Deutsches
Forschungszentrum
für Künstliche
Intelligenz GmbH

Research
Report
RR-94-36

Issues in Concurrent Knowledge Engineering: Knowledge Sharing and Knowledge Base Evolution

Manfred Meyer

Oktober 1994

Deutsches Forschungszentrum für Künstliche Intelligenz GmbH

Postfach 20 80
67608 Kaiserslautern, FRG
Tel.: (+49 631) 205-3211/13
Fax: (+49 631) 205-3210

Stuhlsatzenhausweg 3
66123 Saarbrücken, FRG
Tel.: (+49 681) 302-5252
Fax: (+49 681) 302-5341

Deutsches Forschungszentrum für Künstliche Intelligenz

The German Research Center for Artificial Intelligence (Deutsches Forschungszentrum für Künstliche Intelligenz, DFKI) with sites in Kaiserslautern and Saarbrücken is a non-profit organization which was founded in 1988. The shareholder companies are Atlas Elektronik, Daimler-Benz, Fraunhofer Gesellschaft, GMD, IBM, Insiders, Mannesmann-Kienzle, SEMA Group, and Siemens. Research projects conducted at the DFKI are funded by the German Ministry for Research and Technology, by the shareholder companies, or by other industrial contracts.

The DFKI conducts application-oriented basic research in the field of artificial intelligence and other related subfields of computer science. The overall goal is to construct *systems with technical knowledge and common sense* which - by using AI methods - implement a problem solution for a selected application area. Currently, there are the following research areas at the DFKI:

- Intelligent Engineering Systems
- Intelligent User Interfaces
- Computer Linguistics
- Programming Systems
- Deduction and Multiagent Systems
- Document Analysis and Office Automation.

The DFKI strives at making its research results available to the scientific community. There exist many contacts to domestic and foreign research institutions, both in academy and industry. The DFKI hosts technology transfer workshops for shareholders and other interested groups in order to inform about the current state of research.

From its beginning, the DFKI has provided an attractive working environment for AI researchers from Germany and from all over the world. The goal is to have a staff of about 100 researchers at the end of the building-up phase.

Dr. Dr. D. Ruland
Director

Issues in Concurrent Knowledge Engineering: Knowledge Sharing and Knowledge Base Evolution

Manfred Meyer

DFKI-RR-94-36

This work also appears in the Proceedings of the First International Conference on Concurrent Engineering (CE'94), Pittsburgh (USA).

This work has been supported by a grant from The Federal Ministry for Research and Technology (FKZ ITW-9304/3).

© Deutsches Forschungszentrum für Künstliche Intelligenz 1994

This work may not be copied or reproduced in whole or in part for any commercial purpose. Permission to copy in whole or in part without payment of fee is granted for nonprofit educational and research purposes provided that all such whole or partial copies include the following: a notice that such copying is by permission of Deutsches Forschungszentrum für Künstliche Intelligenz, Kaiserslautern, Federal Republic of Germany; an acknowledgement of the authors and individual contributors to the work; all applicable portions of this copyright notice. Copying, reproducing, or republishing for any other purpose shall require a licence with payment of fee to Deutsches Forschungszentrum für Künstliche Intelligenz.

ISSN 0946-008X

Issues in Concurrent Knowledge Engineering: Knowledge Sharing and Knowledge Base Evolution

Manfred Meyer
German Research Center for Artificial Intelligence (DFKI)
P.O. Box 20 80, D-67608 Kaiserslautern, Germany

Abstract

The development of knowledge-based (KB) systems for solving real-world problems has become a very time-consuming and expensive task. Over the last years, more and more people started to investigate how to share and reuse knowledge once it has been formally represented. Moreover, as the world around is changing continuously, the KB system and its knowledge base have to change, too. Thus, recently the issue of knowledge base evolution came up that deals with the continuous improvement of a KB system during its entire life-time starting with the first formalizations and still continuing along its practical use. We discuss the current efforts in knowledge sharing and reuse and then study the knowledge evolution approach, that can be seen more technically as an interleaved combination of knowledge validation and exploration techniques.

Contents

1 Introduction **3**

2 A Software Engineering Approach to Knowledge-based systems **3**

3 Knowledge Sharing and Reuse **4**

3.1 Interlingua 5

3.1.1 Dealing with Different Paradigms 6

3.1.2 Dealing with Different Constructs 6

3.1.3 Dealing with Implicit Control Information 7

3.2 Knowledge Base Communication 7

4 Knowledge Base Evolution **8**

4.1 The KB Evolution Scenario 9

4.2 KB Evolution as Theory Revision 10

4.3 Knowledge Base Exploration 11

4.3.1 Least Generalization 11

4.3.2 Generalized Subsumption 12

4.3.3 Generalization for KB Evolution 12

4.3.4 Abduction 13

4.4 Knowledge-Base Verification 13

5 Conclusion **14**

1 Introduction

The role of knowledge-based (KB) systems in complex applications becomes more and more important. Recognizing the potential of KB technologies, many advanced applications tried to make use of them. Advances in this area have been resulting in the development of many tools and we are now facing an increasing number of KB applications. Today, KB systems are playing a significant role in many organizations. Aside from notable and well-publicized successes, failures of KB systems in real-world applications are not uncommon.

