A Methodology for the Computer–Aided Cleaning of Complex Knowledge Databases*

José A. Alonso-Jiménez, Joaquín Borrego–Díaz, Antonia M. Chávez-González
Miguel A. Gutiérrez-Naranjo and Jorge D. Navarro-Marín
Dept. of Computer Science and Artificial Intelligence
University of Sevilla
Avda. de Reina Mercedes s.n., 41012–Sevilla. Spain.
E–mail: *tchavez@us.es*

Abstract– **In environments with complex cognitive structure (as the Semantic Web or sophisticated spatial databases for Geographical Information Systems), classical methods for detecting anomalies can be inadequate. In this paper the use of an automated theorem prover to detect anomalies in knowledge bases within a complex ontology is proposed. The authors argue that it will need to integrate such systems in some intelligent agents. The loss of real-time execution -in some cases- is discussed with examples.**

1 Introduction

In the Knowledge Engineering Field, the verification of knowledge bases occupies a significant position. The verification task is a complex and, in general, unsolvable problem. The advent of environments with complex *cognitive* structure (as the Semantic Web) must to induce to a revision of some of the classical solving methods, mainly the agent-oriented ones. Many of them are designed for time constrained environments. A new aim in the AI field will be the re-balancing of reactive and proactive attitudes in the design of systems known as *intelligent agents*. Both problems (verification of knowledge bases and the balance remarked above) are combined in the task of the verification, by intelligent agents, of Web metadata. Some problems will demand more quality of information losing pure reactive answers. The deliberative agents (agents with logic-based behaviour, produced by a deductive process) will be free of *tight* real-time constraints in tasks as the verification of formal ontologies, information retrieval in a structured form (in a *logical theory* form), semantic web mining, etc. A key environment is the Semantic Web, which changes the notion of perception (or message): the agent will receive from its environment *more complex stimuli* [1]. This issue becomes more important when the number of (heterogeneous) interacting systems increases (as well the number of different ontologies).

*Partially supported by the MCyT project TIC 2000-1368-C03-02 and the project TIC-137 of *Plan Andaluz de Investigación*.

We propose a methodology in order to verify Knowledge Databases with the assistance of an automated theorem prover (a.t.p.). We believe that this methodology must be a first step in the design of *intelligent cleaning agents* for the Semantic Web.

Let us a few remarks about the organization of the paper. In next section a brief description of the notion of *intelligent agent* is given, as well as a discussion about the problem of the real-time requirement for deliberative agents. Our main purpose is to present a methodology to verify Knowledge databases with complex domain knowledge (section 4). Finally, the methodology is illustrated with a case study (section 5).

2 Intelligent Agents

There are many definitions of *intelligent agent*, but there exists a consensus in which are the essential requirements in order to consider a system as intelligent agent:

1. *Reactivity:* it is able to respond *effectively* to the perceptions received from its environment.
2. *Pro-activeness:* it is able to exhibit a behaviour driven to reach its objectives.
3. *Sociability:* it is able to communicate with other agents (in order to cooperate, to ask for information, etc.).

From the basis proposed in the above minimal definition many agent architectures emerge, and there exists an established ground theory to work with them [2]. A general and complete architecture for an agent is illustrated in the fig. 1 from [3].

We are interested here in *cognitive agents:* agents for which the knowledge processing — by deductive methods — is the heart of the behaviour. The *cognitive processing* is implemented in very different architectures, from implementations of logical theories of action and change up to sophisticated applications of logic programming. There are also many architectures with successful applications [4], some of them are based in philosophical theories of practical reasoning, as the Belief-Desire-Intention model.

