Foundational Challenges in Automated Semantic Web Data and Ontology Cleaning

José A. Alonso-Jiménez, Joaquín Borrego-Díaz, Antonia M. Chávez-González, and Francisco J. Martín-Mateos, *Universidad de Sevilla*

Applying automated reasoning systems to Semantic Web data cleaning and to cleaning-agent design raises many challenges. We can build trust in Semantic Web logic only if it's based on certified reasoning.

Nowadays, Web-based data management needs tools to ensure secure, trustworthy performance. The utopian future shows a Semantic Web providing dependable frameworks that can solve many of today's data problems. However, the realistic immediate future raises several challenges, including foundational Semantic Web issues, the abstract definition of data, and incomplete, evolving ontologies. In either case, the marriage of data and ontologies is indissoluble and represents the knowledge database (KDB), a basic ingredient of the Semantic Web.

Automated reasoning systems (ARSs) might be helpful in cleaning KDBs. Because we're particularly interested in logical reasoning and its robustness for data cleaning and preprocessing, our work focuses on qualitative KDBs. In this article, we look closely at problems in data analysis, the first phase of data cleaning.

Currently, ARSs play the assistant role in an ARS-aided methodology to clean KDB anomalies.[1] The overall question that it addresses is whether it's possible to design trustworthy data-cleaning systems to certify both the KDB and the reasoning on it, as the Semantic Web promises. This challenge emphasizes the current need for explaining the reasoning behind cleaning programs.

The challenges

The long-term goal for Semantic Web data cleaning is to design general-purpose cleaning agents—intelligent agents that can find and repair both ontology and data anomalies in KDBs. To achieve this goal, we first need to analyze the challenges that this goal raises.

What is the problem's logical complexity?

Cleaning KDBs in a dynamic environment such as the Semantic Web is quite hard. Here are some reasons:

- We can't suppose that the KDB is complete. Even if we have a good KDB, anomalies might arise again the next time we introduce data.
- The ontology of the KDB in the Semantic Web is usually a complex theory, so the classical database theories (closed-world assumption, unique-names principle, and domain closure axiom) would likely become inconsistent. Nevertheless, the KDB itself represents a real model.
- The KDB doesn't contain facts about all the relations in the language. Typically, only nearly complete information about some relations and concepts (perhaps considered the primary ones) is present. Other complex notions must be deduced.

We obtained this list of reasons by revising the classical verification and validation problem in the expert systems field.[2] Anomalies might come from several sources, such as the following:

- The data set might be inconsistent with the ontology, owing to formal inconsistencies produced by wrong data or the absence of some knowledge.
- The database might not give us complete information about primary predicates. For example, the automated theorem prover (ATP) might deduce the existence of objects whose names aren't explicit in the KDB.

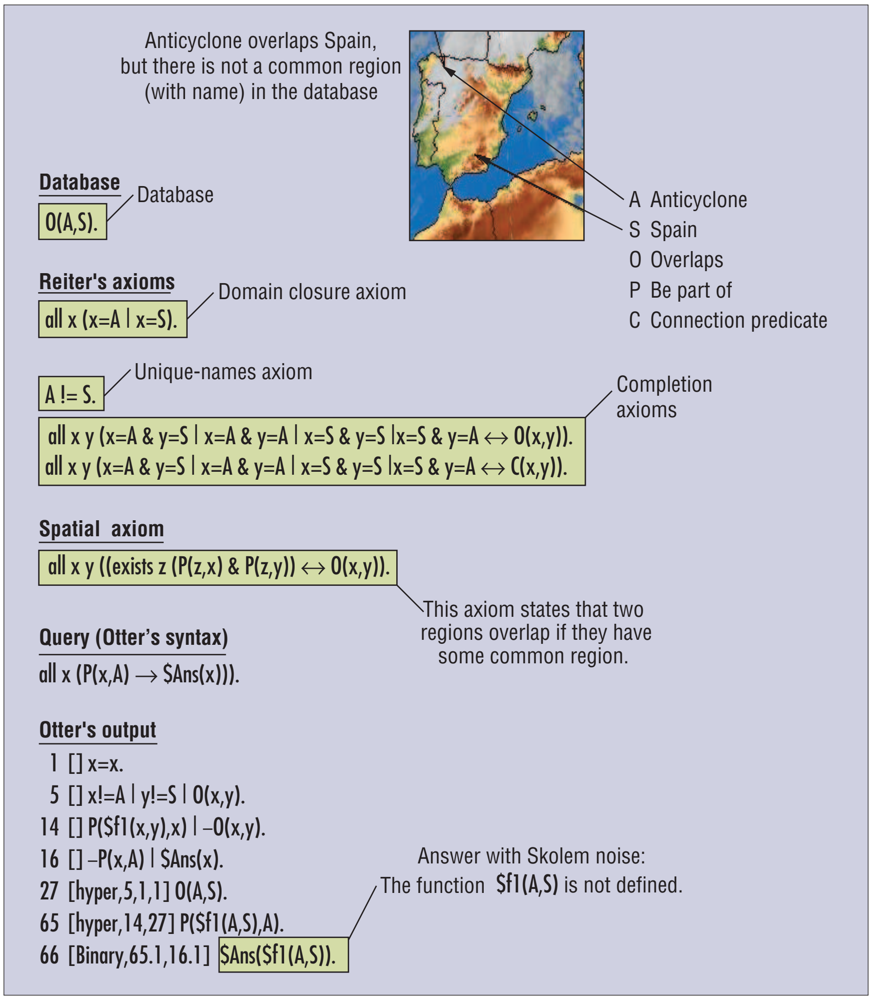
- Answers to standard queries might be disjunctive (a logical deficiency).
- The ontology might be inconsistent.

Also, when we work with many information sources, to accept the answers computed from selected resources, we must estimate the quality of the answer—a potentially critical task.

This problem forces us to balance real-time processing and complex reasoning. This dilemma is, in essence, the same as reactivity versus proactivity attitudes in agents. The ARS assistant might need more autonomy to work with complex logical features, so tightly constraining the ARS's processing time isn't advisable. Nevertheless, how much autonomy it needs isn't clear: an ATP can produce an overspill of new knowledge, not all of it interesting. So, it needs a strict referee layer to manage the workflow. This isn't a new idea in automated deduction.[3] These are the main drawbacks for integrating an ARS in an agent architecture. However, losing real-time requirements might not be important: the system could work as a night cleaning service, debugging metadata the user has implemented during the computer's idle time.

Another interesting aspect is that some languages for specifying ontologies (for example, KIF, or Knowledge Interchange Format; http://logic.stanford.edu/kif/kif.html) let us design syntactically complex theories that can be hard to manage. Complex axioms in an ontology description often come from a deficient or inconsistent set of concepts and relationships. A paradigmatic case is where several programmers in a company are developing an ontology and associated tools while other programmers are inputting the data. This cooperative work could produce inconsistencies or complex axioms that make the ontology messy. Finally, mechanisms for evaluating ontology engineering tasks are sometimes lacking, preventing their commercial use.[4]

Finally, providing metadata is a difficult task that users tend to avoid: to save costs, they might use an ontology-learning system to build an initial ontology, but then they'd have to debug it. An ontological analysis of the intended meaning of the ontology's elements would even be necessary.[5] In any case, it's necessary to know how to work with poor (or provisional) ontologies.


