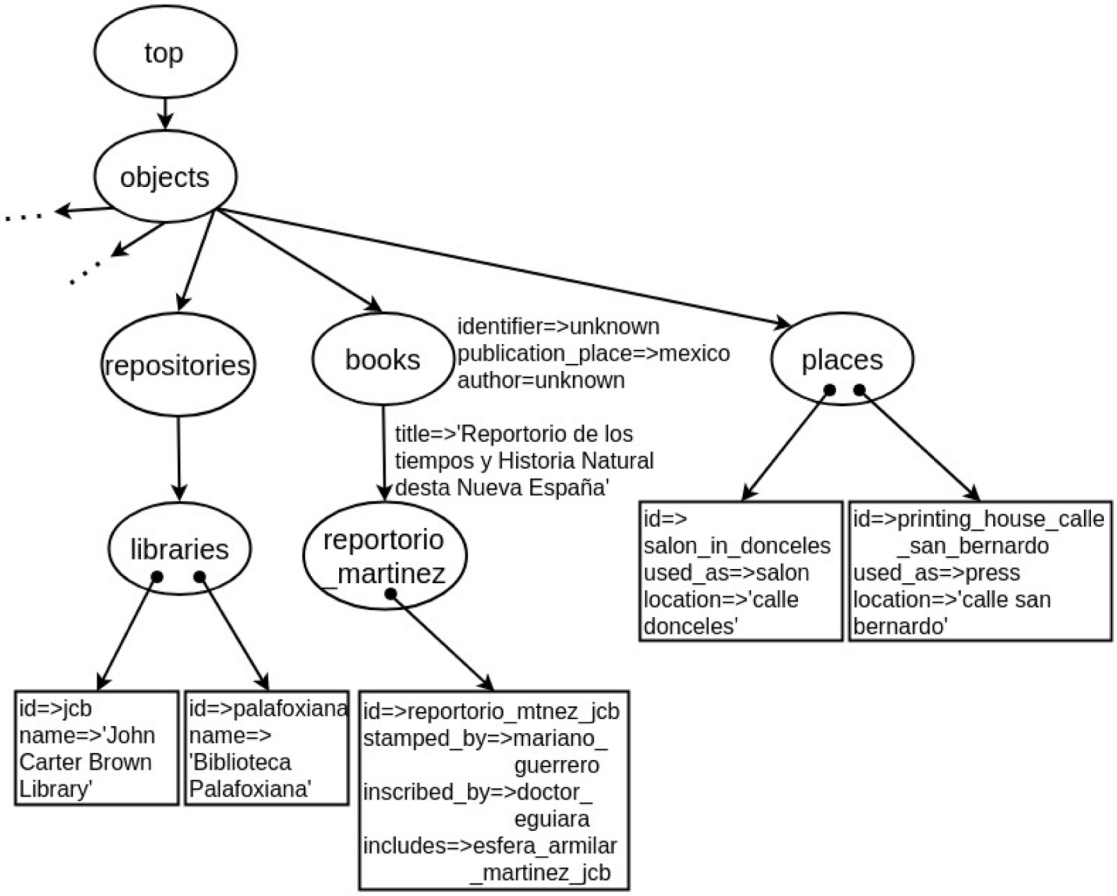


Fig. 5. Non-monotonic hierarchy in digital humanities (Part two).

Suppose further that throughout the research it will be observed that there are individuals in the KB who share a large number of attributes and hence they can be subsumed into a super-ordinate new class. For instance, `selenografia_alzate_jcb` and `selenografia_alzate_palafoxiana` are two material instances of Selenografía Alzate, which may be construed as an abstract class. The addition of such abstract class is an example of the application of the class abstraction operation. Refining the ontology with this new insight is also a major operation consisting in adding a new class and changing the extensions of the classes involved. The present system has three KB-Services to achieve these kinds of transformation: `add_class`, `remove_class` and `change_object_class`. The procedures for refining the ontology are shown in Appendix B.

