INSTITUT FÜR INFORMATIK
Lehr- und Forschungseinheit für
Programmier- und Modellierungssprachen
Oettingenstraße 67, D–80538 München

Ludwig
Maximilians–
Universität
München

On Knowledge Assimilation in Deductive Databases

Hendrik Decker

full paper appears in B. Freitag, H. Decker, M. Kifer, A. Voronkov (eds),
Transactions and Change in Logic Databases, Springer LNCS 1472, 1998
http://www.pms.informatik.uni-muenchen.de/publikationen
Forschungsbericht/Research Report PMS-FB-1998-10, July 1998

On Knowledge Assimilation in Deductive Databases

extended abstract

Hendrik Decker
Institut für Informatik, LMU München, hdecker@informatik.uni-muenchen.de

Abstract The full version of this extended abstract, entitled *Some Notes on Knowledge Assimilation in Deductive Databases*, appears in [FDKV]. The full paper provides a background for studies in the field of knowledge assimilation. Rather than presenting a formal theory or a complete survey, it provides a rhapsodic, phenomenological characterization of some typical issues of knowledge assimilation. This extended abstract summarizes the main chapters, and also contains some material which is not included in the published version.

Introduction

Deductive databases (abbr.: DDBs) are conceived as logic theory systems that change over time. State changes are triggered by update requests, which ask for the assimilation of new or revised knowledge into the database. Knowledge Assimilation (KA) supports the incorporation of actualized knowledge. View updating in relational databases is a special case. In the full paper, we discuss various tasks of KA, particularly those related to the integrity-preserving satisfaction of update requests, and also the use of abductive logic programming (ALP) for KA. ALP homogenizes queries and updates: Queries can be interpreted as update requests; inferred hypotheses which justify computed answers can be interpreted as updates which satisfy the requests. To a considerable extent, ALP implements KA on the level of the logical theory embodied by the database, i.e., not just on top of some general purpose programming paradigm (no matter if that is embodied by an imperative or declarative or whatever kind of programming language).

KA provides mechanisms to assist the adequate incorporation of new or revised knowledge into an existing assemblage of data. The aim of KA is, on one side, to satisfy the requests (i.e., short term intents) of the user, and on the other, to maintain integrity of the stored knowledge across state changes (i.e., long term intentions of the database designer). The field of KA was identified in [Ko1]. The latter contains an abstract description of KA for dealing with the dynamics of evolving databases. In [Ko2], that description is further refined, and abduction is mentioned as a means to generate hypothetical assumptions for accommodating new information into the database.

In the three main sections of the full paper, some typical, mutually related issues in the fields of DDBs, KA and abduction are characterized. Particular attention is paid to the various kinds of hypotheses used for implementing knowledge assimilation by ALP. In this extended abstract, an overview of each of the three main sections of the full paper is given. Moreover, a section with remarks about some problems of knowledge representation (which precedes KA, according to the scenario of knowledge engineering sketched in 2.1) are included in this paper.

1 Deductive databases

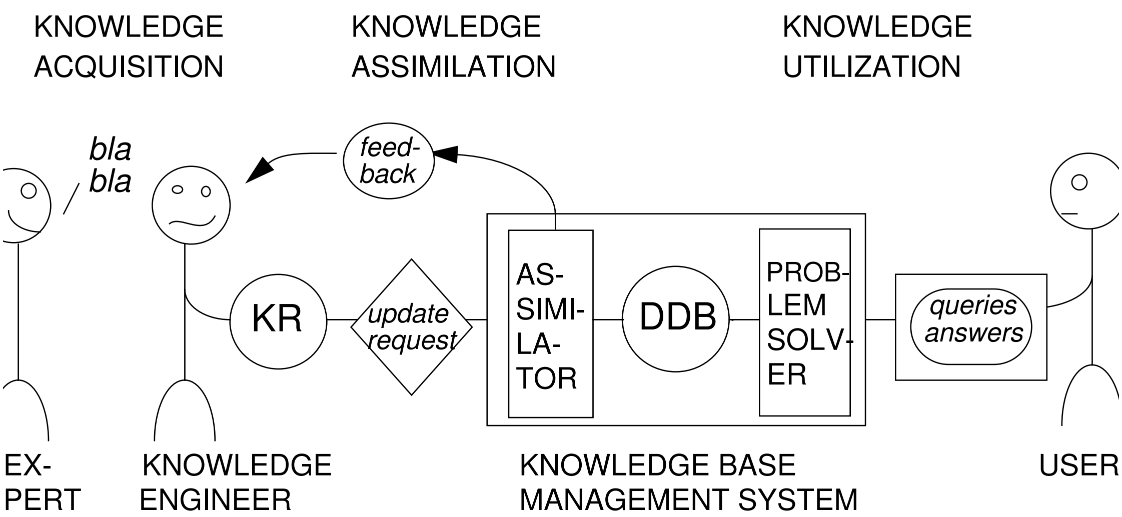
It can be said that the field of deductive databases has developed out of two roots: Question answering by theorem proving [GR] and relational databases [Co]. Both share the use of a declarative, symbolic language for data description and queries, which is independent of the machine-internal representation and organization of the data. Relational databases can be seen as a special case of DDBs.

