Reconciling Distributed Information Sources*

José Luis Ambite and Craig A. Knoblock
Information Sciences Institute and Department of Computer Science
University of Southern California
4676 Admiralty Way
Marina del Rey, CA 90292
{jambitem, knoblock}@isi.edu

Abstract

In a network of information agents, the problem of how these agents keep accurate models of each other becomes critical. Due to the dynamic nature of information and the autonomy of the agents, the models that an agent has of its information sources may not reflect their actual contents. In this paper, we propose an approach to automatically reconcile agent models. First, we show how an agent can revise its models to account for the disparities with its information sources. Second, we show how to learn concise descriptions of the new classes of information that arise. Third, we show how the refined models improve both the accuracy of the knowledge of an agent and the efficiency of its query processing. A prototype for model reconciliation has been implemented using a SIMS mediator that accesses relational databases.

Introduction

With the current explosion of data, the problem of how to combine distributed, heterogeneous information sources becomes more and more critical. Agents that integrate the information resources available for particular domains provide a way of structuring this vast information space. These *information agents* provide expertise on a specific topic by drawing on their own knowledge and the information obtained from other agents. The SIMS architecture (Arens *et al.* 1993, Knoblock *et al.* 1994) implements such an agent.

Previous work has assumed that the contents of the information sources were well known, fairly static and with all semantic disparities resolved at design time. However, in a network of *autonomous* agents these assumptions will not hold in general. First, the designer of the models might not have had a complete understanding of the semantics of the information provided by each agent. Second, even if at integration time the models were accurate, the autonomy of the agents will cause some concepts to drift from their original meaning. The dynamic nature of the information force us to provide mechanisms to detect inconsistency and/or incompleteness in the agent's knowledge. Third, the analysis described below may feed additional information to a fully automatic integration system.

In this paper, we propose an approach to automatically reconcile agent models, by analyzing the evolving contents of the information sources and modifying the models accordingly. A prototype for model reconciliation has been implemented using a SIMS mediator that accesses relational databases. The paper is organized as follows. First, we describe the knowledge of an agent which is represented as a set of interrelated models. Second, we show how to analyze these models to detect disparities and how the models are refined to resolve the disparities. Third, we describe a learning approach to generate declarative descriptions of the new concepts found in the previous analysis. Fourth, we show how this work improves not only the accuracy of the models, but also the efficiency of the query processing. Fifth, we describe related work. Finally, we discuss the contributions and future directions.

Agent Models

Each agent contains a model of its domain of expertise and models of the other agents that can provide relevant information. We will refer to these two types of models as the domain model and information source models. These models constitute the knowledge of an agent and are used to determine how to process an information request. The domain model is an *ontology* that represents the domain of interest of the agent. It gives the terminology for interacting with the agent. The information source models describe both the contents of the sources and their relationship to the domain model. They do not need to contain a complete

*This research was supported in part by Rome Laboratory of the Air Force Systems Command and the Advanced Research Projects Agency under contract F30602-91-C-0081, and in part by the National Science Foundation under grant IRI-9313993. The first author is supported by a Fulbright/Spanish Ministry of Education and Science grant. The views and conclusions contained in this report are those of the authors and should not be interpreted as representing the official opinion or policy of RL, ARPA, NSF, the U.S. Government, MEC, the Fulbright program, or any person or agency connected with them.

description of the other agents, but rather only those portions that are directly relevant. All these models are expressed in the Loom description logic (MacGregor 1990). Loom provides a language for representing hierarchies of classes and relations, as well as efficient mechanisms for classifying instances and reasoning about class descriptions.

Figure 1 shows a fragment of the knowledge of an agent. Its domain of expertise is organization modeling. It integrates several databases of a company into a common enterprise model. The nodes represent concepts (classes). Those unshaded belong to the agent's domain model and those shaded to the agent's information source models. The thick arrows represent subsumption, the thin ones, roles (binary relationships). All of the concepts and roles in the information source models are mapped the domain model. A mapping link (dashed lines) between two concepts or roles indicates that they represent the same class of information. Two source concepts mapping into the same domain concept are interpreted as both denoting the same set of individuals. Some of the properties of the individuals might be equivalently obtained from either source, but other properties might only be described in one of them. We assume that individuals are identified by key attributes (underlined). For example, `personnel.emps` and `accounting.staff` can be considered *equivalent* sources of `employee` information (in their common roles). Their instances are identified through the key role(s) `ss#`.

