Combining Inductive and Deductive Inference in Knowledge Management Tasks *

Alvaro A. A. Fernandes
Department of Computer Science, University of Manchester
Oxford Road, Manchester M13 9PL, United Kingdom
email: a.fernandes@cs.man.ac.uk

Abstract

This paper indicates how different logic programming technologies can underpin an architecture for distributed knowledge management in which higher throughput in information supply is achieved by a (semi-)automated solution to the more challenging problem of knowledge creation. The paper first proposes working definitions of the notions of data, knowledge *and* information *in purely logical terms, and then shows how existing technologies can be combined into an inference engine, referred to as a* knowledge, information and data engine *(KIDE), integrating inductive and deductive capabilities. The paper then briefly introduces the notion of* virtual organizations *and uses the set-up stage of virtual organizations to exemplify the value-adding potential of KIDEs in knowledge management contexts.*

1. Introduction

The current pace of change in technological, organizational and economic contexts seems to project into the future with little likelihood of slowing down. This creates significant challenges for the management of knowledge, information and data.[1] One implication is that the shelf life of knowledge and data decreases under pressure from constantly changing information needs. The life cycle of KIDM tasks must, therefore, be as short as possible if value is to be adequately added to data, thereby satisfying the information needs of organizations and individuals. The question then arises as to the conditions under which KIDM is feasible, even from a purely cost/benefit point of view, without some degree of automation.

A solution would be particularly relevant to market situations in which the ability to exploit opportunities requiring extremely short KIDM life cycles is a vital competitiveness issue. One such situation is characterized when market opportunities emerge that might recommend the setting up of a virtual organization (VO) [1, 5, 20] or an e-business.

An aspect of the broader KIDM problem that has not been addressed in a well-founded and systematic manner is that of knowledge creation. This is somewhat surprising insofar as the foundations with which to attempt an expansion of data integration goals to encompass knowledge creation seem to be in place in a context that particularly suits the database view on KIDM via the deductive database paradigm [6, 7, 15]. Such foundations lie in research into inductive logic programming (ILP) [12, 16, 17]. ILP applies classical machine learning techniques [11] to the induction of logic programs [2, 13]. Inducing, from examples, a logic program that characterizes a concept or a function is a form of knowledge creation. Given that many database concepts can be formulated as logic programs (e.g., database states, views, integrity constrains, implicit data, schema definitions) [6, 7, 15], there has been interest in using ILP techniques to (semi-) automate design tasks such as the formulation of views, integrity constraints, and others [3, 4], using stored data as, and perhaps additionally complementing it with, examples.

This paper indicates how different logic programming technologies can underpin an architecture for KIDM in which the problem of knowledge creation is addressed in a well-founded and systematic manner. The paper does not report on completed research, but rather outlines a research programme whose long-term goal is a more principled, well-founded generic set of solutions to the problems arising from the need to ensure that KID are served to applications in the quantity and quality needed. The paper is, therefore, best understood as a position paper in which, as a first step, an architecture for KIDM is proposed whose main novelty lies in the idea of a *KID engine* (KIDE) integrating induction and deduction.

---

* (An extended version [8] is available.)

[1] The acronym 'KID' is used throughout to refer, collectively, to these three distinct notions. Section 2 describes in which sense they are, in this paper, to be distinguished. 'KIDM' is used as a shorthand for 'knowledge, information and data management'.

The remainder of the paper is structured as follows. Section 2 proposes working definitions of the notions of data, knowledge and information in logical terms. Section 3 suggests that existing technologies emerging from research into the logic programming paradigm for computing and artificial intelligence can be combined in an inference engine, referred to as a KIDE, integrating inductive and deductive capabilities. Section 4 uses an example to argue that KIDEs add value to the outcome of KIDM process. Section 5 briefly introduces the notion of VOs and uses the set-up stage of VOs to argue that the value-adding potential of KIDEs in KIDM contexts can make a crucial difference regarding whether a VO is set up or not. Section 6 draws a few conclusions about the approach contributed in the paper.