DDBs combine the quantitative potential of conventional databases for handling large amounts of factual data with more qualitative aspects of rule-based expert systems for intelligent symbol processing. In the full paper, we recall some of the advantages of using logic programming (LP) for databases.

2 Knowledge assimilation

2.1 A knowledge engineering scenario

In the scenario of knowledge engineering (KE) as sketched below, KA is sandwiched between phases of knowledge acquisition and representation on one side, and knowledge utilization on the other. Similar scenarios have been outlined in [K+] (with a broader understanding of 'knowledge acquisition' and a roughly synonymous use of '– assimilation' and '– accommodation'), [M+] and others.



The players (agents) are either human (expert, knowledge engineer, user) or programmed (knowledge assimilator, problem solver). Each of the human actors could also be imagined as a programmed agency. For example, the expert might as well be thought of as one of several, possibly heterogeneous sources of semi-structured information (e.g., a collection of linked pages on the world wide web). The user may as well be a sophisticated analytical process, e.g., for periodically rolling up furbished data aggregates. Even the decisions, actions and reactions of the knowledge engineer for KA might be mechanizable to a large extent (while KR is more difficult; cf. 2.3). The role of any group of individual agents in the scenario can overlap (e.g., the knowledge engineer may also be a user), or be played by a single agent (as, e.g., in a small, burgeoning enterprise where one person does it all). In fact, one of the main achievements of ALP for KA is that the roles of assimilator and problem solver can be accomplished by a single abductive proof procedure which uniformly tackles both tasks.

The props (data) of the scenario are the expert's knowledge, usually transmitted in narrative or semi-structured form; its symbolic representation (KR) by the knowledge engineer; its (abstract) machine representation, in the form of a DDB, essentially consisting of facts, rules and integrity constraints; the user's queries and problems, communicated via a terminal; and the answers (solutions) computed by the problem solving agent of the knowledge base management system (KBMS).

The roles (actions, processes) performed by the actors in the plot: Elicitation of knowledge and learning about a domain, either directly from an expert or by structured domain analysis, is known as *knowledge acquisition*. The formalization of acquired knowledge in a symbolic language is called *knowledge representation* (KR). At the other end of the scenario, *knowledge utilization* means to employ the problem solver (i.e., a query answering module) for satisfying the user's requests. *Knowledge assimilation* is accomplished via an interactive dialogue between knowledge engineer and database assimilator. The latter reacts to update requests of the former, who intends to revise existing database knowledge, and possibly revises her own intentions upon feedback from the assimilator (who, e.g., may signalize that a given update request would lead to redundancy, or integrity violation). Also the end user may have the option, not just to query, but also to update her views of the database. Other kinds of updates, e.g. schema updates, are usually reserved to the knowledge engineer. More about the KA players' roles can be found in the full version.