The current state of the practice in the KB software area is not satisfactory. Reasons for failures of KB systems can be discussed from organizational, behavioral, and technological viewpoints [15]. Many of the causes are interrelated and cannot be viewed in isolation. In the focus of our research are issues regarding the technology of *Concurrent Knowledge Engineering*, where KB systems are no longer built in isolation but take part in the integrated product development within a concurrent enterprise.

In this paper, we will first briefly discuss the consequences of supporting the *Concurrent Engineering* (CE) approach by using KB technology. By drawing parallels to traditional *Software Engineering* approaches, in section 2 we will identify two most important issues to be tackled when trying to use KB technology in real-world concurrent enterprises: Similar to the sharing and reuse of software modules, techniques for sharing knowledge bases and reusing the knowledge for different applications have to be developed. Thus, in section 3 we will review and discuss the current efforts towards *Knowledge Sharing and Reuse*. The second issue that comes up when viewing CE from the perspective of *Concurrent Knowledge Engineering*, is the problem of KB maintenance and the validation and verification of KB systems. In section 4 we will present a general framework for *Knowledge Base Evolution* as well as several exploration and verification techniques for supporting the evolution of KB systems over their entire life-time. The paper will then be concluded by summarizing and discussing the central message of this paper: Concurrent Engineering approaches will not succeed without using KB technologies. On the other side, KB technologies will not be able to contribute to Concurrent Engineering without putting intensive effort to Knowledge Sharing and Reuse as well as to Knowledge Base Evolution research.

2 A Software Engineering Approach to Knowledge based systems

Traditionally, many KB systems were demonstration systems for which no long-term or widespread use was contemplated. As a result, many implemented KB systems are fragile. The Software Engineering community, concerned with practical problems of making a project work reliably denied a potential of KB systems to be used for practical applications. On the other hand, in the Knowledge-Based Systems community, formal Software Engineering was viewed as an enterprise with

little chance of success and practical software engineering methods were regarded as a drudgery [25].

However, as well the development in the AI area resulted in KB systems which have shown their practical values, Software Engineering also made advances. It was recognized that the traditional waterfall model is not the solution for the software crisis and that traditional software process models discourage more effective approaches to software development.

The spiral model of software development and enhancement [2] is particularly interesting because of its potential to be applied to the development of KB systems. Prototyping had become the central role in a software process, but this kind of process is different from the evolutionary prototyping which is often advocated as a methodology for the development of KB systems. The main danger with the latter approach is that its lack of discipline often ends in the code-and-fix model and in unreliable and unmaintainable code.

The growing complexity of problems and programs and the inability of ad hoc methods to cope with them led to the development of abstractions and encapsulation concepts enabling the sharing and reuse of software modules in conventional software systems. The same is to be expected for KB systems, too. Encapsulation in the area of KB systems will lead to encapsulation on a higher level capturing not only data, but also knowledge, behavior, and know-how within the specific community in which software is being developed. Additionally, the software maintenance problem also applies to KB systems and opens a new research branch: Together with the issue of knowledge sharing and reuse making knowledge a universally available and portable good, the problem of knowledge base maintenance becomes more and more important: As the world around is dynamically changing, knowledge bases—regarded as a model of at least the relevant aspects of the world around—will also have to reflect these changes and to evolve along their practical use.

3 Knowledge Sharing and Reuse

Recently, some researchers have begun to view KB systems from the standpoint of Software Engineering. Emphasis is laid not only on maintaining knowledge bases and adapting established Software Engineering methods for the development of KB systems, but also facilitating the sharing and reuse of knowledge bases. The goal is to acquire and formulate knowledge of a domain only once and to use it for different applicable problems. There are numerous knowledge sources like research results, medical knowledge, government regulations, product information but also common sense knowledge which are already public and could be useful for many applications if they were available in an inexpensive and convenient way.

However, currently little research is done on how to adapt a knowledge base to uses different than the anticipated ones. In conventional programming similar problems are well known and partially solved by the concepts of modular programming, high-

level programming languages, etc. In KB systems the problems of sharing and reuse are just beginning to be recognized and analyzed.

Traditionally, first-order predicate calculus was proposed as the underlying standard of all knowledge representation formalisms, namely in the seminal papers by McCarthy and Hayes (e.g., [22]). Following the "Knowledge Representation Standards" workshop held in March 1990 in Santa Barbara, California, the topic of knowledge sharing receives more and more attention. This intensified interest was started by an initiative of DARPA (Defense Advanced Research Projects Agency) where for the first time a large group of scientists tries to solve the problems of knowledge sharing and reuse. For this reason, four working groups were established (cf. [24]).