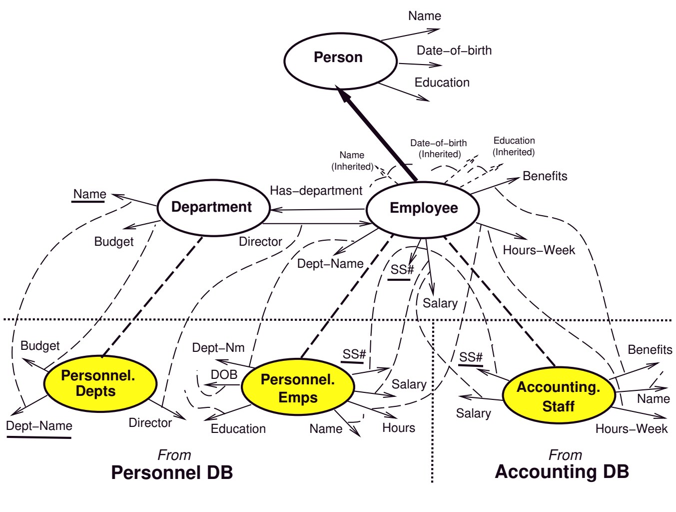

Figure 1: Fragment of the Knowledge of an Agent

Refining the Models

This section describes how the semantic disparities among concepts belonging to different agents (thus possibly different ontologies) are recognized and how the models are modified to resolve such disparities. The main tool will be to analyze the extensions of inter-related concepts looking for possible mismatches between the definitions stated in the models and the actual instances. In the first case, only one source provides some class of information. In the second case, two sources provide instances for the same domain concept. We will illustrate these ideas with examples from the domain of Figure 1 by assuming different extensions of the information sources, which will represent different underlying semantics.

Verifying a Single Information Source

The first verification phase consists in checking that the extension of an information source concept actually satisfies the definition of the corresponding domain model concept. Consider the domain model concept `employee` (`DMC`) and its corresponding information source concept `personnel.emps` (`ISC`). Assume that there is a constraint in the definition of `employee` that states that `salary` has to be greater than the legal minimum wage. However, after analyzing the actual instances of `emps` present in the `personnel` database, a group of instances identified by null `salary`, violates this constraint. This new class of instances might correspond to visitors, which have external funding. The model will be changed as shown in Fig 2. A new concept `C1` (`personnel`) is automatically created in the domain model to represent the actual contents of the information source. This concept will have as subclasses both the original `DMC` (`employee`) and another newly created concept `C2` (`visitor`) that represents the information discovered. The definition of `personnel` is a generalization of `employee` by removing the constraints in conflict. The definition of `visitor` inherits the properties of `personnel` plus the additional constraints that characterize its instances. In the next section we will discuss how these new descriptions are learned. For now, their names will hint at the actual operational explanation.

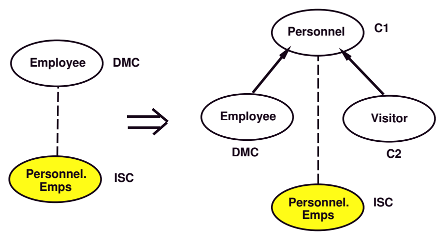

Figure 2: Model Refinement when Data is Inconsistent with Initial Definition

Reconciling Two Related Sources

Consider two information source concepts (`ISC1`, `ISC2`) mapped to the same domain model concept (`DMC`), with extensions that have evolved so that the initial integration is no longer accurate. In order to detect and resolve the disparities among the models, we will gather the actual instances of the source concepts and compare them using their key roles. There are four cases:

Contained. Imagine that the `accounting` department keeps track of part-time employees in its database. However, the `personnel` department does not consider them as part of the organization. In such case, the set of keys obtained from `personnel.emps.ss#` will be contained in the set `accounting.staff.ss#`. To solve this mismatch, the domain model will be changed as shown in Fig 3. A new concept `C1` (`full-time employee`) is created in the domain model representing the actual contents of `ISC1` (`personnel.emps`) including in its definition a description that explains why it is a specialization of `DMC` (`employee`). The mappings between `ISC1` and `DMC` are retracted and restated between `ISC1` and `C1`. `ISC2` (`accounting.staff`) mappings remain unaltered. Another new subconcept `C2` (`part-time employee`) is created to represent those instances present in `ISC2` but not in `ISC1`.