2. KID: A Logic-Based View

A *data* item is the representation of one particular fact. For example, that a person has certain competences, knows certain programming languages, has studied in certain universities, etc. Here, data items are assumed to be expressed as a tuple of values (e.g., `employee(fiona, databases, oql, manchester)`). A tuple of values records something about the state of the modelled domain in the wake of an event occurrence (e.g., that Fiona, a Manchester graduate, was employed as a database professional for her expertise in OQL), although event types and event occurrences are most often not modelled as such (except, e.g., in some classes of temporal databases and other more specialized systems than those that lie in the scope of this paper). Note that the string `employee(fiona, databases, oql, manchester)` is, in the context of this paper, a well-formed formula (wff) in a formal system with well-defined proof and model theories. The formal system to be had in mind in this case is Datalog [6, 7, 15] (or any of its well-defined extensions) and the remainder of this document assumes basic familiarity with Datalog and its notational conventions.

Data is purely extensional. By *extensional* is meant that the representation (in this context, the asserted ground sentence expressed as a Datalog fact) does not denote more than one fact, and hence, e.g., it says something about one event occurrence only. To elicit all the information of interest about a set of event occurrences, in the case where each is represented by a data tuple, all that needs to be done is to enumerate the wffs that denote those event occurrences. Of course, assuming enumeration, one can also filter, combine, aggregate, etc.

A *knowledge* item is the representation of many particular facts, in the sense that from a single knowledge item many particular facts can be made explicit by the application of inference rules. For instance (and carrying on from the previous example), from the knowledge that whoever knows OQL also knows SQL, one may infer the fact `employee(fiona, databases, sql, manchester)` *and* similar facts for every other employee of whom it has been explicitly asserted that they know OQL. Knowledge items, like data items, are also expressed as wffs in a system of inference (e.g., a Datalog rule such as `employee(P, C, sql , S)` $\leftarrow$ `employee(P, C, oql , S)`)).

Knowledge is primarily intensional. By *intensional* is meant that the representation (in this context, the asserted closed universally quantified sentence expressed as a Datalog rule) denotes more than one fact, and hence, e.g., it says something about many event occurrences. In this case, to elicit all the information of interest a simple enumeration of wffs does not suffice. Information is implicit in such wffs that still needs to be elicited. One needs to apply to such wffs an inference procedure to compute its logical completion, which, by definition, contains all the facts inferable from those wffs. In the Datalog context, this completion is a finite set of data representations (i.e., grounded sentences), that could, ultimately, be enumerated.

An *information* item is, again, a wff. More specifically, it is either a data or a knowledge item derived from current data and knowledge stocks by an inference procedure. Given a set $K \cup D$ of knowledge and data items, the amount of available information is that superset thereof characterizable from $K \cup D$ by application of an inference procedure (in the broad sense, i.e., including classical query evaluation. enumeration, filtering, combination, aggregation, etc.). Information is, therefore, being thought of as knowledge and data that can be derived from knowledge and data by inferential processes. For example, the answers to a request such as 'enumerate all represented facts of a specified kind' constitute information in the form of data. For example, given the request `?- employee(E, _, oql, _)`, a Datalog evaluation engine would enumerate the names of employees who know OQL, among which is `fiona`. If the rule expressing the relationship between OQL and SQL suggested above were available, then given the request `?- employee(E, _, sql, _)`, a Datalog evaluation engine would also provide the information that `fiona` knows SQL even though it was only ever explicitly asserted that `fiona` knows OQL. In the broad context of this discussion, the following two procedures for computing information from data and knowledge representations are usually considered. In the absence of knowledge representations, one can use traditional query answering algorithms (as embodied in mainstream database products). In the presence of knowledge representations, one can use Datalog evaluation algorithms (as embodied in deductive database engines).

A logic-based approach to formalizing the relationship between knowledge, information and data can, therefore, be characterized as follows. Data, knowledge and information are uniformly represented as wffs in a system of inference.