Anticyclone overlaps Spain, but there is not a common region (with name) in the database
A Anticyclone
S Spain
O Overlaps
P Be part of
C Connection predicate
Database
Domain closure axiom
Unique-names axiom
Completion axioms
This axiom states that two regions overlap if they have some common region.
Answer with Skolem noise: The function $f1(A,S) is not defined.

```
Database
O(A,S).

Reiter's axioms
all x (x=A | x=S).

A != S.

all x y (x=A & y=S | x=A & y=A | x=S & y=S |x=S & y=A <-> O(x,y)).
all x y (x=A & y=S | x=A & y=A | x=S & y=S |x=S & y=A <-> C(x,y)).

Spatial axiom
all x y ((exists z (P(z,x) & P(z,y)) <-> O(x,y)).

Query (Otter's syntax)
all x (P(x,A) -> $Ans(x))).

Otter's output
 1 [] x=x.
 5 [] x!=A | y!=S | O(x,y).
14 [] P($f1(x,y),x) | -O(x,y).
16 [] -P(x,A) | $Ans(x).
27 [hyper,5,1,1] O(A,S).
65 [hyper,14,27] P($f1(A,S),A).
66 [Binary,65.1,16.1] $Ans($f1(A,S)).
```

Figure 1. A simple example of Skolem noise created by the Otter system as it reasons with a poor ontology.

How do we work with poor ontologies?

The problem is partially one of logical reasoning with weak theories. We assume that the Semantic Web's content is (explicitly or implicitly) in the form of logical formulas. From this viewpoint, we think that using ontologies to manage data implies logical reasoning that's deeper than what we believe is advisable. Ontologies aren't only logical theories; they also have additional features such as backward compatibility.[6] Clearly, however, a primary security criterion for ontologies is logical trust.

From a logic point of view, data are ground terms constrained by an ontology language and, more important, whose behavior is specified by the ontology. A partially built ontology constrains us to reason with a poor language or a deficient set of axioms.

Unstable ontologies persist for several reasons. Most ontology research involves developing representation paradigms rather than features associated with evolution such as persistence, transactions, or scalability.[7] Experience has shown that building a robust ontology, including metadata, is expensive. So, although it's best to invest that effort in the early development stages, this impedes significant changes later on, due to its high cost.

In ATP-aided cleaning tasks, an interesting reason for bottom-up change generation in ontologies is *Skolem noise* (the noise produced when we work with domain closure axioms but when the domain knowledge is not clausal). Figure 1 shows an example of Skolem noise created by the Otter ATP system as it reasons with a poor ontology. The noise is caused by the analysis of the interaction among the user, the KDB, and the Otter ATP when the user works with provisional ontologies. We can't wait until the ontology becomes stable—but at least we can wait until the meaning of "stable," "better," or "robust" is clear.

When is an ontology robust?

Ontologies are presumably designed to protect us from managing information incorrectly. But, can we be sure of the current ontology? Is it possible to predict that only minor changes will occur? Ontologies must be maintained just like other parts of a system (for example, by refinement[8] or versioning[9]). The definition of ontology robustness has several perspectives; all are necessary, but none are sufficient.

From the Semantic Web perspective, what's considered essential for robustness comes from some of the requirements for exploiting Semantic Web ontologies, mainly these two:[6]

- A Semantic Web ontology should be able to extend other ontologies with new terms and definitions.
- An ontology's revision shouldn't change the well-foundedness of the resources that commit to an earlier version of the ontology.

Nevertheless, even if we can extend an ontology, its *core* should be stable. The core is the portion of an ontology's source code that represents a theory with well-known properties and is accepted as best fitting the ontology's concepts. The core should include the top of the ontology (general concepts). The top can be a standard proposal—for example, SUMO (Suggested Upper Merged Ontology; http://ontology.teknowledge.com).

From a logical perspective, *robust ontology* should mean *complete logical theory*. We might apply this definition in the context of the full OWL (www.w3.org/TR/owl-features) or at least for some coherent parts of an ontology. However, this isn't a local notion: minor changes compromise logical completeness in a dramatic way. Other logical notions, such as *categoricity*, clash with natural logical principles for database reasoning such as the closed-world assumption or unique-names principle.

Therefore, robustness should combine both perspectives—the Semantic Web perspective and the logic perspective—along with a notion of a clear (as opposed to a messy) ontology. A messy ontology might become the target of a cleaning process if daily management of the ontology suggests it. So, an ontology is robust if

- its core is clear and stable (except for extensions),
- every model of its core exhibits similar properties with regard to the core language, and
- it can admit (minor) changes made outside the core without compromising core consistency.

We understand *similar properties* as a set of metalogical properties, so all models with similar properties are essentially the same: they can't be distinguished by means of natural properties.

But you can't understand robustness without considering dynamic aspects. Mainly, we regard robustness as a measure: evolution can extend the core to a big portion of the ontology, except for data. Evolution lets us locate possible inconsistencies in the data, so the problem begins to resemble integrity constraint checking.

Robustness should combine both the Semantic Web perspective and the logic perspective, along with a notion of a clear ontology.

In the example in figure 1, the Skolem noise phenomenon suggests that we should add to the ontology the interpretation of `$f1` as the intersection of regions (when they intersect). Thus, it's advisable to evolve the ontology to represent the behavior of the intersection between spatial regions.

The dynamics of ontology evolution lead us to adapt other ontologies to extend ours, or generally to design systems for managing multiple and distributed ontologies.[10] This question brings up another: how can we design sound ontology mappings for automated reasoning?

Does ontology mapping mean theory interpretation?

The semantic heterogeneity of ontologies is a major barrier to cleaning data. Ontology mapping will become a key tool in heterogeneous and competitive scenarios, such as in e-commerce, knowledge management in large organizations, and—in the Semantic Web—the usual extraction/transformation/loading process[11] for data cleaning. However, an ontology mapping's logical—and cognitive—effects on automated reasoning aren't clear. A logical perspective once again leads to a rigid concept, *interpretation among theories*. This is the most suitable for logical automated reasoning, but this approach has limited application. In practice, one solution is to use contexts (or microtheories), as in the Cyc (www.cyc.com) ontology. Other proposals such as Context OWL[12] might be useful in controlling the expansion of anomalies to the complete ontology.

Ontology mapping has limited usefulness. At some point, intelligent KDB analysis aided by an ATP will need a logical interpretation of basic data-like integers (for example, when we accept that *number of parts* is a positive integer). Building an ontology on numeric data and their properties isn't easy. Even if we have one, the ontology mapping can be understood as a logical interpretation of an arithmetic theory (remember that, in OWL, mappings aren't part of the language). Because interpretation means, up to a point, incorporating a logical theory about such data into the target ontology, logical incompleteness isn't only assumed but unavoidable. (Also remember that although OWL integrates data types, it includes nothing about integer arithmetic, for example.) We can tame this phenomenon by syntactically restricting queries, but the solution will be hard in any case: extending OWL (regarding it as a logical theory) will add features for numerical-data representation and reasoning. Moreover, undecidability will be intrinsic to any language with powerful features and more complex tools. This is a definitive barrier to ontology language design. We must find a way to escape from this in practice.