In general, there is a set of complementary KB-Services, in addition to the basic ones provided in Section 4.2. In addition the system allows the definition by expert and general users of specialized Prolog queries, providing thus a means to retrieve particular information on demand according to the specific knowledge of interest for the problem in turn, this process is analogous to the SQL query generation common in relational data-bases. For the case-study presented in this section some specialized queries are shown next. The variable `KB` in the argument list holds the content of the full ontology.[4]

1. `extension_keyword(Keyword,KB,Extension)`: provides the set of objects that contain the given keyword.
2. `extension_all_images(KB,Extension)`: provides the set of images contained in the KB.
3. `extension_books_in_library(Library,KB,Extension)`: provides the set of books of a given library.
4. `content_of(Book,KB,Extension)`: provides the set of individuals contained in a given book.
5. `cause_of_technique(Image,KB,Cause)`: provides the possible cause of a given image for having the stated technique property.

[4] This is included in the file `ains_taxonomy/kb_ains_final.txt`. The Prolog code for the first query is given in Appendix C. The specification of the rest of the queries is similar and can be consulted at `https://github.com/KBS-Lab-IIMAS-UNAM/non-monotonic-KBS-for-DH` in the file `ains_taxonomy/cust_queries_ains.pl`.

```
[
class(top,none,[],[],[]),
class(objects, top, [], [], []),
class(images, objects, [], [],
 [
  [id=>selenografia_alzate_jcb,[[identifier=>'B770.A478f',0]],
                                    [[collector=>nicolas_leon,0]]],
  [id=>esfera_armilar_martinez_jcb,[[identifier=>'B06.M385R',0]],[]],
  [id=>selenografia_oculus_bnm,[],[[collector=>siguenza,0],
                                       [inscribed_by=>siguenza,0]]]
 ]),
class(people, objects, [], [],
 [
  [ id=>alzate,[[name=>alzate,0],[occupation=>author,0]],[] ],
  [ id=>lafragua,[[name=>'Jose Maria Lafragua',0],
                                    [occupation=>collector,0]],[] ],
  [ id=>siguenza,[[name=>'Carlos De Siguenza y Gongora',0],
                               [occupation=>[reader,author],0]],[] ]
 ]),
class(books, objects, [], [],
 [
  [id=>reportorio_martinez_jcb,[],[[includes=>
                  esfera_armilar_martinez_jcb,0],[held_in=>jcb,0]]]
 ]),
class(repositories, objects, [], [], []),
class(libraries, repositories, [], [], [
  [id=>jcb,[],[]],
  [id=>bnm,[],[]]
 ])
]
```

Listing 2. Initial taxonomy for the Astronomical Images of New Spain case-study.

6. Discussion and further work

In the present paper we have investigated the creation and use of knowledge resources for un-regimented knowledge domains. These domains are focused on the representation of individuals with unique characteristics or uncommon objects that merit particular studies. The specification of the domain involves the definition of a taxonomy, with classes and individuals, that have general and specific properties and relations respectively. The knowledge is normally acquired incrementally, its expression may require strong negation and the addition of new knowledge may result in incoherent or even contradictory propositions. The interpretation of expressions in un-regimented domains needs to assume that the knowledge is incomplete due to omissions or perhaps because there are objects that have not been discovered yet.

For these reasons such kinds of domains cannot be easily captured with predefined schemes or data-base models in which the knowledge is regimented and subject to standardization. For instance, if a standard relational data-base is adopted but the knowledge turns out to be un-regimented, even small additions can lead to a full redesign of the conceptual model, and this is a costly and time consuming exercise. In addition, while in regimented domains there is a clear demarcation between developers and users, the same individual or team can play both roles in un-regimented ones.

The main proposition of this paper is that the dynamic extension of the conceptual model gives rise to non-monotonic phenomena and requires a non-monotonic knowledge-base service. We have shown how such a kind of resource can be applied to the