Information is obtained by deductive processes, i.e.: $\mathit{deduction} : (\mathit{data} \cup \mathit{knowledge}) \rightarrow \mathit{information}$. In traditional databases, $\mathit{knowledge} = \emptyset$ and $\mathit{deduction} := QP$, where QP denotes classical query processing. In deductive databases, $\mathit{knowledge} = SCH \cup ID \cup IC$ and $\mathit{deduction} := DE$, where SCH, ID and IC are sets of Datalog rules, denoting, resp., schema definitions, implicit data and integrity constraints; and DE denotes a deductive database engine (i.e., a Datalog evaluator [6, 7, 15]). As already noted, many instantiations of DE would serve the purposes of the proposed research (see, e.g., [14] for a recent unified treatment).

Knowledge, on the other hand, is obtained by inductive processes, i.e: $\mathit{induction} : (\mathit{data} \cup \mathit{knowledge}) \rightarrow \mathit{knowledge}$. If we let $\mathit{induction} := ILPE$, where $ILPE$ denotes any one of many ILP engines proposed in the literature [12, 16, 17], then we can let *data* and *knowledge* have the same syntax, semantics and metalogic as assumed by DE above. This means that there is a uniform, well-founded approach to creating knowledge (via induction) and deploying and disseminating it (via deduction), insofar as acquired knowledge is concomitantly available to generate information via deductive inference. This observation suggests it would be useful to build inference engines combining deduction and induction.

3. An Inference Engine for KIDM

The notion of a KID engine embodies an attempt to integrate deductive and inductive inference using logic programming techniques. Interactions with, and within, KIDEs are depicted in Figure 1, with a round-cornered box representing a process, arrows representing flows, an open-sided rectangle representing a store and a square representing an external entity. Arrow labels are explained below.

User-provided observations[2] are stored as data and used deductively, as facts, and inductively, as evidence (i.e., examples for the inductive learning task performed by the inductive engine). User-provided knowledge is stored and used deductively, as application specific axioms (e.g., schema definitions, implicit data and integrity constraints), and inductively, as generic, background knowledge for the inductive learning task. The user also specifies the inductive task (e.g., the syntactic form of the knowledge to be induced, bounds on complexity parameters, etc.). The output of inductive process are hypotheses which incorporate whatever evidence and background knowledge was said to be relevant. When hypotheses are assimilated they are accepted as knowledge created by the inductive engines. Finally, in this context, information is served in response to queries. If both induction and deduction operate over representations whose syntax, semantics and metalogic are the same, then the way is open for an architecture that integrates knowledge and data to increase the variety and quantity of information in a semi-automated manner that is well-founded in logic. In logical terms, the formal relations in Figure 2 characterize the parameters and the nature of the inferences depicted in Figure 1. Note the use of '$\vdash$' to denote deductive inference and '$\sim\!\!|$' to denote inductive inference. Note that induction is being viewed here as an inversion of deductive processes [16, 19] but the notion of a KIDE does not strictly require this approach to induction, other approaches (see [10, 17] for a comprehensive survey) might also be effective. Note also that, in Figure 1, the arrows labelled check and outcome denote the dependency of the inductive engine on the deductive engine to check that hypotheses under consideration imply the positive examples given and do not imply negative ones. This reflects the fact that many ILP approaches rely on testing whether conjectures imply or not certain pieces of evidence provided for the inductive task at hand.

[2]The word 'observation' is used here for its generality. It should be thought of as the representation of events of interest.