0-7803-7474-6/02/$17.00 ©2002 IEEE

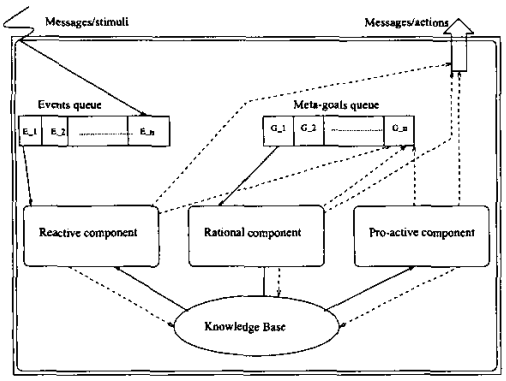

Figure 1: A general architecture of agent

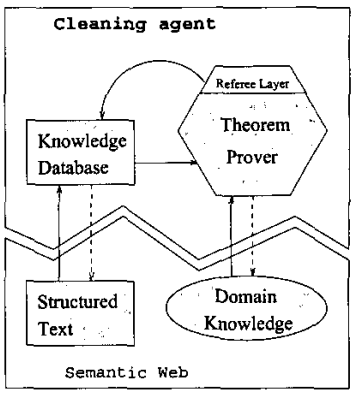

Figure 2: Cleaning service process

2.1 Real-time processing versus complex reasoning

The dilemma that entitles the subsection is essentially the same that of reactivity versus proactivity attitudes in the design of the agent. How can we escape from the real-time constraints paradigm (when it is necessary)? We are interested in three proposals:

- To think the balance between Symbolic representation and effective processing: there exists an almost complete classification of the complexity of many languages extending the classical Horn Logic (from Logic Programming paradigm), thus we can decide to extend the richness of the Knowledge Representation, even to extend the analysis to other more specific representations.
- Several automated theorem provers can be specialized as useful extensions of Logic Programming, and they can be the natural candidates for the *thinking component* of the agent.
- It can be advisable, in many cases, to substitute the real-time constraints by other notions of effectiveness.

All these aspects are relevant for the possible embedding of an automated theorem prover in a cognitive agent.

3 Embedding deductive attitudes in the agent

The ability of complex reasoning is possible if the agent has a powerful rational component. Some of the actions/messages produced by the agent will be consequence of a deductive process. Thus a natural option can be to integrate an a.t.p. as rational component. But a strict estimation of the *interest* of the stream data from the deductive component to the knowledge base (*referee layer*), or to the environment (*supervisor layer*) is needed. The reason is that an a.t.p can produce an *overspill* of new knowledge, not all of it interesting for the problem. Referee and supervisor layers are not a completely new idea in automated deduction [5], but in our case the referee is reduced to setting different flags with which the agent decides the interest of blocks of information, as well as the time complexity (steps of deduction) that the component will work on data. In general, the ratio of useless results is 1000:1 [5], thus, the referee component must be *strict* in order to accept new knowledge.

The main problem of the integration of an a.t.p. in the agent is that the behaviour of the deductive component is underspecified, because it is autonomous in the data processing. Thus we face a coarse specified system. Moreover, we can not constraint the behaviour of the components by real-time requirements and the symbolic representation is not limited to a concrete sublanguage of the full first order logic.

4 Towards Cleaning services in the Semantic Web

The Semantic Web is a Web of data that can be processed directly or indirectly by machines. Briefly, the architecture of Semantic Web is decomposed in several layers: the ground layers concern to the symbolic level of data and metadata and they are the eXtensible Markup Language XML and the Resource Data Framework RDF; the RDF language provides an uniform data model for representing metadata and it makes easy the processing of data by machines [6]; the Ontology layer prevents ambiguous meanings of *words*; at the top, logic layer is able to reasoning with the data and the Web Trust prevents inconsistency.

4.1 Description of the problem

The cleaning service concerns the optimization of ontologycal representation and detection of inconsistencies (not only in the knowledge domain, as in [7]), by comparing results derived by the agent with the own ontology or the domain knowledge, etc. (see fig. 2).