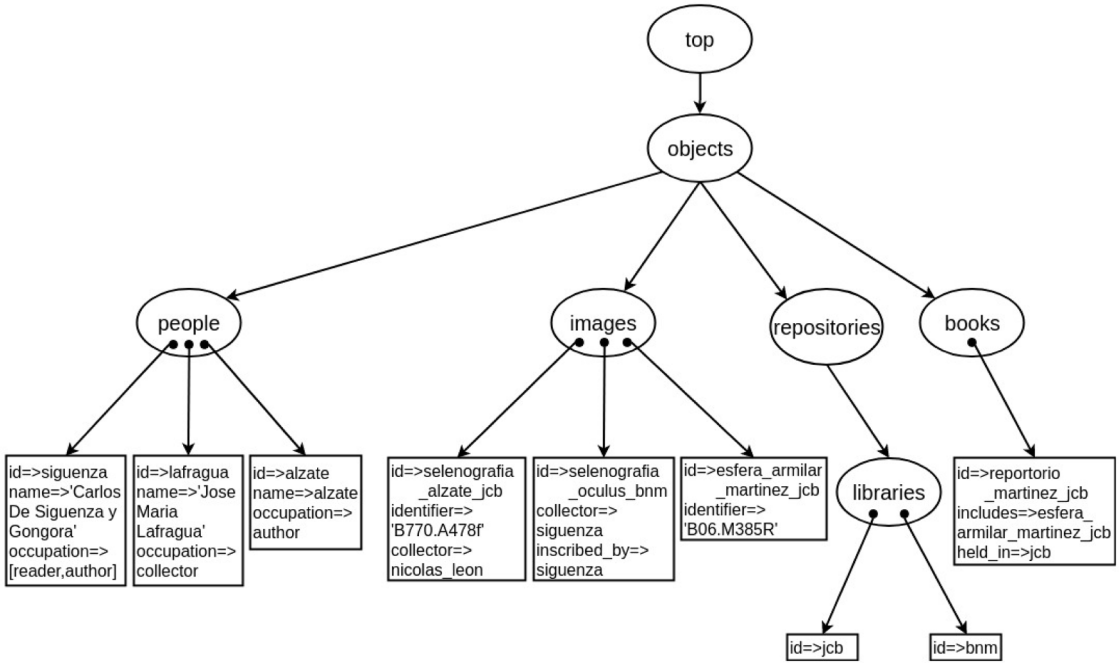


Fig. 6. Initial hierarchy for the Astronomical Images of New Spain case study.

construction of information systems for un-regimented domains, and a case study in the digital humanities was developed with promising results.

The present resource can have practical applications for the development of distributed open-ended knowledge domains. We imagine an scenario in which researchers and specialists around the world can develop a large knowledge resource focused on individual objects or collections; they may work in different languages and have different –not completely coherent– perceptual perspectives, and play the role of developers and final users in different occasions.

Such an environment would require the creation of segments of the knowledge-base, placed in arbitrary locations, that could be merged in central repositories or split for particular views. For this the KB-service would need to have dynamic interfaces through which the user could visualize queries in textual, graph, diagrammatic or graphical forms. The queries and updates may involve references to images of the actual objects of interest, and related information.

The user would be able to consult the information, but also to input or modify the description of objects, with their properties and relations, and also to modify the structure of the knowledge-base, with the guarantee that the integrity of the information is preserved through such manipulations.

For further work we plan to create a distributed multimodal interface that can be used by a community of researchers working in different places and times. This resource will be used to carry on an evaluation exercise consisting in the construction of a large knowledge resource in a considerable larger scenario –including a much larger number of classes and individual objects.

CRediT authorship contribution statement

Luis A. Pineda: Writing - original draft, Writing - review & editing. **Noé Hernández:** Writing - review & editing. **Iván Torres:** Writing - review & editing. **Gibrán Fuentes:** Writing - review & editing. **Nydia Pineda De Avila:** Writing - review & editing.

Acknowledgments

The authors acknowledge the support of CONACyT's Project 178673 and PAPIIT-UNAM Projects IN109816 and IN112819.

Appendix A. Use and incremental development of the knowledge-base

The use of the KB-Service is illustrated in this appendix. The knowledge-base is specified as a external text file (e.g., `kb_university.txt`) that is read into a Prolog variable `KB`, which is passed as an argument of the KB-Services as follows:

1. Return the extension of a class:

```
?- open_kb('kb_university.txt',KB),
   class_extension(abstract,KB,Extension_Class).
Extension_Clas = [prog,ai].
```

(i.e., all individuals within the class *abstract*).

2. Return the extension of a property:

```
?- open_kb('kb_university.txt',KB),
   property_extension(study,KB,Extension_Property).
Extension_Property = [pete:uk,anne:mexico].
```

(i.e., all individuals that have the property *study*).