2.2 Induction, abduction and deduction for KE

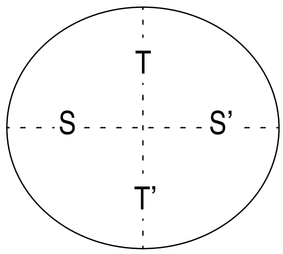
Automation of knowledge acquisition can be supported by methods of learning and *induction*. Knowledge utilization mainly relies on the *deduction* of answers to queries. *Abduction* and ALP is of use for knowledge acquisition and representation (e.g., natural language understanding, perception modeling, vision and, more generally, theory formation), and for knowledge utilization (e.g., non-monotonic inference of default answers and query-the-user facilities). Sections 3 focuses on the use of abduction for KA. The next subsection, though addressing an important prerequisite of KA, is peripheral to the subject of this paper. It addresses the pragmatics and some fundamental problems of knowledge representation (KR). We return to KA proper in 2.4.

2.3 Difficulties of KR

KR comprises what is often called *database* (*schema*) *design*. KR is perhaps the most difficult role to play in the KE plot, since there is no ready-made computational logic formalism for implementing it. KR can be assisted but hardly replaced by mechanical tools. Typical problems of KR: What are the appropriate entities and relationships of the world to be modeled by the database? What to represent by either predicate symbols or constants or functional terms? What to represent by facts or rules or integrity constraints? Solving such kind of problems usually involves a good deal of pragmatics and requires a reasonable intuition, guidance by experience and a sense of adequacy — much easier said than done... In general, the identifiable perception of relevant classes of objects as precisely circumscribed, discrete entities of interest is not necessarily evident nor immediate. To get started with KR in terms of LP and DDBs, cf. [Ko2] [Dl].

Database design theory helps to identify and avoid redundancies and other anomalies of KR for databases (cf. [Ul]). However, in practice, perfect normalization of database schemata is not always feasible. Moreover, in data warehouses, redundancy and other deviations from normal forms are the rule, rather than exceptions. Anyway, normalization can come in only *after* the designer has answered the more basic KR questions, i.e., which should be the entities and relationships for modeling the universe of discourse.

The relative arbitrariness of adequate conceptualizations already surfaces in the following simple example. Suppose the world to be modeled consists of a set W of finitely many objects of different kinds. A partition of W into two disjoint subsets be considered adequate. Assume it is at the designer's discretion to opt for one of several possible partitions, e.g. $W = S \cup S'$ or $W = T \cup T'$, as pictured below (where W is represented by the circled area). Each of S, S' or T, T' can be conceived as a predicate which holds of precisely those object individuals that have the respective property. For instance, a world W of vehicles may be divided into either two- and four-wheeled vehicles, or into vehicles driven by muscle and engine power.



A more refined modeling of W (e.g. by predicates corresponding to the four segments above, capturing a distinction of bicycles, motor bikes, carriages and motor cars, or by a differentiation into electricity-, gas-, diesel-, horse- and pedal-driven vehicles) may be considered unnecessarily fine-grained and/or too costly to maintain. Pragmatic decisions then have to be taken for classifying bikes with only a weak motor which forces bikers to pedal for speeding up or going uphill. Also the classification of bikes with additional lateral wheels for beginners may pose a problem. Or, what with a van with three wheels only? The reader may wish to fancy more dainty design issues.

The arbitrariness of schema design also suggests that any attempt to weigh knowledge in terms of its semantic significance is possibly fastidious and fallible. In particular, the notion of minimality of changes deserves critical reflection. Minimality is often required for choosing among several possible solutions for satisfying update requests. Two possible ways to measure minimality is to either compare the number of clauses (usually basic facts) in updates, or the number of inferrable consequences. However, besides being very different, both often mean nothing much better than to measure haphazard or prejudiced choices of the database designer. Moreover, minimality may conflict with database evolution, since minimality is measured only relative to present or past states, but not wrt future updates (cf. [D1]). Similar comments apply to other, related measures of minimality and simplicity, as discussed, e.g., in [CP] [NM]. Rather than using questionable quantifications of knowledge, integrity constraints can (and should) be used as filters for choosing among several update candidates, whenever possible.