Is there a model theory for the Semantic Web?

At the top of the Semantic Web "cake" proposed by Tim Berners-Lee, logic and proof appear as the bases beneath trust (see figure 2).[13] Logical trust, in a broad sense, is based on logical semantics, and this deals with models and the definition of truth. Ontology designers often forget about model-theoretic analysis because they have a particular (real or intended) model in mind. A well-known principle in knowledge representation states that no language or KDB exists that can faithfully represent the intended world where we want to work; that is, unintended models exist (see figure 3). In the Qualitative Spatial Reasoning field, this principle turns into the *poverty conjecture*: no purely qualitative, general-purpose kinematics exists. Therefore, no categorical ontology exists for QSR.

One goal of logical-model theory is to study all of a theory's models, including the unintended ones. This isn't only a semantic basis for good specification. It affects practical proving: the existence of unintended models is the consequence of incompleteness, and vice versa. For example, in figure 3 the existence of a model in which A equals S implies that it cannot be proved that Spain and the region affected by the anticyclone are different.

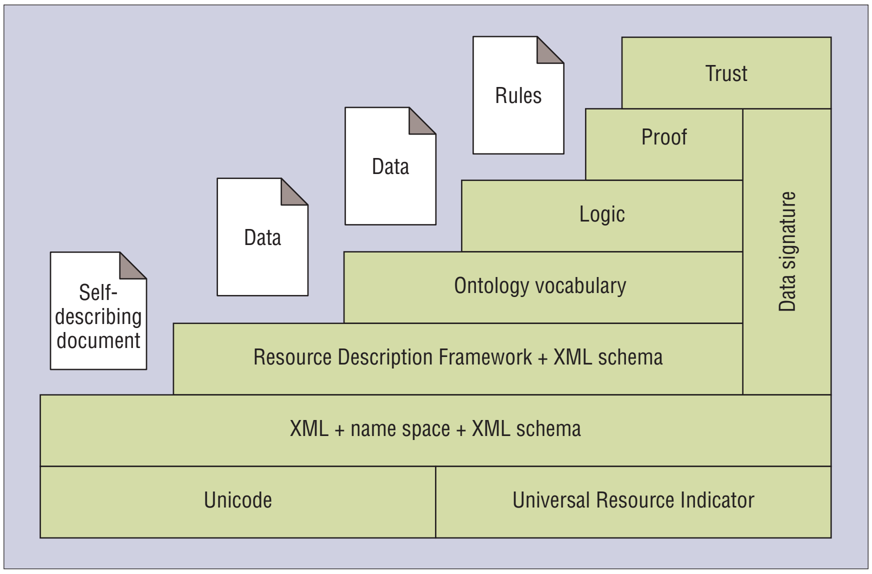
We can combine model theory with other features, especially of a linguistic nature. In this way, we can specify nonlogical information. However, the number of linguistic and lexicographic problems that this option supposes warns us to carefully explore this combination. On the other hand, it's not strange to come up against inconsistent information in a KDB on the Web, and in this case, model theory is useless: there are no models.

How can we reason with inconsistent information?

The larger the KDB, the less likely it will be consistent. Logical inconsistency, one of the main sources of distrust, is frequent whenever a KDB has a great deal of custom information (and whenever it's a relatively interesting part of the Web). Model search (consistency-checking) systems can't handle big KDBs. Thus, we only have to partially handle consistency analysis: if the analysis refutes the KDB, it will be inconsistent. So, inconsistency turns out to be the main anomaly (see the sidebar "The Main Logical Anomaly: Inconsistency").

If we accept having to work with inconsistencies, the goal is to design logical formalisms that limit the inconsistencies that can be inferred from an inconsistent KDB. One approach to acquiring trustworthy information is to express in *arguments* the information extracted from data.

An argument is simply a pair < S; fact > in which S is a subset of the KDB that proves a particular fact. We can consider an argument with an unacceptable conclusion to be a report about an anomaly, and the cleaner must find the reason for it. Any anomaly warns us about a mistake, but the expert decides which type of arguments to analyze. However, the ARS offers more arguments than human analysis can study. Although a referee layer for ARS's output could partially solve the overspill of arguments, we must design efficient criteria to discard the uninteresting ones, but this isn't easy. There's an interesting hierarchy of arguments (based on such first-order-logic notions as consistency and entailment) that estimates, at least theoretically, each argument's robustness.[14]



Trust
Rules
Proof
Data
Data
Logic
Data signature
Self-describing document
Ontology vocabulary
Resource Description Framework + XML schema
XML + name space + XML schema
Unicode
Universal Resource Indicator

Figure 2. The Semantic Web "cake" proposed by Tim Berners-Lee and colleagues, in which logic and proof are the bases beneath trust.

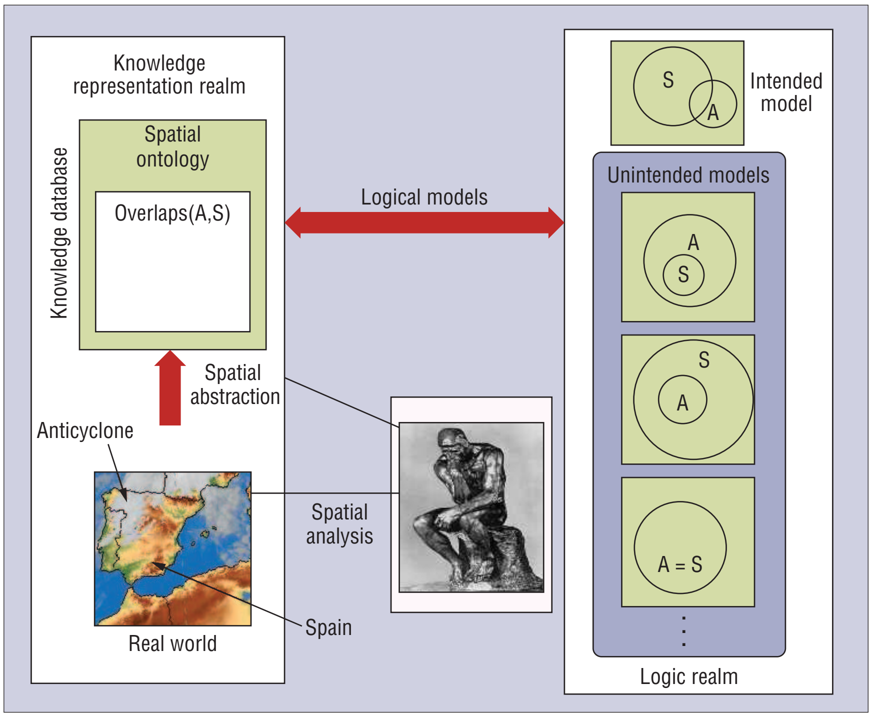


Figure 3. Unintended models for a poor knowledge database (with the natural notion of an Overlaps role).

Is first-order logic the universal provider for formal semantics?

First-order logic provides a formal KDB verification framework in which we can define some anomalies in specifications. We can actually verify and revise ontology languages such as DAML+OIL (www.w3.org/TR/daml+oil-reference) by translating

The Main Logical Anomaly: Inconsistency

The main drawback of many inconsistency-handling methods is their computational complexity. Additionally, the relevant role of ontologies in knowledge database (KDB) design for the Semantic Web requires a revision of both classical database semantics (which identify correct answers and logical consequences) and the definition of consistency itself (focused on integrity constraint checking). Following are some recent proposals for inconsistency handling.