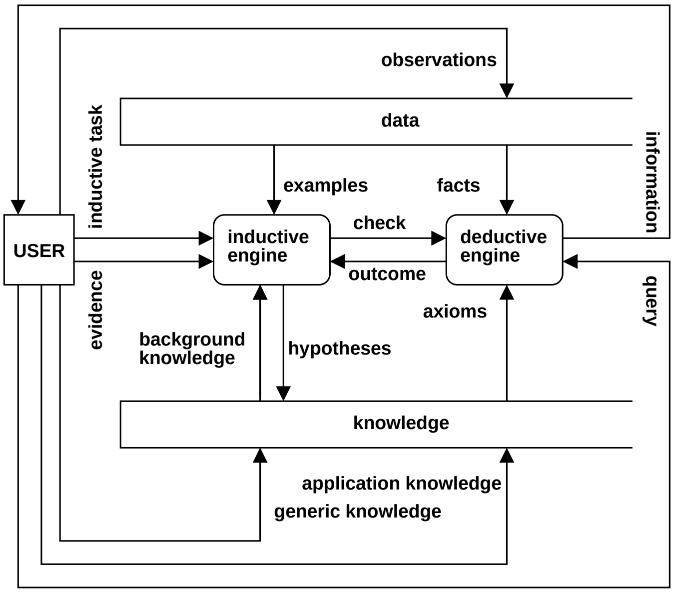

Figure 1. Interactions in KIDEs

4. KIDEs Add Value to KIDM

This section revisits in more detail the examples used in Section 2 and indicates how the KIDEs introduced in Section 3 and depicted in Figure 1 can supply valuable information that would be much harder to obtain without them. An example can show how the addition of inductive capabilities to deductive ones helps in keeping pace with changes in underlying KID stocks and in speeding up, in a cost-effective

$$
\begin{array}{rcl}
hypothesis & \mathrel{\sim\!\!\dashv} & background \cup examples \\
background \cup hypothesis & \vdash & examples \\
hypothesis & \subseteq & \textsf{KNOWLEDGE} \\
background & \subseteq & \textsf{KNOWLEDGE} \\
examples & \subseteq & \textsf{DATA} \\
facts & \subseteq & \textsf{DATA} \\
axioms & \subseteq & \textsf{KNOWLEDGE} \\
facts \cup axioms & \vdash & \textsf{INFORMATION}
\end{array}
$$

Figure 2. Knowledge, Information and Data

manner, the assimilation of new knowledge, which, in turns, helps in meeting information needs that would otherwise only be met at a much higher cost, much later or perhaps not at all.

With reference to Figure 1, assume that data is available about the competences and education of available personnel. In a classical database setting, a query for employees `E` that are competent in data mining (i.e., `?- competent(E, data_mining)`) would return an empty set unless some matching fact had been explicitly asserted as a data item. With deductive capabilities, one may extend the stock of information by asserting knowledge of implicit data (e.g., via a rule `competent(E, data_mining)` ← `competent(E, databases), competent(E, machine_learning)`). Using this rule, the same query above might return as answers employees that are now seen to be competent in data mining because they are competent in two areas that feed into it. Thus, deductive capabilities increase information stocks via user-supplied knowledge.

However, relying on users to supply all the knowledge required to meet information needs is slow, costly and cannot keep pace with changes in KID stocks. Inductive capabilities semi-automate the supply of knowledge thereby speeding up the process, reducing its cost and making it viable to try and keep pace with changes. Assume, for example, that `?- competent(E, software_agents)` returns an empty set and, moreover, users are not themselves knowledgeable enough to specify a derivation rule to conclude on such competences on the basis of competence in other areas. With inductive capabilities one can attempt to infer the derivation rule on the basis of other data. For example, one might set up an inductive task to augment the definition for `competent/2` in terms of knowledge of programming languages (available as data) plus collaboration with research centres known to be active in certain research areas.