The first group is developing an intermediate language (*Interlingua*) called KIF ("Knowledge Interchange Format") [10]. KIF has the expressive power of predicate logic permitting nonmonotonicity, metaknowledge, and partial definitions. It employs a LISP-like syntax and a kind of declarative semantics. One of the basic problems of knowledge sharing and reuse are the different representations used. The translation of a knowledge base into another representation is facilitated by an Interlingua and compilers to and from the Interlingua for every representation.

The second group analyzes different paradigms of knowledge representations (frames, rules, semantic networks, etc.) in order to develop a Knowledge Representation System Specification (KRSS). Once the characteristics of a paradigm are found, a core system can be defined which has to be subsumed by any knowledge representation language claiming to support this paradigm.

The third group is interested in the standardization of the interface between knowledge-based systems and data-based systems or application programs. By developing a standard Knowledge Query and Manipulation Language (KQML) [9] knowledge-based systems can be used as "black boxes". Each system is formulated in its own specialized internal representation but is nevertheless able to communicate and cooperate with other systems via the KQML.

The fourth group clarifies how a knowledge base has to be developed in order to be sharable and reusable [13]. To accomplish this it is necessary to create common ontologies for domains (e.g., materials, mechanics, medicine) and tasks (e.g., diagnosis, simulation, planning). An ontology is a vocabulary of representational forms with agreed-upon definitions which are readable by people and machines.

In the following we will discuss some ideas regarding knowledge translation and cooperating knowledge bases.

3.1 Interlingua

If several knowledge bases are to be used together, an obvious solution is to merge them. For this it is necessary to compile the knowledge bases from their own representations into a common one. Even if only a single knowledge base is reused it can be necessary to compile the knowledge base into another representation which is more desirable in the new application. However, to write compilers from every

representation into every other means $n^2 - n$ compilers for n representations. An Interlingua obviously reduces the number of compilers to $2n$ [18]. Nevertheless, this approach poses a new problem, namely to find an adequate Interlingua.

3.1.1 Dealing with Different Paradigms

Each knowledge representation paradigm is based on its own view of the world. This view determines which characteristics and properties of the domain are emphasized and which are ignored.[1] Therefore, it is a severe problem to translate a description based on a certain view of the world into another, unless they are equivalent regarding expressiveness. If the equivalence cannot be proven, however, the paradigm of the Interlingua can only be chosen arbitrarily, favoring the knowledge bases of the same paradigm while disadvantaging the others.

A possible solution is the definition of several Interlinguae, one for each paradigm. This facilitates not only the translation between languages of the same paradigm, but it also enables the translation between languages of different paradigms by dividing the translation into three steps: (1) translation from the source language into the Interlingua of the same paradigm, (2) translation into the Interlingua of the other paradigm, and finally (3) translation into the target language. As an advantage of this method the hardest part (the translation between different paradigms) has to be considered only once and can be liberated from language specifics. In this approach, the problem of finding an Interlingua reduces to finding an Interlingua for a certain paradigm.

3.1.2 Dealing with Different Constructs

Even knowledge representation languages of the same paradigm are not only syntactically but also semantically different and can support different constructs. For example, some rule-based languages can express fuzzy knowledge but fuzzyness is not part of the rule-based paradigm itself. It is an open question how a construct can be translated if it is supported by the source language but not by the target language [6, 18].

To define the Interlingua for a certain paradigm it is necessary to analyze and characterize this paradigm. This is exactly what the second DARPA working group does. One of the aims of the KRSS group is to define a core system for each paradigm. This core system is the part which is shared by all languages of this paradigm, i.e. the 'intersection' of the languages.

As an advantage, the intersection provides a canonical form. First, all languages for which a compiler into the canonical form exists can be objectively compared and evaluated. Second, the canonical form provides a uniform method to define the semantics of the constructs. Third, different tools (e.g., for validation and verification,

[1] It is in principle impossible to represent a domain completely due to its complexity (e.g., a complete representation of a physical object could not ignore its atomic structure).

cf. Section 4) can use the canonical form as input and ideally have to be developed only once for all languages. Moreover, the union has a severe disadvantage: Whenever a new construct in any language is developed, not only the Interlingua but also every compiler from the Interlingua into any language must be augmented by this construct. Nevertheless, the DARPA working group takes this direction with KIF by trying to integrate all paradigms and features in KIF. The future will show whether this idea is practically feasible and implementable.[2]

3.1.3 Dealing with Implicit Control Information

In KB systems the knowledge and the inference engine using this knowledge to solve a problem should be independent of each other. This is exactly the advantage of KB systems: The inference engine is a (more or less) general problem-solver applicable for every domain; the knowledge consists of facts, relations, causalities, etc., which are true independently of their usage for a certain problem.