A related difficulty of designing an "intelligent" deductive database (e.g., for supporting rational decision making) is to determine an adequate degree of refinement (depth and breadth) of rules by which database facts are reduced to their basic ingredients (causes and components). Eventually, it might well be that the basicness of logical causes and the atomicity of basic components of objects is but an ontological myth, or even a false belief. After all, in physical reality, there seems to be no such thing as most fundamental causal reasons or most basic elements. Rather, physical reality is perhaps best approximated rationally by quantum logic. Naively, some of the latter can perhaps be modeled by circular rules such as

$$\mathrm{corpuscule}(x) \leftarrow \sim\mathrm{wave}(x),$$
$$\mathrm{wave}(x) \leftarrow \sim\mathrm{corpuscule}(x),$$

rather than by a non-circular hierarchy of clauses. The age-old deductive paradigm of structural sciences seems to have seduced scientists to always search for first principles from which everything else could be unfolded. But that search might well be a never-ending story, as suggested by modern physics.

Anyway, us database theoreticians and practitioners are usually happy to assume "natural" hierarchies of the domains at hand, in order to model them by rules recurring to basic extensional facts. Good schema design intends to model an abstraction of our perception of the world (which in itself already is an abstraction of reality). In most practical cases, we get away with pragmatic design decisions by using hierarchical structures that are deemed good enough for the given application. In the end, however, they may not faithfully represent reality. Thus, it is sometimes useful to remind ourselves that database schemata (and logic or other formalisms in general) are, at best, an approximation of perceivable structures of reality, but never more.

Finally, we emphasise that the difficulties with logic as a formalism to support the symbolic representation of knowledge can hardly be overcome by compromising logic in favour of other approaches that are less amenable to formal reasoning. After all, whenever any artificial, human-made system works in some intelligible way, its behaviour can in principle be conceived rationally, and hence can be re-engineered logically.

2.4 The relationship of KA to the deductive structure of the database

According to similar distinctions in [Ko1–3], knowledge assimilation can be related to a given database by several cases: The data to be assimilated is either already

1) true, or

2) neither true nor false, or

3) false

in the current database state. Case *2)* means that the current database is consistent with the data to be assimilated, i.e., integrity is not violated when the latter is assumed to be true. Case *3)* means inconsistency, i.e., integrity is violated, but perhaps with a chance of repair. According to *1)– 3)*, the tasks, and thus the possible outcome of KA, are as follows.

For *1)*, nothing has to be changed, i.e. the current state may persist. However, a (possibly singleton) set of clauses to be assimilated might as well be added to the database, in spite of their redundancy, if the knowledge engineer wants to have it that way, e.g. for reasons of efficient access to the new information.

For *2)*, KA infers consistent hypotheses in the form of extensional facts which, when added to the database, entail the data to be assimilated. In the simplest case, the hypotheses are the data to be assimilated themselves. However, if, e.g., these data are expressed as a view update request, then the assimilator is expected to compute hypotheses of extensional facts from which the view is derived, such that adding the hypotheses as updates makes the view update request derivable and thus satisfies the request.

For *3)*, facts which contribute to integrity violation have to be revised such that the update request can be satisfied without integrity violation, if possible at all. For that aim, KA attempts to compute hypothetical insertions or deletions of extensional facts which, when executed, lead to the required outcome.

In the full paper, we discuss at some length the KA issues of updating, integrity and belief revision. Subtopics of updating are redundancy, minimality and multiplicity of solutions for satisfying update requests, etc. Subsections of integrity address the topics of syntax and semantics of integrity constraints, integrity checking, integrity maintenance, satisfiability of integrity constraints, security, etc.

For belief revision, hypothetical reasoning by ALP is the key technology. In general, hypothetical reasoning is needed to check the consistency of updates or hypotheses for satisfying update requests, and for sounding out consequences of hypothetical facts that need to be assumed if updates would otherwise violate integrity. Also the handling of exceptions to database rules may require hypothetical reasoning. In a way which is usually not explicitly noticed, hypothetical reasoning also takes place for reasoning with default assumptions in the database (cf. 3.1). For instance, that is the case for negative facts which are consequences of the well-known closed world assumption. In the full paper, we relate the three cases of KA as distinguished above to corresponding kinds of hypotheses in ALP. We come back on this issue in section 3.3.