3. Return the extension of a relation:

```
?- open_kb('kb_university.txt',KB),
   relation_extension(lectures,KB,Extension_Relation).
Extension_Relation = [mary:[ai],tom:[prog]].
```

(i.e., all individuals that are related to *lectures*).

4. Return the extension of an explanation:

```
?- open_kb('kb_university.txt',KB),
   explanation_extension(study=>uk,KB,Extension_Explanation).
Extension_Explanation = [pete:(work=>uk)].
```

(i.e., all individuals that have the property *study* $=>$ *uk*).

5. Return the classes of an individual:

```
?- open_kb('kb_university.txt',KB),
   classes_of_individual(mary,KB,Classes_of).
Classes_of = ['faculty members',people,concrete,objects,top].
```

(i.e., all classes of *mary*).

6. Return the properties of an individual:

```
?- open_kb('kb_university.txt',KB),
   properties_of_individual(tom,KB,Properties_Of).
Properties_of = [sport,size=>short,fun,not(teach)].
```

(i.e., all properties of *tom*).

7. Return the relations of an individual:

```
?- open_kb('kb_university.txt',KB),
   relations_of_individual(pete,KB,Relations_Of).
Relations_Of = [enrolled=>[ai]].
```

(i.e., all relations of *pete*).

8. Return the explanations of an individual:

```
?- open_kb('kb_university.txt',KB),
   explanation_of_individual(anne,KB,Explanation_Of).
Explanation_Of = [(study=>mexico):(work=>mexico)].
```

(i.e., all explanations of *anne*).

The KB-Services are also used to define customized queries; for instance, for retrieving the students enrolled in a course or the rooms that are not assigned to courses. These are standard Prolog queries using the KB-Services. The definitions of these predicates are available at `university_taxonomy/cust_queries_univ.pl` in the GitHub repository, and are loaded by running the command `consult('cust_queries_univ.pl').` in the same session where the system is active. Their use is illustrated as follows:

- Return the students attending a course:

```
?- open_kb('kb_university.txt',KB),
   students_course(ai,KB,Students_at_Course).
Students_at_Course = [pete,anne].
```

  (i.e., all students attending the course *ai*).
- Return the rooms not assigned to courses:

```
?- open_kb('kb_university.txt',KB),
   available_rooms(KB,Rooms_Available).
Rooms_Available = [].
```

  (i.e., there is no *room* not assigned to courses).

Next, we illustrate examples of services for adding, removing and changing the KB's content.

- The class `teaching assistants` is added as a subclass of `people`. The new class has the property that its individuals `teach` and `grade`. The items that they grade are: `quizzes`, `projects` and `problem sets`. Also, teaching assistants gather in the `teachers_room`. These changes are expressed below. First, the current KB is retrieved from an external file. Then each modification by the KB-Services is saved in a variable holding the current state of the KB, that is passed to the next predicate. The final KB is saved as `kb_add_class.txt`:

```
open_kb('kb_university.txt',KB),
add_class('teaching assistants',people,KB,KB1),
add_class_property('teaching assistants',teach,yes,KB1,KB2),
add_class_property('teaching assistants',grade,[quizzes,
                          projects,'problem sets'],KB2,KB3),
add_class_relation('teaching assistants',found,teachers_room,
                                                  KB3,KB4),
save_kb('kb_add_class.txt',KB4).
```

- The individual `john` is added to the class teaching assistants –created in the previous example. A property and a relation are defined for him: john is a `sport` person and `works` for `mary`. Then the `teachers_room` is added as an individual to the class `rooms`. These additions are saved as the external file `kb_add_obj.txt`.

```
open_kb('kb_add_class.txt',KB),
add_object(john,'teaching assistants',KB,KB1),
add_object_property(john,sport,yes,KB1,KB2),
add_object_relation(john,works_for,mary,KB2,KB3),
add_object(teachers_room,rooms,KB3,KB4),
save_kb('kb_add_obj.txt',KB4).
```