The problem is that reasoning with inconsistency in structured text will be needed. As we will remark in

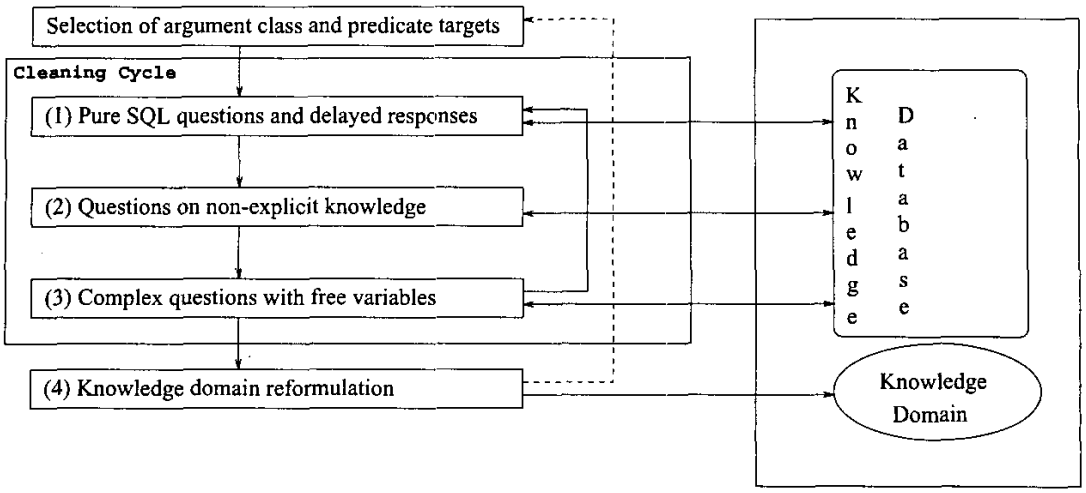

Figure 3: Scheme of cleaning process using an a.t.p.

the next section, this type of reasoning can be interpreted in the theorem prover. Due to the hardness of the verification of complex Knowledge Bases, this interpretation will be possible if the agent has delayed response behaviour.

The main drawback for the integration of an a.t.p. in the agent, the loss of real-time requirements, does not matter for this problem. We want a system which works as a *night cleaning* service (as a personal assistant): the system debugs the metadata included in the Semantic Web by the user, using the idle-time of the computer.

4.2 Anomalies in the database

The agent will find four main types of errors or anomalies (presented as arguments):

(A1): The contradictions of the base due to the bad implementation of data (for example, lack of data.

(A2): The anomalies due to the inconsistency of the model: the theorem prover derives from the database the existence of elements which have not a name (possibly because they have not yet been introduced by the user). This anomaly can also be due to the *Skolem's noise*, produced when we work with the domain closure axioms but the domain knowledge is not clausal. Thus, it is also possible that the agent can not obtain any answer to the requirements of another agent.

(A3): Disjunctive answers (a logical deficiency).

(A4): Inconsistency in Knowledge Domain.

The anomalies comes from several reasons, for example:

- The set of data is inconsistent with the Domain Knowledge due to formal inconsistencies produced by incorrect data.
- The database is never completed, that is, the user will keep on introducing data.

Technically, the absence of certain facts about a predicate implies deduction with the Knowledge Domain, and — due to above reasons — no-desired answers can be obtained.

4.3 Methodology

The methodology is shown in fig. 3. The cleaning cycle combines SQL (Structured Query Language) questions with others which can be asked by other agents, or by the user in order to complete the knowledge with other facts not explicit in the database (for example, facts about predicates not explicit yet in the data introduced by the user(s)). The step (4) is invoked when an anomaly of the Knowledge Domain is found.

5 A case study

We report an experiment made with the cleaning service, working on a database within a complex ontology. The example concerns the debugging of a Spatial Database, that we may suppose written in a structured text form. The structured text is a general and simple form of knowledge representation that should be easy to view as a subset of XML, for example.

Concretely, we will work with a database on the relationships between three types of regions: counties, districts, and available maps on Andalucía, a Spanish autonomous region (see fig. 4).