If these ideal requirements are met, not only can an inference engine be used for a great variety of domains but also a knowledge base can be used for different tasks. Theoretically, to plug a knowledge base out of one KB system and into another should pose no problems. However, in practice, special features of the inference engine are exploited when formalizing the knowledge. The knowledge is no longer necessarily 'true' in the logical sense but is formulated to be optimally useful for a certain inference engine in finding a solution for certain classes of problems. For example, for many problems the order in which the steps to reach the solution are performed is important. Every (sequential) inference engine must use the knowledge in some order and has its own strategy to do this, e.g., the order in which the knowledge is written down. Knowledge engineers are tempted to take advantage of this strategy to force a certain order on the usage of the knowledge. Another inference engine with another strategy is therefore not able to use the knowledge successfully.

The trouble is that control information, which is knowledge about the usage of knowledge, is not made explicit, but used implicitly by taking advantage of peculiarities of the inference engine. Instead of hiding knowledge in such a way, it should be exposed because that is what declarative programming demands.

However, most existing KB systems are custom-made for a certain inference engine. Therefore the actual knowledge is so intertwined with implicit control information that it is extremely difficult to separate reusable knowledge from 'programming hacks'.

3.2 Knowledge Base Communication

Apart from the idea of translating a knowledge base in order to share and reuse it, there exists a totally different direction: Each knowledge base is formulated in its own representation, but a common standardized query language is used. Via

[2] A critique of KIF can be found in [12].

this language the system can be asked a question which is answered if possible. As an enhancement, the knowledge base could be also manipulated by means of the standard language. This language should be usable for people, conventional software and also for other KB systems as an interface.

In this approach, a knowledge base is regarded as a database with the ability for deduction [4]. The query and manipulation language of a database language can be relatively simple because data only has to be queried, added and modified. This data can have the same form for all domains (e.g., relational).

The analogous solution for KB systems is to query data or facts too, but not only those explicitly added to the knowledge base but also deduced facts. Such a query language is useful for simple database-like systems. Nevertheless, it is unsufficient for most KB systems, because they contain not only data or facts but also causal dependencies (rules), hierarchical knowledge (classes), procedural knowledge, constraints, etc.

A standard query and manipulation language thus should allow query, add and modify higher-level constructs too. Because this language must be uniform for all KB systems, it is easy to see that similar problems arise as with the Interlingua: Which paradigm, which constructs should be supported by the standard query language, how to represent procedural knowledge declaratively, etc.? In [10] this analogy is recognized and the view is taken that KIF can also be used as a communication language between cooperating agents. The expressiveness of a query and manipulation language and of an Interlingua have therefore to be similar.

4 Knowledge Base Evolution

As KB systems are brought to practical applications and knowledge bases are to be used over years, the problem of knowledge base evolution naturally comes up: The key issue is how to ensure that the knowledge base of a KB system does always represent all the knowledge that is relevant for solving its task, i.e. being *'complete'*, and does not become out of date or invalid, i.e. remaining *'correct'*. Although this is a goal hard to achieve, it shows the direction in which KB evolution research should work: to overcome the (always damned but nevertheless done) accumulation of 'small local hacks' causing unforeseeable consequences and to find a compromise between this ad-hoc knowledge base modification approach and the other extreme of restarting the whole knowledge engineering work ranging from the formal specification down to the concrete representation with each modification of the KB system.

In this section we will present a general KB evolution architecture that basically consists of an interleaved combination of exploration and validation tasks in order to support the evolution of knowledge bases over their entire life-time.

4.1 The KB Evolution Scenario

Knowledge base evolution operates on the knowledge base of a KB system. Thus, for an overall description of knowledge base evolution we distinguish two main units (Fig. 1): the KB system itself (KBS) and the knowledge base evolution system (KES). The KES operates as a metalevel system on the object level KBS. Reasoning

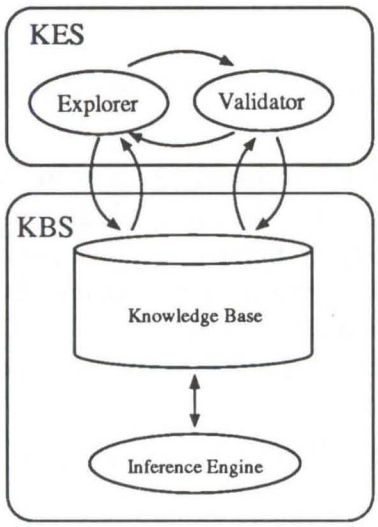

Figure 1: The Knowledge Base Evolution Architecture

in the knowledge evolution system is performed by the exploration and validation components.