The inductive task might be characterized as follows: *derive rules for* `competent/2` *in terms of* {`employee/4, involved_with/2, affiliation/2, author/2, competent/2`}. Background knowledge is input in two steps. Firstly, the set of concepts, in terms of which a definition for `competent/2` is sought, is provided as a set of rules (e.g., one might define `involved_with(E, P)` ← `(employee (E, _, _, P) ; affiliation(E, P) ; (co_author(E, Z), affiliation(Z, P))))`, then `co_author/2` is defined in terms of `author/2`, and this, in turn, is declared to be defined extensionally, and so on. Secondly, extensions (e.g., for `author/2, affiliation/2`) would be input (e.g., sourced from electronic bibliographies). Finally, evidence is provided as training data. i.e., positive (and, possibly, negative) examples of people competent in software agents (e.g., sourced from research centre web pages). An inductive engine might then produce a definition such as `competent(E, software_agents)` ← `employee( E, _, java, _), involved_with(E, ibm_tokyo)`. The new knowledge, based on the fact that IBM Tokyo has developed software agent technology in Java, hypothesizes that employees who are knowledgeable in Java and have been involved with IBM Tokyo are likely to be competent in software agents. Augmenting the definition of `competent/2` with this new rule may change the answer to `?- competent (E, software_agents)` from the empty set to one populated with employees satisfying the conditions on the right-hand side of the hypothesized rule above.

This section has shown, through a small example, how extending the class of supported functionalities from one comprising only query answering and information extraction with one that, in addition, includes knowledge creation increases (and dynamically adjusts) information supplies. The overall effect of KIDEs is to meet more information needs than could be met before. Since such gains were possible through a semi-automated process, it is likely that this specific information need was met more cost-effectively than before.

5. KIDEs in Virtual Organizations

This section provides an abstract characterization of VOs [1, 5, 20] in order to exemplify how KIDEs might make a significant difference in important contexts.

VOs exhibit a number of distinctive characteristics, including:

- a VO is a cooperative network of organizations;
- VO members retain their legal and economic identity and independence;
- a VO is created in response to a market opportunity and exists while that opportunity lasts;
- the purpose of a VO is to embody an optimal configuration for the exploitation of members' competences and resources with a view to satisfying the requirements of the market;

- VO members can come from the same or from different economic sectors;
- a VO may have a core member (and perhaps a star-like topology) or not (and probably a ring-like topology);
- VOs are founded on trust, and since life-span is limited, this trust typically has to arise from open protocols, careful matches, and few members, among other factors; and
- a VO is often inconceivable without information and telecommunication technologies.

The dependency of VOs on source integration systems has been noted [18]. This section aims to add evidence to the argument in [18] by showing that VOs, particularly in their set-up stage, would benefit from the unified approach to KIDM embodied by KIDEs.

A brief abstract characterization of VOs now follows. A **VO** can be understood conceptually as a configuration of *competences* and *resources* to meet a set of *requirements*. This set of requirements is referred to as a **market opportunity**. Given a set of identified requirements, resources and competences are sourced from each **member organization** $o \in O$, where O is assumed to be an extensible **pool**. Thus, if one denotes the set of available competences by C, the set of available resources by S and the set of identified requirements by R, a member organization o is characterized by the competences and resources that can be sourced from it, i.e., $o \subseteq 2^C \times 2^S$ and a VO V is characterized by an association of requirements with organizations (equivalently, to sources of competences and resources), i.e., $V \subseteq R \times O$. In the crucial set-up stage of the VO life-cycle, the main goal is to leave no requirement $r \in R$ unassigned. Roughly, if some r cannot be assigned some $c \in 2^C$ or $s \in 2^S$ then one assumes it is either because it cannot be sourced from any member of O or because the knowledge of C or S derivable from O is imperfect. Two possible actions then are: either to increase the membership of O with organizations that have the missing competence $c' \notin C$ or resource $s' \notin S$, or to increase the membership of C to $C \cup \{c'\}$ or of S to $S \cup \{s'\}$ for some $o \in O$, i.e., while keeping O constant. In KIDE terms, the actions above can be described as follows: either increase the stock of data so as to identify a new $o' \notin O$ such that o' possesses c' or s', or increase the stock of knowledge in such a way as to make it possible to identify some $o \in O$ that possesses c' or s', or both.