The ontology we use comes from a formalization of the relationships between regular regions of the space, the RCC theory [8]. Thus the knowledge domain is computationally complex. Several problems on spatial configurations via Constraint Satisfaction Problems — in the relational sublanguage of RCC — are extensively studied in Artificial Intelligence [9], as well as in the

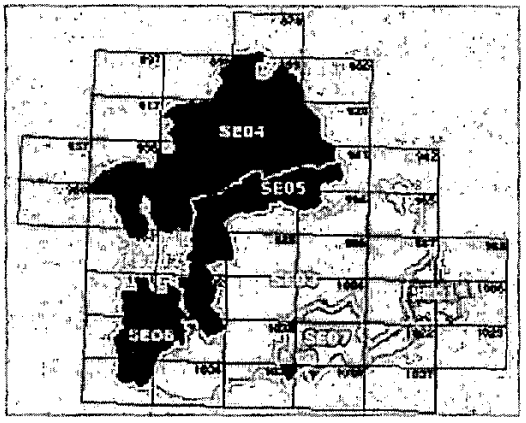

Figure 4: Partial view of the autonomous region

```
⟨Connect : SE04, SE05⟩
⟨Connect : SE04, Map − 941⟩
⟨Overlaps : Map − 920, SE04⟩
⟨Part − of : SE04, SEVILLA⟩
    ⋮
```

Figure 5: Some facts of the spatial database from fig.4

field of Geographic Information Systems [10]. However, the full theory is computationally unacceptable.

The system works on a data base built in base of the relationships of connection (`Connect`), nonempty-intersection (common subregion, `Overlaps`), and part-of (`Part-of`) (see fig. 5).

Therefore a lot of hidden information exists, a lot of knowledge with respect to other topological relations between regions, not explicit in the database, that the a.t.p. might to derive (and, eventually, to add to the database). The universe is formed by 260 regions, approximately, for which we have a data base with 34000 facts (included first-order formalization of databases, but the number can be reduced using some features of the theorem prover). This base has been made for a human, and possibly, it contains errors.

The complexity of the formal models of this type of reasoning makes necessary to work with inconsistent knowledge; therefore it is necessary to use a method of reasoning with inconsistencies, and we have selected the *argumentative reasoning* for working with them. We should define the arguments considering the automated theorem prover OTTER [11] as deductive component. An *argument* is a pair $\langle \Phi, \alpha \rangle$ such that Φ is a subset of database and $\Phi \vdash \alpha$. An O-argument (argument for OTTER) $\langle \Phi, \alpha \rangle$ is an argument such that α is *provable from* Φ *by* OTTER. Selecting different types of arguments it is posible to derive useful knowledge from inconsistent databases [12]. Using the answer literal `$Answer` of OTTER we can determine the argument in each experiment (see fig. 6).

It is not our aim to find only inconsistencies in the domain knowledge (A4). In [7] the authors show an application of an automated theorem prover (the SNARK system) to provide a declarative semantics for languages for the Semantic Web, by translating first the forms from the semantic markup languages to the first-order logic. The translation allows them to apply the theorem prover to find inconsistencies. Our problem is not exactly the same. We assume that the domain knowledge (the RCC theory and eventually the composition table for the eight relations) is consistent, and that it is highly possible that RCC jointly with the database becomes inconsistent. However, the step (4) was invoked in one of the experiments, since the theorem prover found an error in the composition table for the RCC-calculus showed in [8].

6 Experiments

It has been used a computer with two Pentium III (800 Mhz) processors and 256 Mb RAM. The machine runs with Red Hat Linux operating system 7.0. The processed database has 40242 clauses, and it is processed in 6.5 seconds).