Paraconsistent logics

This solution attempts to control the undesirable (logical) consequences we obtain from an inconsistent database.[1] The idea is to design inference rules that do not allow inconsistent knowledge, even though the KDB might be inconsistent.

Nonrepairing methods

These preserve the information source and decide trueness by analyzing the retrieved knowledge. Here are some recent examples from the literature:

- Establish a preorder on information sets that estimates their plausibility. In this way, deduction procedures are only applied on the KDB's belief subsets that have been selected according to the preorder.[2]
- *Argument* hierarchies classify arguments according to their robustness with respect to other KDB subsets[3] or their relationships with other arguments (for example, the argumentative framework based on the defeating relationship[4]). See elsewhere[5] for an application to databases.
- Researchers have used *contextualization*, a classic idea in artificial intelligence, to contextualize ontologies and data.[6]

Merging-oriented techniques

To handle inconsistencies through merging, we can

- design rules that enable the consistent fusion of jointly inconsistent information,[7] or
- develop solutions provided by the theory of merging databases.[8]

Measuring anomalies

Another option is to estimate the anomaly. For example, we can do the following:

- Evaluate by means of paraconsistent logics.[9]
- Measure inconsistent information by means of *measures for semantic information*, which estimate the information that the KDB gives on the model that it represents. Measures exist for working with inconsistencies.[10]

Repairing techniques

Repairing KDBs has two main drawbacks: the complexity of revising the full KDB, and the possible discarding of inconsistent but potentially useful data. So, it seems advisable to focus repair on relatively small data sets. The revision of ontologies is essentially different, because these represent a key organization of the owner's knowledge, and minor changes can produce unexpected and dangerous anomalies. Solutions include the following:

- The Fellegi-Holt method, which many government agencies have used in the past, is a safe logical method.[11] It's based on searching all deducible requirements to decide which fields to change to correct an incorrect record.
- Once we've defined the notion of database repair in a logical form, we can simulate the definition's entailment component using calculus—for example, through the tableaux method.[12]
- We can use consistent querying to repair databases: the answer itself drives the repair. For example, by splitting the integrity constraints according to the character of the negation involving each constraint, we can produce consistent answers and repair the database.[13]
- We can enforce database consistency. The aim is to systematically modify the database to satisfy concrete integrity con-

specifications into first-order logic and then applying an ATP.[15] Researchers have also investigated reasoning services for this class of languages on the basis of their relationship with description logics (a subset of first-order logic).[16] We select first-order logic as a translation target because this gives us strong reasoning methods for several sublogics that expand the ontology language. In addition, we have an ideal framework in which to expand the expressivity of the language itself. Others have proposed extensions of the first-order logic language (such as F-logic,[17] which naturally solves problems such as reification, a feature of RDF).

Modal logics for beliefs and knowledge enrich the representation and reasoning with mental attitudes. Mental attitudes don't seem adequate to be added to ontology reasoning, but other intentional operators that are used in logic programming for metareasoning (such as Provable from Ontology O) might be interesting to consider. But how?

For the time being, we tend to think of ontologies as static theories—that is, a set of axioms, separate from reasoning (or rules). This option reduces the problem to investigating which rule language is the best for each purpose.[18] But it doesn't seem adequate for verifying a KDB. Solving this limitation for KDB cleaning tasks is advisable, and not only by means of external reasoners. An ontology should ontologically define its reasoning framework, or better, the ontology itself should be a potential reasoner. We'll return to this question, and solution, later.

We carried out our data-cleaning experiments in the field of qualitative spatial reasoning, where ontology design itself is a challenge. We used first-order logic as a language so that we could focus on foundational rather than representational issues. We also used Otter (www-unix.mcs.anl.gov/AR/otter), a powerful ATP based on resolution.[19] Figure 4 shows a specific-purpose cleaning process. In the figure, to represent real-world qualitative information, the ontology expert determines the key spatial concepts and roles. Choosing the language induces a first qualitative representation—a graph, in this case—of the real model. The KDB associated with the representation, along with the spatial ontology, is cleaned by means of a cleaning agent.

Is there an ontology for untrustworthy information?

As we mentioned earlier, an argument's existence enables us to locate and repair an anomaly—if the argument points to one—by means of expert analysis of its content. A deep straints. A promising method uses greatest consistent specialization (adapting the general method, which might be undecidable).[14]

Consistent answering without repairing

We can also transform the query to obtain consistent answers or use paraconsistent inference.[15]

Preserving consistency

The idea here is to update the KDB preserving its consistency (in a wide sense, not only the satisfaction of integrity constraints). We also must prove the method's consistency and some level of completeness.[16]

References

1. J. Grant and V.S. Subrahmanian, "Applications of Paraconsistency in Data and Knowledge Bases," *Synthese,* vol. 125, no.1, 2000, pp. 121–132.
2. P. Marquis and N. Porquet, "Resource-Bounded Paraconsistent Inference," *Annals of Mathematics and Artificial Intelligence,* vol. 39, no. 4, 2003, pp. 349–384.
3. M. Elvang-Goransson and A. Hunter, "Argumentative Logics: Reasoning with Classically Inconsistent Information," *Data and Knowledge Eng.,* vol. 16, no. 2, 1995, pp. 125–145.
4. P.M. Dung, "On the Acceptability of Arguments and Its Fundamental Role in Nonmonotonic Reasoning, Logic Programming and n-Person Games," *Artificial Intelligence*, vol. 77, 1995, pp. 321–358.
5. S. Pradhan, "Connecting Databases with Argumentation," *Proc. 14th Int'l Conf. Applications of Prolog* (INAP 2001), LNCS 2543, Springer, 2001, pp. 170–185.
6. P. Bouquet et al., "C-OWL: Contextualizing Ontologies," *Proc. 2nd Int'l Semantic Web Conf.* (ISWC 03), LNCS 2870, D. Fensel et al., eds., Springer, 2003, pp. 164–179.
7. I. Bloch et al., "Fusion: General Concepts and Characteristics," *Int'l J. Intelligent Systems*, vol. 16, 2001, pp. 1107–1134.
8. L. Cholvy and S. Moral, "Merging Databases: Problems and Examples," *Int'l J. Intelligent Systems*, vol. 16, no. 10, 2001, pp. 1193–1221.
9. A. Hunter, "Evaluating the Significance of Inconsistencies," *Proc. Int'l Joint Conf. Artificial Intelligence* (IJCAI 03), Morgan Kaufmann, 2003, pp. 468–473.
10. K.M. Knight, "Two Information Measures for Inconsistent Sets," *J. Logic, Language, and Information*, vol. 12, no. 3, 2003, pp. 227–248.
11. A. Boskovitz, R. Goré, and M. Hegland, "A Logical Formalisation of the Fellegi-Holt Method of Data Cleaning," *Proc. 5th Int'l Symp. Intelligent Data Analysis* (IDA 2003), LNCS 2810, M. Berthold et al., eds., Springer, 2003, pp. 554–565.
12. L.E. Bertossi and C. Schwind, "Database Repairs and Analytic Tableaux," *Annals of Mathematics and Artificial Intelligence*, vol. 40, nos. 1-2, 2004, pp. 5–35.
13. S. Greco and E. Zumpano, "Querying Inconsistent Databases," *Proc. 7th Int'l Conf. Logic for Programming and Automated Reasoning* (LPAR 00), LNCS 1955, M. Parigot and A. Voronkov, eds., Springer, 2000, pp. 308–325.
14. S. Link, "Consistency Enforcement in Databases," *Proc. 2nd Int'l Workshop Semantics in Databases*, LNCS 2582, L. Bertossi et al., eds., Springer, 2003, pp. 139–159.
15. P. Marquis and N. Porquet, "Resource-Bounded Paraconsistent Inference," *Annals of Mathematics and Artificial Intelligence,* vol. 39, no. 4, 2003, pp. 349–384.
16. E. Mayol and E. Teniente, "Consistency Preserving Updates in Deductive Databases," *Data and Knowledge Eng.*, vol. 47, no. 1, 2003, pp. 61–103.