The *knowledge explorer* scans the KB in search for interesting patterns. Exploration (left part of Fig. 2) can be seen as an iterative process starting with the generation of a pattern hypothesis, proceeding with a search for the pattern in the KB, and resulting in a possible interactive assimilation of the discovered pattern into the KB. Thus, inductive techniques play a major role for knowledge exploration.

The *knowledge validator* can perform verification and also validation (with appropriate user assistance). It examines the KB to detect structural or functional defects. Validation and verification (right part of Fig. 2) can also be seen as an iterative process starting with the generation of a defect suspicion, proceeding with a check for a defect w.r.t. the suspicion in the KB, and resulting in a possible defect description or repair suggestion. Here, techniques for checking integrity constraints become most relevant.

The iteration cycles shown in Fig. 2 can be arbitrarily interleaved, permitting evolution to consist of dual validation and exploration processes. Together they form a heuristic, approximative process that alternates focusing and processing phases and improves the KB any time a sufficient amount of knowledge for an update (i.e.,

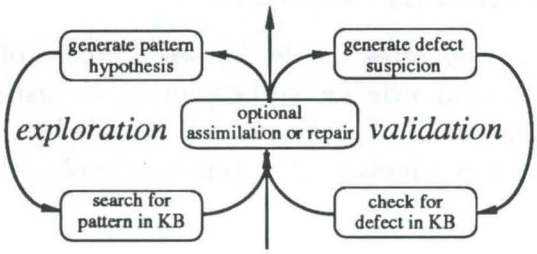

Figure 2: The Evolution Processes

assimilation or repair) has been accumulated within the KES or by the user. For example, assume that the validator has identified a rule whose premises cannot be satisfied in a given KB. The explorer could then, e.g., try to generalize that particular rule or to complete the missing knowledge reachable from its premises. Conversely, after the explorer has discovered a pattern (e.g., a new or generalized rule) the validator may be asked to verify the KB, focused on the assimilated pattern.

4.2 KB Evolution as Theory Revision

The problem of building up a knowledge base (knowledge acquisition) can be seen as a two-phase process (cf. [11]): In the first phase the knowledge engineer builds an initial model. In the second phase this initial knowledge base is refined or revised into a high performance knowledge base. Thus, knowledge base refinement can be regarded as knowledge base *validation* [20].

During the further practical use of the KB system, the dynamically changing world around may cause the knowledge base to become invalid in one of the following senses:

- New developments may cause new problem cases not being covered by the knowledge base. This results in the KBS not being able to solve these problems. For example, neglecting the development of new materials would leave the system incapable to find the best production process.

- Some knowledge stored in the knowledge base may become out of date and should no longer be used as it would lead to solutions that for some reasons are no longer valid in the current application environment. For example, a material that has become known to be noxious should no longer be used in products that people will have every-day contact with.

In the first situation we have a new application case (i.e. a positive example) that is not yet derivable from the knowledge base. In the second second situation, we can derive a specific solution from the knowledge base which is no longer admissible

(e.g. because of new environmental protection laws). This is consequently called a negative example.

From a more formal point of view, this means that a given knowledge-base KB has to be revised using positive examples E^+ and/or negative examples E^- such that all the positive examples but none of the negative examples are covered by the resulting knowledge base KB'. Taking the knowledge base as a Horn theory $T = F \cup R$ consisting of facts F and rules R and satisfying a set of integrity constraints IC, the *exploration task* of theory revision is to change T into T' such that $T \vdash e \forall e \in E^+$ and $T \nvdash e \forall e \in E^-$. The resulting theory T' must, of course, still satisfy the given integrity constraints, i.e. $IC \cup T'$ must be consistent. This integrity checking represents the *validation* or *verification task* of theory revision and thus again demonstrates the interleaved exploration and validation principle.

The main task, however, remains how to obtain the revised theory T'. In principle, there are two possibilities: First, we can modify the rules R, for instance by using generalization or specialization techniques, or we can extend the set of facts F where the additional facts can be found by abduction. In the following section we will study generalization and abduction for knowledge-base exploration in more detail before we discuss knowledge-base validation and verification issues in section 4.4.

4.3 Knowledge Base Exploration

Generalization is one of the basic techniques for KB exploration. Generalization operators perform two basic syntactic operations on a clause: (1) apply an inverse substitution to the clause, or (2) remove a literal from the body of a clause. In this section we will first review the least generalization and generalized subsumption framework defined by Plotkin and Buntine respectively before we extend these techniques for the needs of theory revision in practical applications.

4.3.1 Least Generalization

Least Generalization was originally introduced by Plotkin [26]. It is the opposite of most general unification [27], therefore it is also called *anti-unification*. Given two atomic formulas $p(f(a), x)$ and $p(f(y), b)$, unification computes their most general specialization $p(f(a), b)$ while anti-unification computes their most special generalization $p(f(y), x)$.