3 Abduction

In dictionaries, entries for *abduction* are somewhere between 'abdomen' and 'aberration'. Well, perhaps abduction is gutsy, and in general, the term abduction has irritatingly diverse connotations of medicinal, criminological and technical kind. But it sure is not as eccentric as might be suggested by the prefix 'ab-', which means 'departing from' or 'outside of'. Rather, logical abduction has been characterized as common-sense reasoning to hypothetical but plausible explanations and to justifications of default conclusions (cf. [KKT]). Thus, it captures what is tacitly going on almost all the time in everyday life. Applications of abduction where inferred hypotheses do not necessarily hold by default are diagnosis, belief revision and KA.

3.1 Abduction and default reasoning

Consciously or unconsciously, all kinds of reasoning seem to rely on some basic tacit default assumptions. For instance, common sense has it that the vast majority of things remain the same whenever a particular event which changes a part of the world occurs. More fundamentally, the persistence of basic laws of perception (that even can be attributed to animals [Lo]) must be believed in, in order to make sense out of anything under consideration. In general, default reasoning borders on, and probably overlaps with prerationality, and continues to be a matter of ongoing debate (cf., e.g., [Br] [Sh]). In the stylized, discrete framework of LP, default reasoning by abduction has been dealt with in [Po] [KKT] and others. A major influence on that work has been the work reprinted in [Li], and [Re]. In ALP, hypotheses by which default conclusions are tacitly or subconsciously justified are made explicit, such that they can be rationally reflected and reasoned with, as arguments in a disputation (cf. [DKT]).

For KA, default reasoning is particularly interesting: KA needs to assume hypothetical updates with which desired changes of the database can be satisfied without violating integrity. Clearly, hypotheses that hold by default are preferable to non-default hypotheses. Default inference is needed to check if an assumed hypothesis holds by default or not. As outlined in [KKT], ALP can be used for default and non-default inference. As already indicated above, we propose a differentiation between default and non-default hypotheses in the context of KA and ALP in the full paper.

3.2 Abduction and KA

Abduction was introduced by Peirce [HW]. Deduction infers consequences from facts via if-then rules. Induction generalizes from individual facts to rules. Abduction reasons from given facts via rules back to possible causes that hypothetically explain the facts. The syllogisms below illustrate the principles of deduction and abduction. For simplicity, they are given in propositional form. That ignores the importance of term unification, thus neglecting that the results (below the bar) inferred by deduction and abduction are *specializations* of corresponding atoms in the premises (above the bar), while inductive inference always results in *generalizations* of premises.

Deduction (forward)	Abduction	Deduction (backward)
B	A	$\leftarrow A$
$A \leftarrow B$	$A \leftarrow B$	$A \leftarrow B$
----------	----------	----------
A	B	$\leftarrow B$

Forward deduction infers the consequence A from B via the rule $A \leftarrow B$. Abduction infers B as a possible cause for A, i.e., the hypothesis B explains A. Backward deduction is the well-known inference principle of LP. The structural similarity of syllogisms for deduction and abduction indicates that abduction can be easily simulated by, and thus integrated into, normal logic programming, which yields its extension to ALP.

The syllogism above characterizes abduction syntactically. The usual semantic characterization of abduction in databases is as follows (cf. [KKT]). For a database D, a set IC of integrity constraints and a fact F, an *abductive explanation* (or, *justification*) of F is a set Δ of hypothetically assumed ground facts about extensional predicates, such that

$\Delta(D) \models F$ (i.e., F is a logical consequence of $\Delta(D)$), and

$\Delta(D)$ satisfies IC.

For default reasoning, $\Delta(D)$ is the union $D \cup \Delta$. Consistency of $\Delta(D)$ is obtained by imposing the law of contradiction, expressed by distinguished integrity constraints of the form $\leftarrow p(X) \,\&\, {\sim}p(X)$, for each predicate p (X is a vector of n distinct variables, where n is the arity of p).