It is not our aim to use the theorem prover as a database programming language. The idea is to ask to the system complex questions which are unsolvable by constraint satisfaction algorithms or simple SQL commands. The questions are driven to obtain knowledge on spatial relationships not explicit in the database (as `Proper-part` or boolean combination of complex spatial relations). Some of the questions require an excessive CPU time. Surprisingly, the time cost is justified: the theorem prover *thought* all the time on the database and it found many errors of the type (A1), errors which are not acceptable. The number of useless results of type (A2) obtained can be significatively reduced by a spatial interpretation of some skolem functions from the clausal form of domain knowledge.

We select the predicates `Part-of`, `Proper-part`, `Externally-connect` as targets of the study, in each step 3. Several results are in the fig. 7. (R1) shows the first correct answer to the question, (R2) shows the results 5 seconds later, (R3) shows the first useless result and (R4) shows the first error found (of the database or the Knowledge domain) if one has been found.

Other interesting consequence of the experiment is that the search of non-acceptable inconsistencies (A1) produces, in some cases, *inconsistent argument schemes*: schemes that, by binding of the variables, produces many inconsistencies from the first one founded. This behaviour can be interpreted as a learning process because the system generates a general inconsistent argument to locate lacks or concrete errors in the database. This phenomenon leads us to a fact experimentally checked: when the first one is found, the next ones are quickly found.

It is also interesting to remark that among the features of the system, we can use an option which allows us to find many proofs of the same answer (that is, many arguments). Thus, we can locate several anomalies in the database only with an incorrect answer.

```
Length of proof is 4.  Level of proof is 4.

---------------- PROOF ----------------

5 [] -P(x,y)|P(y,x)|PP(x,y).
8 [] -P(x,y)| -P(y,x)|EQ(x,y).
446 [] P(ANDEVALO_ORIENTAL,x)|x!=HUELVA.
8946 [] HUELVA!=ANDEVALO_ORIENTAL.
63366 [] -PP(x,HUELVA)|$Ans(x).
63371 [] x=x.
69229 [hyper,63371,446] P(ANDEVALO_ORIENTAL,HUELVA).
77764 [hyper,69229,5,unit_del,63366] P(HUELVA,ANDEVALO_ORIENTAL)
      |$Ans(ANDEVALO_ORIENTAL).
83179 [hyper,77764,8,69229,flip.2] $Ans(ANDEVALO_ORIENTAL)
      |EQ(HUELVA,ANDEVALO_ORIENTAL).
89721 [para_from,83179.1.1,8946.1.1] ANDEVALO_ORIENTAL!=ANDEVALO_ORIENTAL
      |$Ans(ANDEVALO_ORIENTAL).
89722 [binary,89721.1,63371.1] $Ans(ANDEVALO_ORIENTAL).

------------ end of proof -------------
```

Figure 6: An example of result for `Proper-part(x,HUELVA) -> $Ans(x)`. The first six clauses and `$Ans(ANDEVALO_ORIENTAL)` form the O-argument.

`Part-of(x,Jaen) -> $Ans(x)` [Step (1)]

Exp.	CPU time (sec.)	generated clauses	results	(A1)	(A2)	(A3)	(A4)
(R1)	53.79	175	1	0	0	0	0
(R2)	59	6,661	25	4	102	0	0
(R3)	58.4	1,098	2	2	1	0	0

`Overlaps(SIERRA_NORTE,x)` $\wedge$ `Overlaps(Cordoba,x) -> $Ans(x)` [Step (2)]

Exp.	CPU time (sec.)	generated clauses	results	(A1)	(A2)	(A3)	(A4)
(R1)	592.55	32,473	1	22	0	0	0
(R2)	597	32,517	5	22	0	0	0
(R3)	54.19	182	0	0	1	0	0

`Proper-part (x, Huelva) -> $Ans(x)` [Step (2)]

Exp.	CPU time (sec.)	generated clauses	results	(A1)	(A2)	(A3)	(A4)
(R1)	2395.31	195,222	1	113	0	0	0
(R2)	2400	201,797	8	113	0	0	0
(R3)	2514.46	287,088	14	117	0	1	0
(R4)	54.15	286	0	1	0	0	0