In addition to the generalization of literals, Plotkin also describes an algorithm for the least generalization of clauses. A clause C_1 generalizes a clause C_2 (denoted by $C_1 \leq C_2$), if C_1 subsumes C_2, i.e. $C_1\theta \subseteq C_2$. This is also called θ-subsumption [5]. A generalization C_1 of a clause C_2 can thus be obtained by applying a θ-subsumption-based generalization operator. ρ that maps a clause C_2 to a set of clauses $\rho(C)$ which are generalizations of C_2. Informally speaking, if clause C θ-subsumes clause D, then D can be converted to C by (1) dropping conditions and (2) turning constants to variables. A clause C is a least generalization of a set of clauses S, if

1. C generalizes each clause in S: $\forall E \in S : C \leq E$

2. C is the smallest clause satisfying condition 1: $(\exists D \forall E \in S, D \leq E) \Rightarrow D \leq C$

4.3.2 Generalized Subsumption

The definition of generality presented so far is local to the set S of clauses. Referring to implication instead of the weaker subsumption relationship would also consider generalization w.r.t. current knowledge. A generalization relative to a set of clauses P is defined as follows: A clause C generalizes a clause D relative to a set of clauses P if there exists a substitution θ such that $P \models \forall(C\theta \rightarrow D)$. Buntine defines *generalized subsumption* of definite Horn clauses as an extension of θ-subsumption with the restriction that the corresponding clause heads must be about the same concept [5]. Informally speaking, if a clause C generally subsumes clause D, then C can be converted to D by (1) turning variables to constants or other terms, (2) adding atoms to the body, and (3) partially evaluating the body by resolving some clause in P with an atom in the body. The third conversion process is additional to the conversion for θ-subsumption.

4.3.3 Generalization for KB Evolution

The condition of covering in the definition of generalized subsumption has the effect, that generalization depends on the actual representation of the clauses. Defining generalization in terms of implication instead of subset-relation would be more suitable. This would lead to a combination of techniques from inductive logic programming (ILP) and explanation-based learning (EBL) [23] by using deduction when deciding the generalization of clauses. Unfortunately, doing so, the test for generalization becomes undecidable. On the other hand, Buntine states that generalized subsumption is semidecidable, although it is guaranteed to terminate if P contains no recursion. Subsumption w.r.t. a DATALOG program, however, is decidable.

Also, for practical applications, least general generalization as defined above can still be too general. Thus, in order to overcome the problems raised by theory revision with background knowledge, namely its undecidability and its results being too general, we investigate how to incorporate more knowledge within the generalization, i.e. how to *control* generalization.

This extension leads to *partial least general generalization* (plgg) and allows us to only partially generalize two literals: We can say that we want to generalize two literals or terms, but can require some arguments to be fixed, i.e., that the two literals must have corresponding values at that specific argument position. Thus, partial least general generalization is a combination of unification and anti-unification. As unification can fail, plgg may fail too, but, of course, in the non-generalized argument positions only.

We can also restrict generalization by requiring some arguments to be of a particular type or sort. Thus, exact match is then replaced by sorted unification for the

non-generalized argument positions. But introducing sorts or types can be a first but only intermediate step: generalization within a sort lattice does already yield a more fine-grain clause ordering than simple θ-subsumption. However, extending the logic-based representation language by substituting or complementing constitutively given sorts by intensionally defined concepts and concept terms in the sense of terminological reasoning (cf. [3]) will be necessary for finding and expressing 'really least general' generalizations and thus being able to support KB evolution over a long period of time. Therefore, in [14] we have presented an alternative to θ-subsumption based on an extension of Horn rules incorporating terminological knowledge representation and reasoning (TL-subsumption).

4.3.4 Abduction

Generalization is applied to the clauses of the theory resulting in more general rules. For abduction purposes, in addition to the Horn-clause theory T consisting of facts F and rules R and a set of integrity constraints IC, we also postulate a set of distinguished ground literals A called *abducibles* and a goal G which drives the abduction process. By abduction we want to find a set of hypotheses $H \subset A$ such that we can derive the (positive) example $e \in E^+$ from $T \cup H$. In the context of theory revision $T \cup H$ gives the new theory T' which again must be consistent with IC, the set of integrity constraints, i.e., the following must hold: $T \cup H \vdash e$ and $T \cup H \cup IC$ is consistent.

In order to achieve abduction, deduction techniques can be employed in a top-down as well as in a bottom-up manner: with top-down reasoning one skips some subgoals instead of proving them if they are in the set of abducibles. If the goal only consists of abducibles, top-down reasoning stops. The set of remaining goals is the set of hypotheses explaining the top-level goal.

As mentioned before, generalization of Horn clauses can be done in different ways, e.g. by generalizing some terms (argument positions) or by dropping entire literals (removing conditions). Thus, the decision about which generalization operation should be applied is still a problem. Abduction can provide considerable help for making this decision.