analysis of the arguments reported by an ARS classifies them according to their trustworthiness. So, an argument hierarchy is a first step toward an ontology of trust. But we also need to clarify the ARS assistant's operational features. The autonomous behavior that we expect of the automated argument searcher can be slanted. Researchers could learn much more about this topic using the automated-reasoning field as background.

The task of recognizing fraudulent information will be arduous; it represents a different problem. If an ontology represents a fraudulent world (an unintended model of the user's knowledge, which is particularly dangerous), the arguments reporting fraudulent information don't show anomalies. To make matters worse, fraudulent ontologies might use linguistic features to hide information. This takes the question beyond our interest: fraud, by definition, can't have logical legitimacy. One solution might be to combine users' trustworthiness in other users.[20] We might also be able to solve the problem of recognizing fraudulent information with a precise analysis of mental attitudes such as *beliefs*, *desires*, or *intentions* to specify the phenomenon in the case of data-mining agents.

Adding mental attitudes to data mining and cleaning

In a scenario with inconsistent information or several cleaning agents at work, is it necessary to add an intentional level to the logical knowledge? We've already commented that adding mental attitudes to the ontology isn't advisable. But in a multiagent-system setting, the specification of knowledge extracted from inconsistent information is unavoidable. In fact, the specifications of agent communication languages such as FIPA-ACL (www.fipa.org/specs/fipa00037/SC00037J.pdf) use mental attitudes. A multiagent data-mining system should manage this kind of information.

To avoid adopting mental attitudes when you work with simple information items, you might be able to design fusion rules based on the nature of information.[21] This option might need to preprocess data from heterogeneous databases. Also, you must extract and translate data (through ontology mapping) during cleaning runtime, making it difficult to achieve acceptable response times, as occurs in classical data cleaning. But the preprocessing in KDB cleaning has another purpose: it's possible for the system to trigger some rules to add new data, transforming the KDB's data source into a consistent instance of the KDB target.

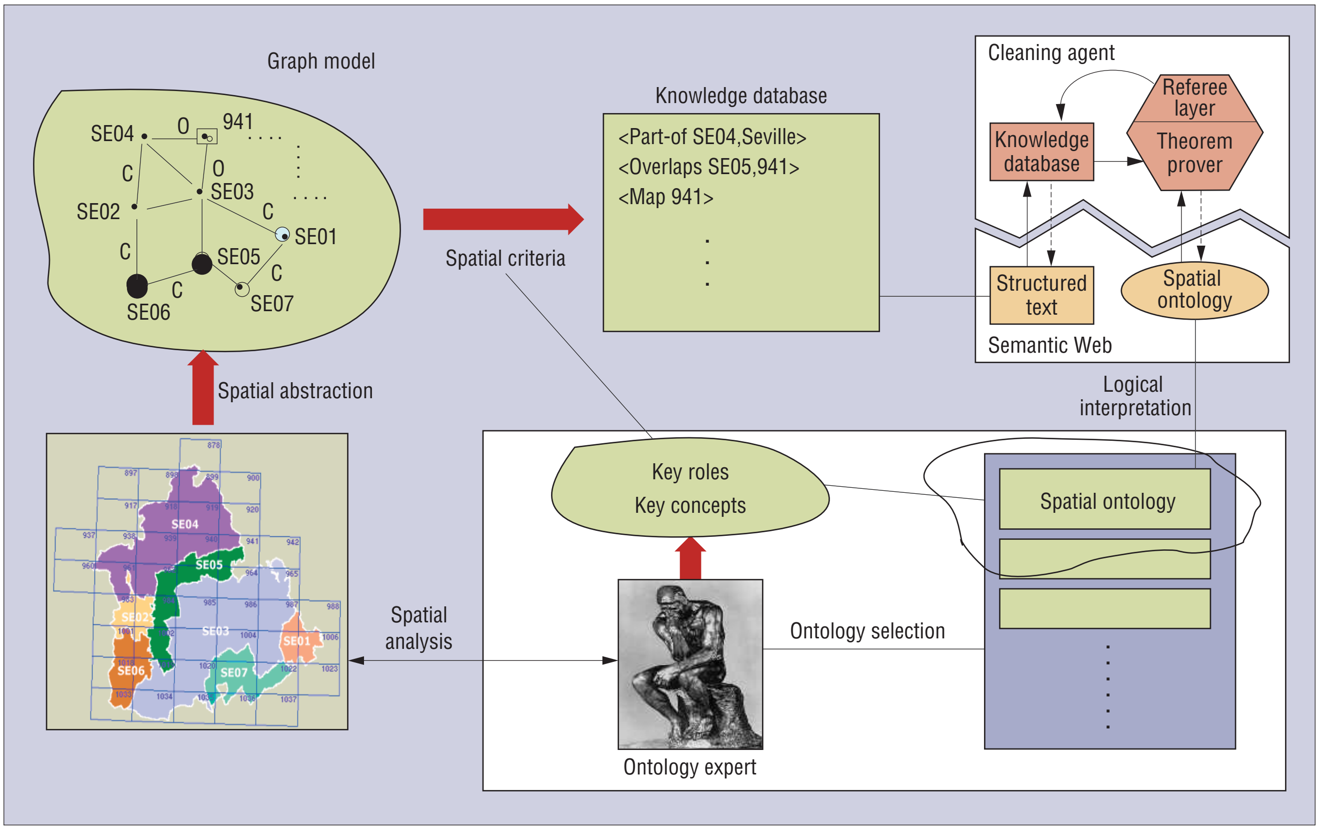


Figure 4. Ideal scenario for a specific-purpose cleaning agent.

Is it possible to preprocess data with respect to an ontology?