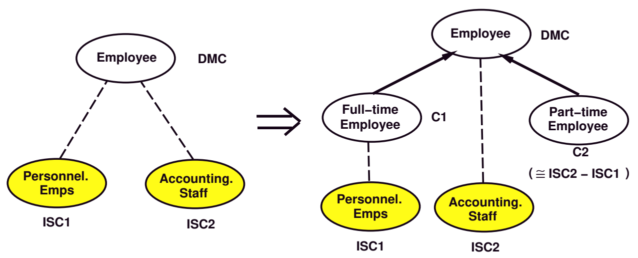

Figure 3: Model Refinement when One Source Is Contained in Another

Equal. If the extension of both key sets is identical the initial integration remains valid.

Overlap. Consider a different situation in which the `personnel` database keeps track of visiting researchers, but does not consider part-time employees. Conversely, the `accounting` department considers part-time employees but not visitors, which are externally funded. By analyzing the extensions of `personnel.emps` (`ISC1`) and `accounting.staff` (`ISC2`), the system notices that their individuals overlap. The domain model is modified as shown in Fig 4. The following new concepts are formed: `C1` (`full-time personnel`) and `C2` (`salaried personnel`) that represent the *actual* contents of `ISC1` and `ISC2`, respectively; `C1-C2` (`visitor`) and `C2-C1` (`part-time employee`), that represent individuals identified by keys in the set differences; and `C1∩C2` (`regular employee`), for the intersection.

Disjoint. Finally, we will present a more complex situation in order to illustrate the case of disjoint data. Imagine that two other departments of the organization, computer support and purchasing, decide to keep employee information for their own purposes and copy the data from the personnel department database (creating `cs.emps` and `pu.emps` respectively). The inte-

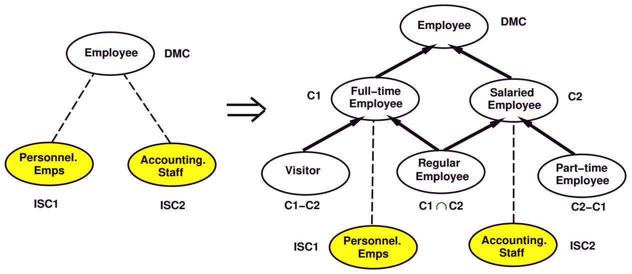

Figure 4: Model Refinement when Information Overlaps

gration is performed during this state of affairs, and the three databases are considered equal sources of `employee` information. However, with time, the computer support department decides that only the personnel of the engineering department is of interest to their applications eliminating the other employees from the database. Similarly the purchasing department restricts its attention to the manufacturing employees. Moreover, they delete the attribute `dept-nm` from their schemas, which is now useless to them. Due to this change in the semantics of the underlying data, the concepts `cs.emps` and `pu.emps` are now disjoint (and both contained in `personnel.emps`). The domain model is changed as shown in Figure 5.

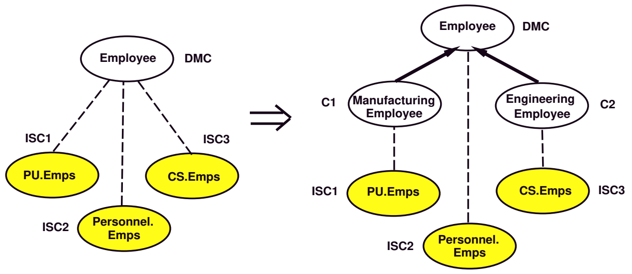

Figure 5: Model Refinement when Information Is Disjoint

Learning New Descriptions

In the previous section, we proposed a way of generating novel concepts that represented misalignments among the models. However, in order to resolve the disparities, not merely acknowledge them, we need to understand why these sets of conflicting instances have appeared in the first place. We need to provide declarative and concise descriptions of these new classes. Machine learning algorithms can be used to discover regularities within these sets of instances. Our system currently uses ID3 (Quinlan 1986). These type of explanations may not be as hard to find in practice as it may seem. Significant disparities will not be mere errors, but follow some rationale. The semantics of some concepts in the information sources might have drifted, but not enough to impose a schema change, as, for example, when data that naturally groups into new classes is added to pre-existing categories. The description of the new classes will combine the original domain concept description and the constraints induced from the actual instances. These new descriptions will be classified appropriately in the domain hierarchy, justifying the subsumption relationships stated in Figures 2 to 5.