4.4 Knowledge-Base Verification

It has already been pointed out that only those generalizations and abductive solutions are accepted which are consistent with the integrity constraints IC. Integrity constraints denote negative or disjunctive knowledge and are represented as denials, i.e. clauses with an empty head. Eshghi and Kowalski use this kind of integrity constraints for their abduction procedure [8]. They can also be represented as clauses with the special atom false as conclusion (cf. [21]). A naive approach for integrity checking would be to use a proof-finding approach and ask the query "?- false". This procedure would invoke all integrity constraints in backward direction even if they

are independent from the new fact. However, it would be more efficient to derive only those facts, that are consequences of this new assertion. In [8] it is proposed to do this kind of constraint checking by forward reasoning starting with the new fact. But forward reasoning alone turns out not to be sufficient.

Thus, in [21] a model-generation approach has been applied to this problem. However, when regarding integrity constraints checking as a consequence-finding problem [17] given an update of the knowledge base, only those rules are applied that are affected by the update operation. This builds on the assumption that the knowledge base satisfied its integrity constraints prior to the update. Derivation is restricted to exactly those facts that depend on an explicitly given set of initial facts, in our case the hypotheses found by abduction.

The extended SLDNF resolution of [28] uses the clauses corresponding to the updates as top-clauses for the search space and thus achieves the effect of simplification methods investigated by [19]. The approach combines forward and backward chaining depending on whether a positive or negative literal is resolved upon.

As an alternative to this tuple-oriented method, a rewriting approach has also been proposed [16]. It is an extension of the Generalized Magic Sets rewriting technique which allows query answering in deductive databases by bottom-up evaluation [1].

The most important property for the use in KB evolution is that this set-oriented approach is usable also for large sets of facts. This is compatible with the objective to develop techniques not only for toy examples but for complex real world problems within the challenge of Concurrent Engineering.

5 Conclusion

Only a few years ago, KB systems were only a research topic, but today they are already brought to practical applications. Now problems of maintenance(i.e., verification, validation, and exploration) and also sharing and reuse should be placed into the foreground as was previously the case with conventional software.

If an existing knowledge base is needed in another representation, a translation makes the reuse possible. The major differences are first the different paradigms and constructs of the knowledge base languages, second the mingling of universal, reusable knowledge with application-specific knowledge and third tailoring knowledge for a specific inference engine, which complicates its reuse by another inference engine.

As it comes to the practical use of knowledge bases, we have also shown that knowledge base evolution can be regarded as a theory revision process. Research in ILP and EBL provides us with a set of techniques that can be applied to incorporate new knowledge into the knowledge base (*knowledge base exploration*), e.g. by generalization and abduction. On the other side, techniques from deductive database research can be used for ensuring the integrity of the knowledge base, i.e., for solving the *knowledge base verification* task. For both tasks we have developed extensions

and modifications motivated by the idea of supporting the maintenance and evolution of knowledge bases over their entire life-time.

Thus, the main message of this paper is that Knowledge Sharing and Reuse as well as Knowledge Base Evolution will play an important role for the practical application of KB technologies to real-life applications. Moreover, as it is believed that Concurrent Engineering approaches will not succeed without incorporating KB technologies, research on the issues presented in this paper will become a challenge for the concurrent enterprise of the future.

References

[1] F. Bancilhon and R. Ramakrishnan. An amateur's introduction to recursive query processing strategies. In *Proc. ACM SIGMOD Conference*, pages 16–52. ACM, 1986.

[2] B. Boehm. A Spiral Model of Software Development and Enhancement. *IEEE Computer*, May 1988.

[3] A. Borgida, R. Brachman, D. McGuinness, and L. Resnick. CLASSIC: A structural data model for objects. In *International Conference on Managment on Data*. ACM SIGMOD, 1989.

[4] R. Brachman and H. Levesque. What Makes a Knowledge Base Knowledgeable? A View of Databases from the Knowledge Level. In L. Kerschberg, editor, *Expert Database Systems*, pages 69–78. Springer-Verlag, 1986.

[5] W. Buntine. Generalized subsumption and its applications on induction and redundancy. *Artificial Intelligence*, 36:149–176, 1988.

[6] B. Engel, C. Baffaut, J. R. Barrett, J. B. Rogers, and D. Jones. Knowledge Transformation. *Applied AI*, 4(1):67–80, 1990.

[7] L. D. Erman and V. R. Lesser. A Multi-Level Organization for Problem Solving Using Many Diverse, Cooperating Sources of Knowledge. In *Proc. IJCAI-75*, pages 483–490, 1975.

[8] K. Eshghi and R. Kowalski. Abduction compared with negation by failure. In *6th International Conference on Logic Programming (ICLP'89)*, 1989.

[9] T. Finin, D. McKay, and R. Fritzson. An overview of KQML: A knowledge query and manipulation language. Technical report, Computer Science Department, University of Maryland, 1992.