An interesting aspect of the cleaning process appears when KDB programmers decide which of an ontology's main roles and concepts they want to focus the cleaning on (the spatial-abstraction step in figure 4). Does Skolem noise mean that the programmers don't know the ontology that the KDB builder is implicitly or unconsciously using? Generally, no. It might only suggest the need to preprocess the knowledge, sometimes applying simple rules associated with the ontology. One option in preprocessing full databases is to extend them with new data produced by applying simple rules, designed from the ontology, that enable the programmers to achieve consistent translations.[1]

A proposal: Extend OWL to ROWL

We hope to solve some of these challenges by designing a language, which we might call ROWL (a Reasonable Ontology Web Language). We don't describe a formal language here; we'll just indicate what that language might look like in its early development and how it could satisfy some of the challenges. The idea stems from the need to attach (specialized) certified logical inference to an ontology—that is, ontological information about how to reason with information. To do this, we should add new features to OWL to specify what type of reasoning and which ARS we need to work with the OWL ontology (also keeping in mind that the ontology is optimized to reason with these new features).

The ROWL design

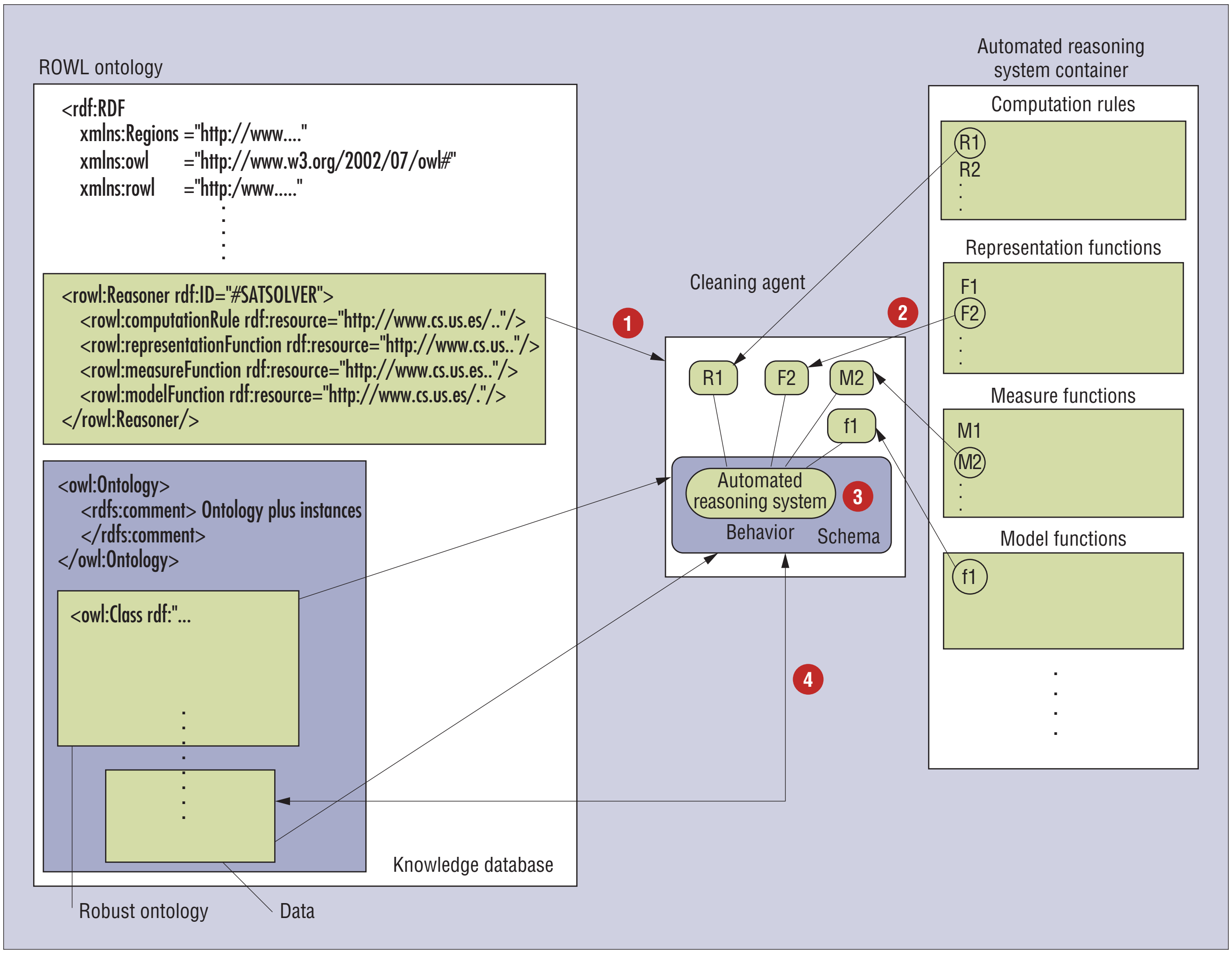
The idea revolves around a *certified generic framework*, which simplifies the design of a certified ad hoc reasoner to provide ATP features that the ontology must supply.[22] In other words, ROWL is OWL plus a CGF. An ontology that accepts a reasoner fitting into this framework must

- specify simple computation rules (to drive the deduction) and simple representation rules (to normalize formulas),
- measure functions on formulas (to prove the halting of deduction methods), and
- model functions (to supply models in some basic cases).

The CGF was programmed in ACL2 (www.cs.utexas.edu/users/moore/acl2), which is both a programming language in which you can model computer systems and a tool to help you prove a model's properties. The ATP obtained is thus sound, complete, and formally verified by ACL2.

The CGF framework[22]—which can be executed because it produces a Common Lisp program—can synthesize satisfiability (SAT) provers, the most important component in many automated-reasoning-based AI systems. (Researchers are still searching for a complete, truly polynomial-time Boolean SAT solver, or a proof that such an algorithm does not exist. However, several ways to solve satisfiability in polynomial time for practical purposes exist.) However, these synthesized SAT provers currently perform far worse than any state-of-the-art SAT prover. Obtaining better provers using more efficient data structures is a future aim.

ROWL should have features that show links to sources where the elements that specify the reasoning with the ontology (measurement, computation rules, and so on) are defined, letting users refine or change the



ROWL ontology
<rdf:RDF
xmlns:Regions ="http://www...."
xmlns:owl ="http://www.w3.org/2002/07/owl#"
xmlns:rowl ="http:/www....."
<rowl:Reasoner rdf:ID="#SATSOLVER">
<rowl:computationRule rdf:resource="http://www.cs.us.es/.."/>
<rowl:representationFunction rdf:resource="http://www.cs.us.."/>
<rowl:measureFunction rdf:resource="http://www.cs.us.es.."/>
<rowl:modelFunction rdf:resource="http://www.cs.us.es/."/>
</rowl:Reasoner/>
<owl:Ontology>
<rdfs:comment> Ontology plus instances
</rdfs:comment>
</owl:Ontology>
<owl:Class rdf:"...
Knowledge database
Robust ontology
Data
Cleaning agent
R1
F2
M2
f1
Automated reasoning system
Behavior
Schema
1
2
3
4
Automated reasoning system container
Computation rules
R1
R2
Representation functions
F1
F2
Measure functions
M1
M2
Model functions
f1

Figure 5. Our proposed general-purpose logic-based agent.

deduction method by slightly changing the KDB.

Generalizing this framework to other ARSs will be interesting. Of course, before determining ROWL's key features, we must design an automated-reasoning ontology. Projects to build such an ontology already exist—for example, MathBroker (www.risc.uni-linz.ac.at/projects/basic/mathbroker) and MONET (http://monet.nag.co.uk/cocoon/monet/index.html).