For KA in DDBs, F represents an update request and Δ is a set of ground literals, i.e., positive or negative facts. $\Delta(D)$ is obtained by adding each positive fact in Δ to D and deleting, for each ${\sim}A$ in Δ, the fact A in D, such that F is true in the resulting new state $\Delta(D)$. "$\Delta(D)$ satisfies IC" expresses that integrity must remain satisfied in the updated state.

In general, abduction is a way to infer hypotheses which, if assumed to hold, contribute to derive a given fact. That fact represents an observation to be explained or assimilated, or, more generally, a goal to be accomplished. The most common form of abductive inference is default reasoning (cf. 3.1). Abduction is frequently used also for diagnosis: Given a symptom of disease or abnormal behaviour, something which potentially causes the symptom (e.g., a virus or a stress factor or an accident) is hypothetically assumed in order to explain the symptom.

Abduction for KA has first been mentioned in [Ko2]. For the related field of diagnosis, an abductive resolution procedure was proposed in [CP]. The start of ALP can be identified with [EK]. In [KM] [D2,3] and others, EK is extended to implement KA. The facts to be abductively explained in KA are update requests, i.e., formulæ required to be made true if they currently are not. Essentially, the class of potential hypotheses in EK, which consists of negative ground facts only, is enlarged in [KM] by allowing also for positive extensional ground facts to be hypothetically assumed, in order to satisfy update requests. In [D2,3], arbitrary integrity constraints are integrated into the abductive process, for filtering out unwanted solutions.

In terms of KA, the syllogism of abduction, above, reads as follows: The update request *insert* A can be satisfied by inserting B. Conversely, the request *delete* A is satisfied by deleting B, as well as sufficiently many other possible causes for A, as given by other rules with head A. In more detail: The request *insert* A can be satisfied by reducing it to the problem of satisfying the request to insert B. If B is a conjunction of extensional facts, then adding ground instances of these facts to the database obviously satisfies the original request. Otherwise, the request to insert B is further reduced via rules that define the predicates in B to more basic causes which, when finally added to the database, satisfy the request. Conversely, the request *delete* A can be satisfied by removing sufficiently many extensional facts that take part in the derivation of A. More precisely, for each rule with A in the head, the deletion of the derivability of its body is requested. If the body contains extensional facts, then its derivability is prevented by deleting one of those facts. If there is no such fact, then the body is further reduced to more basic causes, the deletion of which finally satisfies the original request.

3.3 Toward a classification of abductive hypotheses

Abductive hypotheses for satisfying update requests are always positive or negative ground facts about extensional predicates. View updating also follows this convention.

According to cases *1)–3)* of KA (section 2.4), hypotheses can have different relationships with the deductive structure of the database. In this section, we further refine this case distinction. First, we mention two limit cases. One is when a hypothesis is a logical consequence of the database, and thus redundant. That case can be subsumed under *1)*. The other is when a hypothesis is inconsistent with the database, without any chance of repair, which can be subsumed under *3)*. In both cases, the hypothesis is not useful. It should be dropped in case of redundancy, or withdrawn in case of inconsistency. Among useful hypotheses, one can distinguish between default and non-default hypotheses.

Default hypotheses do not have to be realized by an explicit update, just because they hold by default anyway. As opposed to that, non-default hypotheses need to be stated explicitly, in order to justify either a choice among possible updated states, or to prescribe a revision (expansion or contraction) of the current state. Expansion and contraction of knowledge have been introduced as modes of belief revision in [AGM].

The use of default hypotheses can also be related to case *1)*, i.e., the hypothesized formula is already true, by default. Although KA procedures usually make each hypothesis explicit for justifying inferred results, the database does not have to be modified by explicitly adding default hypotheses, since defaults are implicit assumptions which hold tacitly. The well-known negation-as-failure rule (NaF) infers negative facts that can be justified by themselves as default hypotheses. In the general case of nested negation, additional negative hypotheses may be need to justify the inference of a negative fact.

Rules with (positive) default conclusions can always be transformed such that no default inference rule other than NaF is needed. For example, the default rule that normally, birds can fly, can be expressed by fly(x) ← bird(x) & ~abnormal(x). Thus, we can assume that default rules are always represented by database rules with negative default conditions, such that default hypotheses are always represented by negative literals.