Consider the case in Figure 3 where one information source (`accounting`) contains all the individuals of interest, while other (`personnel`) only stores a subset of them. The system proceeds as follows. First, the system gathers all the data from `personnel.emps` and `accounting.staff`, comparing them using the key role `ss#`. This results in two groups of individuals, one identified by the values of `personnel.emps.ss#`, and the other identified by the values of the set difference between `accounting.staff.ss#` and `personnel.emps.ss#`. Second, these sets are prepared for a two category learning problem. We choose the common attributes (`name`, `ss#`, `salary`, and `hours-week`) as the features over which to learn. Note also that these attributes must be expressed in the terminology of the domain model, where we need to learn the descriptions, not in that of the information sources. Third, we pass this data to ID3. Finally, the resulting decision tree is analyzed to extract concise formulas, expressed in the concept definition language of Loom, explaining the two learned categories. In this example, the constraint that characterizes the set difference is `(<= hours-week 20)`. The complete concept description[1] is:

```
(defconcept part-time-employee :is
            (:and Employee (<= hours-week 20)))
```

The descriptions in the example of overlapping information in Figure 4, which involve constraints with the attributes `salary` and `hours-week`, are obtained analogously.

Consider now the example of disjoint information shown in Figure 5. In this case, the concepts `pu.emps` and `cs.emps` do not have any instances in common because they refer to individuals in different departments of the organization. Moreover, recall that the department to which an employee belongs, which would allow to differentiate the individuals, is not stored in the databases corresponding to these concepts. However, such information can be obtained from `personnel` database. Now, the system builds the three category learning problem involving the mutually disjoint sets `pu.emps`, `cs.emps` and the remainder of `personnel.emps`. The first learned description (corresponding to `pu.emps`), is:

```
(defconcept manufacturing-employee :is
  (:and employee
        (:filled-by dept-name 'manufacturing)))
```

[1]Note that the name of the concepts, such as `part-time employee`, has been assigned manually. The system only generates an unused token, such as `C2` or employee-254. However, names that spell out the learned constraints, such as `employee-with-hours-week-less-than-20`, are feasible, although not necessarily desirable.

The description of the second set, `engineering employee` (corresponding to `cs.emps`) is analogous. Finally, the description learned for the third set is a disjunction of all the departments names except manufacturing and engineering. This kind of constraint does not improve the expressiveness of the domain model. Therefore, no new concept is added. Not all the possible explanations generated by the learning algorithm will be equally useful. In some cases, the constraints generated might be too cumbersome, such as long disjunctions of values, or long conjunctions of constraints that result in too general or too specific concepts. Also, we need to take into account the cardinality of the new classes. If the proposed new class has a very small number of instances, it might signal errors in the database. Our algorithm only creates a new concept if a concise description with sufficient support can be learned.

Using the Reconciled Model

In this section, we will show how the previous analysis not only provides a more accurate description of the information available to an agent, it also improves the efficiency of its query processing. First, we briefly describe how a SIMS agent satisfies an information request. Second, we will illustrate how this processing is improved thanks to the model reconciliation.

Query processing requires developing an ordered set of operations for obtaining a requested set of data. This query access plan includes selecting the information sources for the data, the operations for processing the data, the sites where the operations will be performed and the order in which to perform them. A critical step in this process is query reformulation, which transforms a query expressed in the domain ontology into terms of the appropriate information sources. For example, a domain level query including the concept `employee` and the role `dept-name` would be translated into an semantically equivalent one using the information source terms `personnel.emps` and `dept-nm`, respectively. When there is no source concept directly linked to the domain concept of interest, the SIMS reformulation mechanism will look for one its superconcepts or subconcepts with a direct mapping to some available information source. For example, suppose we ask a query requesting the `data-of-birth` of some `person`, but there is no source that can provide `person` information. First, the query will be reformulated using the subconcept `employee` which does have source links. Then, it will be rewritten again using `personnel.emps`, which is selected because it can provide the required `DOB` data.

The advantages of using the refined models are twofold. Firstly, the available information is represented with greater accuracy. Moreover, this mechanism adapts automatically to the evolution of the information sources, whose contents may semantically drift from the original domain model mappings. Also, human designers may revise these concepts to both refine the domain model and correct errors in the databases. Secondly, the efficiency of query processing is increased. During query reformulation, the new concepts provide better options for building the query plan that retrieves the desired data.