ROWL's advantages

The definitive advantage of a language such as ROWL is that it would let us design general-purpose cleaning agents as opposed to specific-purpose ones based on wrapper technology. Such agents would adopt both the ATP synthesized from the ROWL ontology's information as a component, and the agent's behavior scheme, where modules for repairing anomalies can be implemented (see figure 5).

This proposal is only the first step—much research remains to be done. First, we must perform a complete analysis of the relationships between ACL2 logic (a computational logic closer to a quantifier-free first-order logic accepting some amount of induction) and description logics. But how does this proposal solve any of the challenges we outlined earlier? Basically, it does so by setting standards to attach additional information to the OWL ontology:

- If we assume we're working with a poor ontology (the second challenge), we can request other explicit, specified ROWL features with links to explicit information about some of the anomalies the ARS would find.
- Ontology mapping (the fourth challenge) can be certified to some extent by the ARS associated with the ontology target: trust is augmented if the ARS certifies the translation of essential properties into concepts of ontology source.

In the concrete example of ACL2, exploiting the *certified arithmetic reasoning* embedded in the ACL2 system itself would be extremely interesting: we could add arithmetic reasoning about data to OWL (through logical interpretation in ACL2 logic).

- Evidence of the existence of undesirable models (the fifth challenge) may be their unprovability using the recommended ARS of basic theorems on ontology concepts.
- If we as ontology designers think that inconsistencies will probably appear (the sixth challenge), we can associate, in the earlier stages of ontology evolution, a reasoning model based on arguments. After that, we could use a standard ARS, chang-

```
<rowl:Reasoner
  rdf:ID="#Davis-Putnam-SATSOLVER">
<rowl:ComputationRule
  rdf:resource="#DP-comp-rule"/>
<rowl:RepresentationFunction
  rdf:resource="#DP-repr"/>
<rowl:MeasureFunction
  rdf:resource="#DP-measure"/>
<rowl:ModelFunction
  rdf:resource="#DP-models"/>
</rowl:Reasoner>

<rowl:Assumption
  rdf:resource="#Unique-names-principle"/>
<rowl:Assumption
  rdf:resource="#Domain-closure-principle"/>
```
(a)

```
<owl:Class rdf:ID="Herbivore">
  <rdfs:Comment> herbivores are exactly those animals that eat some
  plant</rdfs:Comment>
    <owl:equivalentClass>
      <owl:intersectionOf rdf:parsetype="Collection">
        <owl:Class rdf:about="animal">
          <owl:Restriction>
            <owl:onProperty rdf:resource=#eats"/>
            <owl:someValuesFrom rdf:resource=#plant"/>
          </owl:Restriction>
      </owl:intersectionOf>
    </owl:equivalentClass>
</owl:Class>

<owl:Class rdf:ID="Carnivore">
    <rdfs:Comment>carnivores are animals that are not herbivores and they eat
    animals </rdfs:Comment>
      <owl:disjointWith rdf:resource="#Herbivore">
      <owl:intersectionOf rdf:parsetype="Collection">
        <owl:Class rdf:about="animal">
        <owl:Restriction>
          <owl:onProperty rdf:resource=#eats"/>
          <owl:someValuesFrom rdf:resource="#animals">
        </owl:Restriction>
      </owl:intersectionOf>
</owl:Class>

<Carnivore rdf:ID="Tarzan">
<eats rdf:resource="Pumbaa">
<eats rdf:resource="potato">
</Carnivore>

<Animal rdf:ID="Pumbaa">
<eats rdf:resource="potato">
</Animal>

<Plant rdf:ID="potato">
```
(b)

```
(<-> carnivore-potato
(& (& (- herbivore-potato) animal-potato)
(/ (& eats-potato-potato animal-potato)
(/ (& eats-potato-pumbaa animal-pumbaa)
(& eats-potato-tarzan animal-tarzan)))))

(<-> carnivore-pumbaa
(& (& (- herbivore-pumbaa) animal-pumbaa)
(/ (& eats-pumbaa-potato animal-potato)
(/ (& eats-pumbaa-pumbaa animal-pumbaa)
(& eats-pumbaa-tarzan animal-tarzan))))

(<-> carnivore-tarzan
(& (& (- herbivore-tarzan) animal-tarzan)
(/ (& eats-tarzan-potato animal-potato)
(/ (& eats-tarzan-pumbaa animal-pumbaa)
(& eats-tarzan-tarzan animal-tarzan)))

(<-> herbivore-potato
(& animal-potato
(/ (& eats-potato-potato plant-potato)
(/ (& eats-potato-pumbaa plant-pumbaa)
(& eats-potato-tarzan plant-tarzan))))

(<-> herbivore-pumbaa
(& animal-pumbaa
(/ (& eats-pumbaa-potato plant-potato)
(/ (& eats-pumbaa-pumbaa plant-pumbaa)
(& eats-pumbaa-tarzan plant-tarzan))))

(<-> herbivore-tarzan
(& animal-tarzan
(/ (& eats-tarzan-potato plant-potato)
(/ (& eats-tarzan-pumbaa plant-pumbaa)
(& eats-tarzan-tarzan plant-tarzan))))

plant-potato
eats-pumbaa-potato
eats-tarzan-pumbaa
eats-tarzan-potato
herbivore-pumbaa
carnivore-tarzan
```
(c)

Figure 6. A Reasonable Ontology Web Language (ROWL) ontology: (a) the four ingredients needed for synthesizing an SAT solver and two classic database assumptions, the unique-names principle and the domain closure axiom; (b) an OWL ontology; and (c) propositional transformation by binding variables.

ing only the corresponding links in the ROWL ontology. In this way, the ontology will benefit from state-of-the-art certified-reasoning models.

This process will certify the framework. Thus, the ATP embedded in the cleaning agent will be correct and won't have anomalous behavior.

A ROWL ontology cleaning session

Let's see how a cleaning session (for consistency analysis) on a ROWL ontology would run. Consider the ROWL ontology in figure 6 (Tarzan is a carnivore *and* an herbivore because he satisfies both definitions. But by definition, this isn't possible.) Figure 6a shows the four ingredients needed for synthesizing an SAT solver (in this case, it synthesizes the classic Davis-Putnam-Loveland-

Longeman procedure for SAT and two classic database assumptions: the *unique-names principle* and the *domain closure axiom*. Figure 6b shows an OWL ontology. Figure 6c shows the ontology's transformation into a set of propositional formulas through the binding of variables.

To synthesize and verify the SAT solver, the CGF essentially takes as input the elements defined in figure 6a; its runtime is 0.28 seconds. Finally, the cleaning agent must only check satisfiability. In this case, it outputs the following:

```
(generic-sat-davisputnam
(build-rowl-example1
(potato pumbaa tarzan)))
; real time : 0.020 secs
; run time : 0.020 secs
; NIL
```

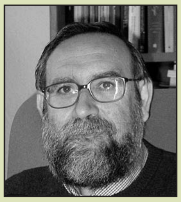
Although the challenges we've discussed represent only a partial view, they affect every phase of typical data cleaning: data analysis, definition of mapping rules, verification, transformation, and backflow of cleaning data.[11] From these foundational issues, we can derive many of the problems in real-life cleaning scenarios.