- Next, the class `library staff` is removed, as well as the individual `pete` within the class `student`. The resulting KB is saved as `kb_rm.txt`:

```
open_kb('kb_add_obj.txt',KB),
rm_class('library staff',KB,KB1),
rm_object(pete,KB1,KB2),
save_kb('kb_rm.txt',KB2).
```

- The value of the property `size` of the class `rooms` is updated, and the default value of this property is now `huge`; the name `john` is changed to `brian`, and the object of the relation `lectures` of `mary` is updated to `prog`. These changes are saved as `kb_change.txt`:

```
open_kb('kb_rm.txt',KB),
change_value_class_property(rooms,size,huge,KB,KB1),
change_object_name(john,brian,KB1,KB2),
change_value_object_relation(mary,lectures,prog,KB2,KB3),
save_kb('kb_change.txt',KB3).
```

- Finally, the weight of a preference is updated. The resulting KB is saved as `university_taxonomy/`

kb_preference_university.txt in the GitHub repository :

```
open_kb('kb_change.txt',KB),
change_weight_class_property_preference(students,like=>'-'
                                        =>>study=>'-',7,KB,KB1),
save_kb('kb_preference_university.txt',KB1).
```

Appendix B. Procedures for refining the ontology

In this section some examples of procedures for modifying the ontology and for consulting the extension of classes in the case-study are presented. The KB is stored in an external file (which is a standard .txt file) which is consulted and stored when the query is started and concluded respectively. As can be seen the whole KB is an argument that is passed as input and output of the predicates that read and update it. The full code of the examples can be seen at the GitHub repository of the KB system.

- Add the image selenografia_alzate_palafoxiana with its id, its collector and the fact that it comes illuminated. Also add the class places with salon_en_donceles as its individual. And amend the id of selenografia_alzate_jcb:

```
open_kb('kb_ains_initial.txt',KB),
add_object(selenografia_alzate_palafoxiana,images,KB,KB1),
add_object_property(selenografia_alzate_palafoxiana,illuminated,
                                                    yes,KB1,KB2),
add_object_relation(selenografia_alzate_palafoxiana,collector,
                                                lafragua,KB2,KB3),
add_class(places,objects,KB3,KB4),
add_object(salon_en_donceles,places,KB4,KB5),
add_object_property(salon_en_donceles,used_as,salon,KB5,KB6),
add_object_property(salon_en_donceles,location,'calle donceles',
                                                        KB6,KB7),
change_value_object_property(selenografia_alzate_jcb,identifier,
                                            'B770.A478e',KB7,KB8),
save_kb('kb_ains_initial_2.txt',KB8).
```

- Add class selenografia_alzate whose mother is images and reassign the objects of the class:

```
open_kb('kb_ains_initial_2.txt',KB),
add_class(selenografia_alzate,images,KB,KB2),
change_object_class(selenografia_alzate_palafoxiana, selenografia_
                                                alzate,KB2,KB3),
change_object_class(selenografia_alzate_jcb,selenografia_alzate,
                                                        KB3,KB4),
save_kb('kb_ains_initial_3.txt',KB4).
```

Appendix C. Example of Prolog code for custom queries

The code in Listing C.3 shows the implementation of the custom query extension_keyword. It first obtains a list of all images that are sub-classes of images, then by the auxiliary predicate check_keyword_img each image is examined to know if its keyword property contains the sough value Keyword. If so, such an image is added to the final list Extension.

In order to use all custom queries defined for this case-study, the code in cust_queries_ains.pl must be first loaded as follows:

```
?- consult('cust_queries_ains.pl').
```

By the query defined above, the following instruction retrieves the images with keyword moon from the KB saved as ains_taxonomy/kb_ains_final.txt in the GitHub repository for this paper:

```
?- open_kb('kb_ains_final.txt',KB),extension_keyword(moon,KB,Extension).
```

The answer returned by the KB system is:

```
Extension = [selenografia_alzate, selenografia_oculus].
```

```
extension_keyword(Keyword,KB,Extension):-
        sons_of_class(images,KB,List_Imgs),
        check_keyword_img(List_Imgs,Keyword,KB,Extension).



check_keyword_img([],_,_,[]).
check_keyword_img([Img|More],Keyword,KB,[Img|Extension]):-
        class_property_value(Img,key_words,KB,Val),
        (is_list(Val)
         -> isElement(Keyword,Val),!
         ;  Keyword = Val,!
        ),
        check_keyword_img(More,Keyword,KB,Extension).
check_keyword_img([_|More],Keyword,KB,Extension):-
        check_keyword_img(More,Keyword,KB,Extension).
```

Listing C3. Prolog code for the custom query `extension_keyword`.