Consider a query that requests all employees that work less than 15 hours a week, posed against the models in Figure 3. In the original model, the system does not have any information to prefer either of the two information sources. However, recall that only the extension of `accounting.staff` included part-time employees (characterized by working less than 20 hours a week). If the system selects `personnel.emps`, the query access plan will be *incorrect*. That information source contains full-time employees exclusively, and will wrongly return a null answer. In contrast, with the refined models the query is reformulated into `(:and part-time-employee (< hours-week 15))`, which is correctly posed against the `accounting` database.

Consider another query that requests all employees in the manufacturing department, posed against the models in Figure 5. In the initial model any of the three information sources could be chosen. If the planner selects `cs.emps` the query would be incorrect as in the previous case. If it selects `personnel.emps`, the query access plan will be *inefficient*. That information source contains information about all the employees in the company, and many more instances will need to be considered. With the refined model, better cost estimates are available. The query will be reformulated using the subconcept `manufacturing-employee` which is linked to the source concept that provides the desired data (`pu.emps`). Finally, note that in complex queries all these savings are cumulative.

Related Work

In a broad sense, the research in knowledge discovery in databases (Piatetsky-Shapiro 1991) is related to the present work. However, our system focuses in learning concepts that explain the disparities in the models, as opposed to arbitrary regularities in the data. Due to this specificity, it can operate without human intervention.

There are several projects that use descriptions logics to access, structure, and discover information from databases. Beck et al. (Beck *et al.* 1994) use conceptual clustering techniques for schema design. Instances that are asserted to belong to a class, but do not satisfy its definition completely, prompt schema changes. In our system, we assume that the domain model reflects an initial integration. However the autonomy of the information sources results in semantic drift and disparities. Our model refinement is geared to keep track of the evolving information sources and only triggered when disparities are detected. Blanco et al. (Blanco *et al.* 1995) build an integrated schema considering equivalent, contained, overlapping and disjoint relations among classes, but these are asserted by a human as time-invariant properties of the databases. No attempt is made to discover an explanation for those relationships.

Finally, (Perkowitz and Etzioni 1994) present an approach to automatically discover correspondences between models by comparing database contents with factual knowledge that an agent already has. In this way, they try to assign a meaning to the database tokens. Our work could provide additional support for the proposed mappings and suggest new refinements to the models.

Discussion

Our goal is to build intelligent information gathering agents that are resilient to the changes that dynamic environments composed of autonomous information sources will impose. Toward this goal we have implemented a model reconciliation system in our agents that allows them to *adapt* their models to better keep track of the contents of their information sources. Moreover, this increases their performance by providing them with better options during query processing.

Future work will include exploring which learning/knowledge discovery algorithms provide more useful concept descriptions (for example, results from (Cohen and Hirsh 1994) may be promising), how to perform the extensional analysis more efficiently (for example, incrementally), and handling more types of inconsistencies.

References

Yigal Arens, Chin Y. Chee, Chun-Nan Hsu, and Craig A. Knoblock. Retrieving and integrating data from multiple information sources. *International Journal on Intelligent and Cooperative Information Systems*, 2(2):127–158, 1993.

Howard W. Beck, Tarek Anwar, and Shamkant Navathe. A conceptual clustering algorithm for database schema design. *IEEE Trans. Knowledge and Data Engineering*, 6(3):396–411, 1994.

J.M. Blanco, A. Illarramendi, and A. Goñi. Building a federated relational database system: An approach using a knowledge based system. *To appear in International Journal of Intelligent and Cooperative Information Systems*, 1995.

William W. Cohen and Haym Hirsh. Learning the classic description logic: Theoretical and experimental results. In Erik Sandewall Jon Doyle and Pietro Torasso, editors, *Priciples of Knowledege Representation And Reasoning. Proceedings of the Fourth International Conference*, pages 121–133, Bonn, Germany, 1994.

Craig Knoblock, Yigal Arens, and Chun-Nan Hsu. Cooperating agents for information retrieval. In *Proceedings of the Second International Conference on Cooperative Information Systems*, Toronto, Canada, 1994.

Robert MacGregor. The evolving technology of classification-based knowledge representation systems. In John Sowa, editor, *Principles of Semantic Networks: Explorations in the Representation of Knowledge*. Morgan Kaufmann, 1990.

Mike Perkowitz and Oren Etzioni. Database learning for software agents. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, Seattle, WA, 1994.

G. Piatetsky-Shapiro. *Knowledge Discovery in Databases*. MIT Press, Cambridge, MA, 1991.

J. Ross Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, 1986.