KIDEs seem a good candidate for a core role in the KIDM function of VOs because:

- they can be deployed over networks to identify (via induction) market opportunities as a set of requirements;
- once an opportunity is identified, KIDEs can be used to identify (via induction) possible members by matching resources and competences with the requirements identified;
- in this way, they (semi-)automate the crucial set-up stage in the life-cycle of a VO;
- they correspond to, and become identified with, the VO by being its unified information source, thereby contributing to openness, sharing and trust;
- they can be set up and wound down relatively fast and at relatively low cost (because the engines are generic, the interfaces are standard, and they operate over networked data); and
- they can be set up and wound down without intrusion and disruption to the KIDM infrastructure of individual VO members (because they are loosely coupled to the latter's hardware and software environment).

Consider the following case of VO set-up. One company $o_1 \in \{o_1, o_2\}$ identifies a market opportunity $M = \{r_1, r_2\}$. To meet r_1, the competences required are $C_1^M = \{c_1, c_2, c_3\}$ and the resources required are $S_1^M = \{s_1, s_2\}$. To meet r_2, the competences required are $C_2^M = \{c_1, c_4\}$ and the resources required are $S_2^M = \{s_3\}$. Now, it is known that $o_1 = \langle\{c_1, c_2\}, \{s_2\}\rangle$ and that $o_2 = \langle\{c_1, c_3\}, \{s_1\}\rangle$. Thus, unless the range of known competences and resources can be extended, the pool will have to be extended.

Firstly, one might use a KIDE over $\{o_1, o_2\}$ to try to identify the missing resource $\{s_3\}$ in either o_1 or o_2. Assume that from the available knowledge and data items, it is impossible to derive the information that $\{s_3\}$ is available in the pool. Secondly, one might use a KIDE over $\{o_1, o_2\}$ to try to identify the missing competence $\{c_4\}$ in either o_1 or o_2. Assume that, using a combination of deductive and inductive processes analogous to those described in Section 4, this succeeds. Thirdly, if so, then one might use a KIDE over the network N to identify a new, potential pool member $o_3 \in (N \setminus \{o_1, o_2\})$ (and similar ones) who is known to have resource s_3. Although by source integration one might fail to identify that o_3 is a suitable potential new entry to the pool (e.g., because o_3 never explicitly asserted that it possesses resource s_3) and by deductive processes alone one might not be able to conclude it does (e.g., because no available knowledge item substantiates the conclusion that o_3 possesses s_3), again, by analogy with the example given above, the availability of inductive inference might be able to generate a new knowledge item that supports the conclusion that o_3 possesses s_3. Fourthly, if this is so, then a VO v for M is configurable from $\{o_1, o_2, o_3\}$. Thus, a KIDE might make it possible for a VO to be set-up that would not have been possible using classical approaches to source integration, even extended with deductive inference, without greater cost and effort. Therefore, the availability of inductive inference makes a qualitative difference to how many market opportunities it might be possible for aspiring VOs to take advantage of.

6. Conclusions

This paper has presented a proposal to integrate deductive databases and ILP and to deploy the combined inference engine over networked sources the interaction with which can be mediated using a wide range of logic-programming inspired techniques. The unique advantage provided by this approach is to use semi-automated methods to handle knowledge creation in such a way that it can be assimilated more easily and hence boost the ability of KIDM tasks to reach higher throughput (or, conversely, quicker life-cycles). The paper did not report on completed research, but rather attempted to contribute a position whose long-term implications may lead to a more principled, well-founded generic set of solutions to the problems arising from the need to ensure that KID are served to applications in the quantity and quality needed. The expectation is that organizations might avoid being dragged into managing a bottleneck of information needs due not, as in the past, to low throughput in the deployment of information systems that can process existing stocks of data and knowledge, but rather to low throughput in the assimilation and dissemination of more, and better quality, data and knowledge items into and out of existing stocks.

References

[1] W. Appel and R. Behr. Towards the Theory of Virtual Organisations: A Description of Their Formation and Figure. *VoNet - Newsletter*, 2(2):15–35, 1998. `http://www.virtual-organization.net/`.