Supplementary material

Supplementary material associated with this article can be found, in the online version, at 10.1016/j.ipm.2020.102214 .

References

Aalberg, C. T., Balzer, D., Bekiari, C., Boudouri, L., Crofts, N., Dunsire, G., ... Žumer, M. (2018). Definition of the cidoc conceptual reference model [6.2.3, concurrent version]. In M. Doerr, P. L. Boeuf, C. E. Ore, & S. Stead (Eds.). .

Aït-Kaci, H., Nasr, R., & Seo, J. (1990). Implementing a knowledge-based library information system with typed horn logic. *Information Processing & Management, 26*(2), 249–268.

Avery, J., & Yearwood, J. (2003). *DOWL: A dynamic ontology language. Proceedings of the IADIS international conference www/internet, Algarve, Portugal*985–988.

Bechhofer, S., Van Harmelen, F., Hendler, J., Horrocks, I., McGuinness, D. L., Patel-Schneider, P. F., Stein, L. A., et al. (2004). OWL Web Ontology Language Reference. In M. Dean, & G. Schreiber (Eds.). *W3c recommendation*World Wide Web Consortium (W3C)https://www.w3.org/TR/2004/REC-owl-ref-20040210/.

Becker, J., Bersch, C., Pangercic, D., Pitzer, B., Rühr, T., Sankaran, B., ... Burgard, W. (2011). *The pr2 workshop-mobile manipulation of kitchen containers. Proceedings of the iros workshop on results, challenges and lessons learned in advancing robots with a common platform, San Francisco, California, USA*.

Bingi, R., Khazanchi, D., & Yadav, S. B. (1995). A framework for the comparative analysis and evaluation of knowledge representation schemes. *Information Processing & Management, 31*(2), 233–247.

Brachman, R. J., & Schmolze, J. G. (1985). An overview of the KL-ONE knowledge representation system. *Cognitive Science, 9*(2), 171–216.

Brickley, D., & Guha, R. V. (2014). RDF Schema 1.1. In D. Brickley, & R. Guha (Eds.). *W3C recommendation*World Wide Web Consortium (W3C)https://www.w3.org/TR/2014/REC-rdf-schema-20140225/.

Bruseker, G., Carboni, N., & Guillem, A. (2017). Cultural heritage data management: The role of formal ontology and cidoc crm. In M. L. Vincent, V. M. López-Menchero Bendicho, M. Ioannides, & T. E. Levy (Eds.). Springer International Publishing.

Buckner, C., Niepert, M., & Allen, C. (2011). From encyclopedia to ontology: Toward dynamic representation of the discipline of philosophy. *Synthese, 182*(2), 205–233.

Doerr, M. (2003). The cidoc conceptual reference module: An ontological approach to semantic interoperability of metadata. *AI Magazine, 24*(3), 75–92.

Domingue, J., Motta, E., & Garcia, O. C. (1999). Knowledge modelling in webonto and ocml – a user guide (version 2.4). Knowledge Media Institute, Open University.

Doyle, J. (1979). A truth maintenance system. *Artificial Intelligence, 12*(3), 251–272.

Fan, Z., Tosello, E., Palmia, M., & Pagello, E. (2014). *Applying semantic web technologies to multi-robot coordination. Proceedings of the workshop on new research frontiers for intelligent autonomous systems, Venice, Italy*.

Görz, Günther, Schiemann, B., & Oischinger, M. (2008). *An implementation of the CIDOC conceptual reference model (4.2.4) in OWL-DL. Proceedings of the 2008 annual conference of cidoc – the digital curation of cultural heritage, Athens, Greece*1–14.

Guha, R. V., & Lenat, D. B. (1991). CYC: A mid-term report. *Applied Artificial Intelligence an International Journal, 5*(1), 45–86.

Guillem, A., & Bruseker, G. (2017). The cidoc crm game: A serious game approach to ontology learning. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, XLII-2/W5*, 317–323.