`Externally-connect (x, Sevilla) -> $Ans(x)` [Step (3)]

Exp.	CPU time (sec.)	generated clauses	results	(A1)	(A2)	(A3)	(A4)
(R1)	2360	188.169	1	113	0	0	0
(R2)	2366	196,575	12	113	0	0	0
(R3)	53,33	184	0	1	0	1	0
(R4)	3845	11,673,078	25	113	0	6	72

Figure 7: Some results of the cleaning cycle

7 Final remarks

A methodology for the cleaning of complex knowledge databases, aided by an automated theorem prover, is given. The cleaning cycle allows to detect anomalies of the data with the knowledge domain, as well as in the self domain. The next challenge is to develop an agent based in this cycle. Previously we must select strict flags to handle big sets of useless data. We believe that the cleaning agent can be a promising application of the field of automated deduction to make consistent the data in environments such as the Semantic Web.

References

[1] J. Broekstra, M. Klein, S. Decker, D. Fensel and J. Horrocks, "Adding formal semantics to the Web: building on the top of RDF Schema," in *Proceedings of the Workshop Semantic Web: Models, Architectures and Management, Lisbon, September 21, 2000*
`http://www.ics.forth.gr/proj/isst/SemWeb/program.html`

[2] G. Weiss, *Multiagent Systems. A Modern Approach to Distributed Artificial Intelligence*, MIT Press, Cambridge: 1999.

[3] M. Bozzano, G. Delzanno, M. Martelli, V. Mascardi, F. Zini, "Logic programming & Multi-agent systems: a synergic combination for applications and semantics," in K. Apt, V. Marek, M. Truszczynski and D. S. Warren (eds.), *The Logic Programming Paradigm - A 25-Year Perspective*, Springer Verlag, Berlin: 1999, pp. 5-32 .

[4] H. van Dyke Parunak, "Industrial and Practical Applications of DAI," in [2], pp. 377-421.

[5] I. Dahn, J. Denzinger, "Cooperating Theorem Provers," in W. Bibel and P.H. Schmitt (eds.) *Automated Deduction. A basis for applications. Vol. II: Systems and Implementation Techniques*, Kluwer Academic Publishers, Dordrecht: 1998, pp. 383-416.

[6] O. Lassila, "Web Metadata: A Matter of Semantics," *IEEE Internet Computing*, vol. 2, no. 4, 1998, pp 30-37.

[7] R. Fikes, D. L. McGuinness, and R. Waldinger, "A First-Order Logic Semantics for Semantic Web Markup Languages", Technical Report n. KSL-02-01. Knowledge Systems Laboratory, Stanford University (2002).
`http://www-ksl.stanford.edu/KSL_Abstracts/KSL-02-01.html`

[8] A. G. Cohn, B. Bennett, J. M. Gooday and N. M. Gotts, "Representing and Reasoning with Qualitative Spatial Relations about Regions," In O. Stock (ed.) *Temporal and spatial reasoning*, Kluwer Academic Publishers, Dordrecht: 1997, pp. 97-134.

[9] J. Renz, B. Nebel, "On the Complexity of Qualitative Spatial Reasoning: A Maximal Tractable Fragment of the Region Connection Calculus," *Artificial Intelligence*, vol. 108, no. 1-2, 1999, pp. 69-123.

[10] B. Bennett, "The application of Qualitative Spatial Reasoning to GIS" In R.J. Abrahart (ed.) *Proceedings of the First International Conference on GeoComputation*, 44-47, Leeds, (1996).

[11] W. McCune, *OTTER's user manual*, Argonne National Laboratory (1994).
`http://www-unix.mcs.anl.gov/AR/otter/`

[12] M. Elvang-Goransson and A. Hunter, "Argumentative logics: Reasoning from classically inconsistent information," *Data and Knowledge Engineering*, vol. 16, no. 2, 1995, pp. 125-145.