Many database designers might not celebrate the marriage of data and ontology, and some kind of classical data cleaning, combined with automated-reasoning engineering, could remain the sole option. To sum up, Semantic Web cleaning agents raise a series of challenges that could overwhelm logic-based agents and classical data-cleaning techniques if they are not combined and implemented together.

Acknowledgments

This work is partially supported by the Ministry of Education and Science project TIN2004-03884, which is cofinanced by FEDER funds (European Union funds for regional development).

The Authors



José A. Alonso-Jiménez is an associate professor in the University of Seville's Computer Science and Artificial Intelligence Department and head of the Computational Logic Group (www.cs.us.es/glc). His research interests include computational logic, formal methods, and the verification of intelligent systems. He received his PhD in mathematics from the University of Seville. Contact him at E.T.S. Ingeniería Informática, Universidad de Sevilla, Avda. Reina Mercedes s/n, 41012 Sevilla, Spain; jalonso@us.es.

Joaquín Borrego-Díaz is an associate professor in the University of Seville's Computer Science and Artificial Intelligence Department. His research focuses on computational logic and automated reasoning and their application to knowledge representation and reasoning, especially the intersection of automated-reasoning systems and the Semantic Web. He received his PhD in mathematics from the University of Seville. Contact him at Departamento de Ciencias de la Computación e Inteligencia Artificial, ETS Ingeniería Informática, Avda. Reina Mercedes s/n, 41012 Sevilla, Spain; jborrego@us.es.

Antonia M. Chávez-González is an assistant professor in the University of Seville's Computer Science and Artificial Intelligence Department. Her research interests include automated theorem proving and intelligent data cleaning. She received her PhD in mathematics from the University of Seville. Contact her at Departamento de Ciencias de la Computación e Inteligencia Artificial, ETS Ingeniería Informática, Avda. Reina Mercedes s.n., 41012 Sevilla, Spain; tchavez@us.es.

Francisco J. Martín-Mateos is an assistant professor in the University of Seville's Computer Science and Artificial Intelligence Department. His research interests include formal methods applied to verification and automated deduction. He received his PhD in mathematics from the University of Seville. Contact him at E.T.S. Ingeniería Informática, Universidad de Sevilla, Avda. Reina Mercedes s/n, 41012 Sevilla, Spain; fjesus@us.es.

References

1. J. A. Alonso-Jiménez et al., "Towards a Practical Argumentative Reasoning with Qualitative Spatial Databases," *Proc. 16th Int'l Conf. Industrial and Eng. Applications of Artificial Intelligence and Expert Systems* (IEA/AIE 03), LNAI 2718, P.W. Chung et al., eds., Springer, 2003, pp. 789–798.
2. T.J.M. Bench-Capon, "The Role of Ontologies in the Verification and Validation of Knowledge-Based Systems," *Int'l J. Intelligent Systems*, vol. 16, no. 3, 2001, pp. 377–390.
3. J. Denzinger and I. Dahn, "Cooperating Theorem Provers," *Automated Deduction: A Basis for Applications*, W. Bibel and P.H. Schmitt, eds., Kluwer Academic Publishers, 1998, pp. 383–416.
4. A. Gómez-Pérez, "Evaluation of Ontologies," *Int'l J. Intelligent Systems,* vol. 16, no. 3, 2001, pp. 391–409.
5. N. Guarino and C.A. Welty, "Evaluating Ontological Decisions with OntoClean," *Comm. ACM*, vol. 45, no. 2, 2002, pp. 61–65.
6. J. Heflin, *Towards the Semantic Web: Knowledge Representation in a Dynamic, Distributed Environment*, doctoral dissertation, Computer Science Dept., Univ. of Maryland, College Park, 2001.
7. A. Maedche et al., "Ontologies for Enterprise Knowledge Management," *IEEE Intelligent Systems,* vol. 18, no. 2, 2003, pp. 26–33.
8. G. Antoniou and A. Kehagias, "On the Refinement of Ontologies," *Int'l J. Intelligent Systems*, vol. 15, no. 7, 2000, pp. 623–632.
9. M.C.A. Klein and D. Fensel, "Ontology Versioning on the Semantic Web," *Proc. 1st Semantic Web Symp.* (SWWS 01), Stanford Univ. Press, 2001, pp. 75–91; www.few.vu.nl/~mcaklein/papers/SWWS01.pdf.
10. A. Maedche, B. Motik, and L. Stojanovic, "Managing Multiple and Distributed Ontologies on the Semantic Web," *Very Large Database J.*, vol. 12, no. 4, 2003, pp. 286–302.
11. E. Rahm and H.-H. Do, "Data Cleaning: Problems and Current Approaches," *IEEE Data Eng. Bull.*, vol. 23, no. 4, 2000, pp. 3–13.
12. P. Bouquet et al., "C-OWL: Contextualizing Ontologies," *Proc. 2nd Int'l Semantic Web Conf.* (ISWC 03), LNCS 2870, D. Fensel et al., eds., Springer, 2003, pp. 164–179.

13. J. Hendler, "Agents and the Semantic Web," *IEEE Intelligent Systems,* vol. 16, no. 2, 2001, pp. 30–37.

14. M. Elvang-Goransson and A. Hunter, "Argumentative Logics: Reasoning with Classically Inconsistent Information," *Data and Knowledge Eng.,* vol. 16, no. 2, 1995, pp. 125–145.

15. R. Fikes, D.L. McGuinness, and R. Waldinger, *A First-Order Logic Semantics for Semantic Web Markup Languages*, tech. report KSL-02-01, Knowledge Systems Laboratory, Stanford Univ., 2002.

16. I. Horrocks and P.F. Patel-Schneider, "Reducing OWL Entailment to Description Logics Satisfiability," *Proc. 2nd Int'l Semantic Web Conf.* (ISWC 03), LNCS 2870, D. Fensel et al., eds., Springer, 2003, pp. 17–29.

17. G. Yang and M. Kifer, "Reasoning about Anonymous Resources and Meta Statements on the Semantic Web," *J. Data Semantics*, vol. 1, LNCS 2800, 2003, pp. 69–97.

18. I. Horrocks et al., "Where Are the Rules?" *IEEE Intelligent Systems*, vol. 18, no. 5, 2003, pp. 76–83.

19. J.A. Alonso-Jiménez et al., "A Methodology for the Computer–Aided Cleaning of Complex Knowledge Databases," *Proc. IEEE Int'l Conf. Industrial Electronics, Control, and Instrumentation* (IECON 02), IEEE CS Press, 2003, pp. 1806–1812.

20. M. Richardson, R. Agrawal, and P. Domingos, "Trust Management for the Semantic Web," *Proc. 2nd Int'l Semantic Web Conf.* (ISWC 03), LNCS 2870, D. Fensel et al., eds., Springer, 2003, pp. 351–368.

21. I. Bloch et al., "Fusion: General Concepts and Characteristics," *Int'l J. Intelligent Systems*, vol. 16, 2001, pp. 1107–1134.

22. F.J. Martín-Mateos et al., "Formal Verification of a Generic Framework to Synthesize SAT Provers," *J. Automated Reasoning*, vol. 32, no. 4, 2004, pp. 287–313.

For more information on this or any other computing topic, please visit our Digital Library at www.computer.org/publications/dlib.