K, S., & Thilagam, P. S. (2019). Crime base: Towards building a knowledge base for crime entities and their relationships from online news papers. *Information Processing & Management, 56*(6), 102059.

Kifer, M., & Boley, H. (2013). *RIF overview (Second Edition)W3C Note,*. W3C.

Klinov, P. (2008). Pronto: A non-monotonic probabilistic description logic reasoner. In S. Bechhofer, M. Hauswirth, J. Hoffmann, & M. Koubarakis (Eds.). *Proceedings of the 5th european semantic web conference, Tenerife, Canary Islands, Spain* (pp. 822–826). .

Levesque, H. L., & Brachman, R. (1985). A fundamental tradeoff in knowledge representation and reasoning. In H. L. R. Brachman (Ed.). *Readings in knowledge representation* (pp. 41–70). Los Altos, CA, USA: Morgan and Kaufmann.

Miles, A., & Bechhofer, S. (2009). *Skos simple knowledge organization system reference. World Wide Web Consortium (W3C)*https://www.w3.org/TR/2009/REC-skos-reference-20090818/.

Motta, E. (1998). *Reusable components for knowledge modelling.* Knowledge Media Institute. The Open University, UK Ph.D. thesis.

Niepert, M., Buckner, C., & Allen, C. (2007). *A dynamic ontology for a dynamic reference work. Proceedings of the 7th acm/ieee-cs joint conference on digital libraries, Vancouver, BC, Canada*288–297.
Niepert, M., Buckner, C., & Allen, C. (2008). *Answer set programming on expert feedback to populate and extend dynamic ontologies. Proceedings of the 21st international florida artificial intelligence research society conference, Coconut Grove, Florida, USA*500–505.
Niwa, K., Sasaki, K., & Ihara, H. (1984). An experimental comparison of knowledge representation schemes. *AI Magazine, 5*(2), 29–36.
Pangercic, D., Tenorth, M., Jain, D., & Beetz, M. (2010). *Combining perception and knowledge processing for everyday manipulation. Ieee/rsj international conference on intelligent robots and systems, Taipei, Taiwan*1065–1071.
Pineda, L. A., Rodríguez, A., Fuentes, G., Hernández, N., Reyes, M., Rascón, C., ... Ortega, H. (2018). Opportunistic inference and emotion in service robots. *Journal of Intelligent & Fuzzy Systems, 34*(5), 3301–3311.
Pineda, L. A., Rodríguez, A., Fuentes, G., Rascón, C., & Meza, I. (2017). A light non-monotonic knowledge-base for service robots. *Intel Serv Robotics, 10*, 159–171.
Reiter, R. (1980). A logic for default reasoning. *Artificial Intelligence, 13*, 81–132.
Ryu, P.-M., Jang, M.-G., & Kim, H.-K. (2014). Open domain question answering using wikipedia-based knowledge model. *Information Processing & Management, 50*(5), 683–692.
Shadbolt, N., Berners-Lee, T., & Hall, W. (2006). The semantic web revisited. *IEEE Intelligent Systems, 21*(3), 96–101.
Sirin, E., Parsia, B., Grau, B. C., Kalyanpur, A., & Katz, Y. (2007). Pellet: A practical owl-dl reasoner. *Web Semantics, 5*(2), 51–53.
Tang, X., Chen, L., Cui, J., & Wei, B. (2019). Knowledge representation learning with entity descriptions, hierarchical types, and textual relations. *Information Processing & Management, 56*(3), 809–822.
Tenorth, M., Kunze, L., Jain, D., & Beetz, M. (2010). *Knowrob-map - knowledge-linked semantic object maps. Proceedings of the 10th ieee-ras international conference on humanoid robots, Nashville, TN, USA*430–435.
Torres, I., Hernández, N., Rodríguez, A., Fuentes, G., & Pineda, L. A. (2019). Reasoning with preferences is service robots. *Journal of Intelligent & Fuzzy Systems, 36*(5), 5105–5114.
Weibel, S. L., & Koch, T. (2000). The Dublin Core metadata initiative: Mission, current activities, and future directions. *D-Lib Magazine, 6*(12).