Non-default hypotheses are needed for cases *2)* and *3)* of KA. In case *2)* of 2.3, the requested update is consistent with but not true of the database, not even by default. Similarly, a hypothesis can be consistent with but not true of the database. There are two subcases. One is that the hypothesis determines an *alternative*, i.e., a choice among several possible states. In the example of line 3, below, ~p would be an alternative hypothesis to ~q. The other subcase of *2)* is to *expand* the existing knowledge by adding a contingent hypothesis to the database.

In case *3)*, the formula requested to become true is inconsistent with the current state. Thus, the latter must be contracted, if possible, in order to restore consistency. Similarly, an inconsistent hypotheses must either be dropped, or some facts which contradict the hypothesis must be retracted. In general, extensional facts the presence of which prevents to derive the update request, need to be retracted in order to restore integrity and to satisfy the request, if possible. If not, then the request is unsatisfiable.

Below, we summarize the classification of abductive hypotheses as sketched above. For each class, a succinct example illustrates the abductive relationship of some hypothesis (q or ~q) to some DDB ({ } is the empty database). Coherence and contingence of hypotheses, as mentioned in lines 3 and 4, are discussed in the full paper.

	update request	*DDB*	*hypo-thesis*	*deductive relationship*	*abductive relationship*
1	p	{p ← q, q ← }	q	*1)*	logical consequence
2	~p	{p ← q}	~q	*1)*	default consequence
3	p	{p ← ~q, q ← ~p}	~q	*2)*	coherent alternative
4	q	{ }	q	*2)*	contingent expansion
5	~p	{p ← q, q ← }	~q	*3)*	contraction
6	q	{q ← ~q}	~q	*3)*	irreparable inconsistency

Conclusion

We have summarized the main sections of the paper *Some Notes on Knowledge Assimilation in Deductive Databases* [FDKV]. The latter is an extensive but informal characterization of KA in DDBs. In particular, it sketches how extensions of LP with integrity constraints and abduction can help to deal better with the dynamics of KA in DDBs. Among topics that have a strong relationship to KA but have not (or only superficially) been addressed in the full paper are *paraconsistency* and *arbitration* of knowledge.

Paraconsistent logic can be understood as a relaxation of less than intuitive inference principles such as the *ex falso quodlibet* rule (cf. [Bo]). For KA, paraconsistency can also be understood as a pragmatic approach to cope with integrity violation and inconsistency in knowledge bases (cf. the concluding section of the full paper, and [D4]).

Arbitration of knowledge bases has been introduced in [Rv] as an approach to merging two or more sources of knowledge. KA, as understood in this paper, is characterized by a predilection of *new* information, represented by the update argument, over the *old* information of the database state to be updated. At the same time, updating and revision in KA is characterized by a conservative attitude towards the preservation of old information and its consistency, expressed by integrity constraints. Often, this attitude is emphasized by requiring that changes of the old state be *minimal*, according to some appropriate measure. As opposed to that, arbitration of knowledge does not presuppose an asymmetry of old and new information. Indeed, merging two knowledge bases is different from updating one of them, taken as the old state, with the other, taken as an update argument.

In [Su], the problem of merging ("amalgamating") knowledge bases has been approached by an extension of LP in terms of paraconsistency (which somewhat differs from what is intended in [D4]). Arbitration of knowledge bases has also received attention in the AI community, in terms of propositional logic (cf. [LS]). To the current knowledge of this author, no study of arbitration in terms of ALP has yet appeared.

References

[AGM] Alchourrón, Gärdenfors, Makinson: On the logic of theory change: Partial meet contraction and revision functions, Journal of Symbolic Logic 50, 1985.

[Bo] Bobenrieth: Inconsistencias, por que no?, Colcultura, 1996.

[Br] Brown: The Frame Problem in Artificial Intelligence (Proc. of the 1987 Workshop), Morgan Kaufmann, 1987.

[Co] Codd: A relational model of data for large shared data banks, Communications of the ACM 13, 1970.

[CP] Cox, Pietrzykowski: Causes for events: Their computation and applications, Proc. 8th CADE, Springer, 1986.