[2] K. R. Apt, V. W. Marek, M. Truszczynski, and D. S. Warren, editors. *The Logic Programming Paradigm: A 25-Year Perspective*. Springer-Verlag, 1999. ISBN 3-540-65463-1.

[3] S. Bell and P. Brockhausen. Discovery of Data Dependencies in Relational Databases. Technical Report LS-8 Report 14, Computer Science Department, University of Dortmund, 1995.

[4] H. Blockeel and L. D. Raedt. Inductive Database Design. In Z. Raś and M. Michalewicz, editors, *Proc. 10th Int. Symp. on Methodologies for Intelligent Systemms*, number 1079 in LNAI, pages 376–385. Springer-Verlag, 1996.

[5] R. Bultje and J. van Wijk. Taxonomy of Virtual Organisations, Based on Definitions, Characteristics and Typology. *VoNet - Newsletter*, 2(3):7–19, 1998. `http://www.virtual-organization.net/`.

[6] S. Ceri, G. Gottlob, and L. Tanca. *Logic Programming and Databases*. Springer-Verlag, 1990. ISBN 3-540-51728-6, 0-387-51728-6.

[7] S. K. Das. *Deductive Databases and Logic Programming*. Addison-Wesley, 1992. ISBN 0-201-56897-7.

[8] A. A. A. Fernandes. Combined Inductive and Deductive Engines for Knowledge Management. Extended Version of [9]. Available at `http://www.cs.man.ac.uk/~alvaro/kide-1.ps.zip`, 2000.

[9] A. A. A. Fernandes. Combining Inductive and Deductive Inference in Knowledge Management Tasks. In *11th Intl. Workshop on Database and Expert Systems Applications*. IEEE Computer Society, 2000.

[10] P. Flener and S. Yilmaz. Inductive Synthesis of Recursive Logic Programs: Achievements and Prospects. *Journal of Logic Programming*, 41:141–195, 1999.

[11] P. Langley. *Elements of Machine Learning*. Morgan Kaufmann, 1996. ISBN 1-55860-301-8.

[12] N. Lavrač and S. Džeroski. *Inductive Logic Programming: Techniques and Applications*. Ellis Horwood, 1994.

[13] J. W. Lloyd. *Foundations of Logic Programming*. Springer-Verlag, 2nd., extended edition, 1987. ISBN 3-540-18199-7.

[14] Y. Loyer, N. Spyratos, and D. Stamate. Computing and Comparing Semantics of Programs in Four-Valued Logics. In *Proc. Mathematical Foundations of Computer Science (MFCS'99)*, number 1672 in LNCS, pages 59–69. Springer-Verlag, 1999.

[15] J. Minker. Logic and Databases: A 20 Year Retrospective. In C. Z. D. Pedreschi, editor, *Logic in Databases, International Workshop LID'96*, volume 1154 of *LNAI*, pages 3–57. Springer-Verlag, 1996. ISBN 3-540-61814-7.

[16] S. Muggleton and L. D. Raedt. Inductive Logic Programming: Theory and Methods. *Journal of Logic Programming*, 19/20:629–679, 1994.

[17] S.-H. Nienhuys-Cheng and R. de Wolf. *Foundations of Inductive Logic Programming*, volume 1228 of *LNAI*. Springer-Verlag, 1997. ISBN 3-540-62927-0.

[18] A. D. Preece, K.-Y. Hui, and P. M. D. Gray. KRAFT: Supporting Virtual Organisations through Knowledge Fusion. In *Artificial Intelligence for Electronic Commerce: Papers from the AAAI-99 Workshop,*, pages 33–38. AAAI Press,, 1999. Technical Report WS-99-01.

[19] L. D. Raedt. Logical Settings for Concept-Learning. *Artificial Intelligence*, 95(1):187–201, 1997.

[20] G. Wiederhold. Mediators in the Architecture of Future Information Systems. *IEEE Computer*, pages 38–49, March 1992.