[D1] Decker: Drawing updates from derivations, ECRC Technical Report KB-65, 1989; short version in Proc. 3rd ICDT, Springer, 1990.

[D2] Decker: An extension of SLD by abduction and integrity maintenance for view updating in deductive databases, Proc. JICSLP'96, MIT Press, 1996.

[D3] Decker: Abduction for knowledge assimilation in deductive databases, in Proc. 17th Int'l Conf. of the Chilean Computer Society (SCCC'97), IEEE Press, 1997.

[D4] Decker: Aspects of paraconsistency in deductive databases, http://www.pms.informatik.uni-muenchen.de/publikationen, 1997.

[Dl] Dahl: Logic programming as a representation of knowledge, The Computer Journal 26, 1983.

[DKT] Dung, Kowalski, Toni: Argumentation-theoretic proof procedures for default reasoning. http://laotzu.doc.ic.ac.uk/UserPages/staff/rak/rak.html, 1997.

[EK] Eshghi, Kowalski: Abduction compared with negation by failure, Proc. 7th ICLP, MIT Press, 1989.

[FDKV] Freitag, Decker, Kifer, Voronkov: Transactions and Change in Logic Databases, Springer LNCS 1472, 1998.

[GR] Green, Raphael: The use of theorem-proving techniques in question-answering systems, Proc. 23rd Nat'l Conf. ACM, Brandon Systems Press, Thompson Book Company, 1968.

[HW] Hartshorn, Weiss (eds): Collected Papers of Charles Sanders Peirce, Harvard University Press, 1933.

[KKT] Kakas, Kowalski, Toni: The role of abduction in logic programming. in Gabbay, Hogger, Robinson (eds), Handbook of Logic in Artificial Intelligence and Logic Programming, Vol. 5, Oxford University Press, 1998.

[KM] Kakas, Mancarella: Database updates through abduction, Proc. 16th VLDB, 1990;

[Ko1] Kowalski: Logic for data description, in Gallaire, Minker (eds): Logic and Data Bases, Plenum Press, 1978.

[Ko2] Kowalski: Logic for Problem Solving, North Holland, 1979.

[Ko3] Kowalski: Problems and promises of computational logic, in Lloyd (ed): Computational Logic, ESPRIT Basic Research Series, Springer, 1990.

[K+] Kitakami, Kunifuji, Miyachi, Furukawa: A methodology for implementation of knowledge acquisition systems, Proc. Int'l Symposium on Logic Programming, IEEE Press, 1984.

[Li] Lifschitz (ed): Formalization of common sense, papers by John McCarthy, Ablex, 1990.

[Lo] Lorenz: Gestaltwahrnehmung als Quelle wissenschaftlicher Erkenntnis, in Lorenz:: Über tierisches und menschliches Verhalten, Gesammelte Abhandlungen II, 1965. See also, Lorenz: Behind the Mirror : A Search for a Natural History of Human Knowledge, Amazon Books, 1978.

[LS] Liberatore, Schaerf: Arbitration (or How to merge knowledge bases), IEEE Transactions on Knowledge and Data Engineering 10, 1998.

[M+] Miyachi, Kunifuji, Kitakami, Furukawa, Takeuchi, Yokota: A knowledge assimilation method for logic databases, New Generation Computing 2, 385-404, 1984.

[NM] Ng, Mooney: On the role of coherence in abductive explanation, Proc. 8th AAAI, 1990.

[Po] Poole: A logical framework for default reasoning, Artificial Intelligence 36, 1988.

[Re] Reiter: A logic for default reasoning, Artificial Intelligence 13, 1980.

[Rv] Revesz: On the semantics of theory change: Arbitration between old and new information, Proc. 12th PoDS, 1993.

[Sh] Shanahan: Solving the Frame Problem, a mathematical investigation of the common sense law of inertia, MIT Press, 1997.

[Su] Subrahmanian: Amalgamating knowledge bases, ACM ToDS 19, 1994.

[Ul] Ullman: Principles of Database and Knowledge-Base Systems, Vol. 1, Computer Science Press, 1988 / 1989.