[10] M. R. Genesereth and R. Fikes. Knowledge Interchange Format Version 3.0 Reference Manual. Technical Report Logic-92-1, Computer Science Department, Stanford University, June 1992.

[11] A. Ginsberg, S. M. Weiss, and P. Politakis. Automatic knowledge base refinement for classification systems. *Artificial Intelligence*, 35:197–226, 1988.

[12] M. L. Ginsberg. Knowledge interchange format: The KIF of death. *AI Magazine*, 12(3):57–63, 1991.

[13] T. Gruber. The Role of Common Ontology in Achieving Sharable, Reusable Knowledge Bases. In J. Allen, R. Fikes, and E. Sandewall, editors, *Proc. Principles of Knowledge Representation and Reasoning (KR'91)*, pages 601–602. Morgan Kaufmann, April 1991.

[14] P. Hanschke and M. Meyer. An Alternative to θ-Subsumption Based on Terminological Reasoning. In C. Rouveirol, editor, *Workshop on Logical Approaches to Machine Learning, ECAI 92, Vienna*, August 1992.

[15] D. E. Hardaway and R. P. Will. A Review of Barriers to Expert System Diffusion. In *Proc. 1990 ACM SIGBDP on Trends and Directions in Expert Systems*, Orlando, Florida, 1990.

[16] K. Hinkelmann. A consequence-finding approach for feature recognition in CAPP. In *Seventh International Conference on Industrial & Engineering Applications of Artifical Intelligence & Expert Systems (IEA/AIE'94)*, June 1994.

[17] K. Inoue. Consequence-finding based on ordered linear resolution. In *Proc. IJCAI-91*, 1991.

[18] R. Knaus and C. Jay. Transporting Knowledge Bases: A Standard. *AI Expert*, 5(11):34–39, 1990.

[19] J. W. Lloyd, E. A. Sonenberg, and R. W. Topor. Integrity constraint checking in stratified databases. *Journal of Logic Programming*, 4:331–343, 1987.

[20] B. Lopez, P. Meseguer, and E. Plaza. Knowledge base systems validation: A state of the art. *AI Communications*, 3(2):58–72, June 1990.

[21] R. Manthey and F. Bry. SATCHMO: a theorem prover implemented in PROLOG. In *Proc. Conference on Automated Deduction (CADE)*, 1987.

[22] J. McCarthy and P. Hayes. Some philosophical problems from the standpoint of artificial intelligence. In *Machine Intelligence*, volume 4. North-Holland, 1969.

[23] R. J. Mooney and J. M. Zelle. Integrating ILP and EBL. *SIGART Bulletin*, 5(1):12–21, 1994. Special Section on Inductive Logic Programming.

[24] R. Neches, R. Fikes, T. Finin, T. Gruber, R. Patil, T. Senator, and W. Swartout. Enabling technology for knowledge sharing. *AI Magazine*, 12(3):36–56, 1991.

[25] D. Partridge. Artificial Intelligence and Software Engineering: A Survey of Possibilities. *Information and Software Technology*, 30(3), April 1988.

[26] G. D. Plotkin. A note on inductive generalization. In B. Meltzer and D. Michie, editors, *Machine Intelligence*, volume 5, pages 153–163. Elsevier North-Holland, New York, 1970.

[27] J. A. Robinson. A machine-oriented logic based on the resolution principle. *Journal of the Association for Computing Machinery*, 12:23–41, 1965.

[28] F. Sadri and R. Kowalski. A theorem-proving approach to database integrity. In J. Minker, editor, *Foundations of Deductive Databases and Logic Programming*, pages 313–362. Morgan Kaufmann Publishers, 1988.

Deutsches
Forschungszentrum
für Künstliche
Intelligenz GmbH

DFKI
-Bibliothek-
PF 2080
67608 Kaiserslautern
FRG

DFKI Publikationen

Die folgenden DFKI Veröffentlichungen sowie die aktuelle Liste von allen bisher erschienenen Publikationen können von der oben angegebenen Adresse oder per anonymem ftp von ftp.dfki.uni-kl.de (131.246.241.100) unter pub/Publications bezogen werden.

Die Berichte werden, wenn nicht anders gekennzeichnet, kostenlos abgegeben.

DFKI Publications

The following DFKI publications or the list of all published papers so far are obtainable from the above address or via anonymous ftp from ftp.dfki.uni-kl.de (131.246.241.100) under pub/Publications.

The reports are distributed free of charge except if otherwise indicated.

DFKI Research Reports

RR-93-14
Joachim Niehren, Andreas Podelski, Ralf Treinen:
Equational and Membership Constraints for Infinite Trees
33 pages

RR-93-15
Frank Berger, Thomas Fehrle, Kristof Klöckner, Volker Schölles, Markus A. Thies, Wolfgang Wahlster: PLUS - Plan-based User Support Final Project Report
33 pages

RR-93-16
Gert Smolka, Martin Henz, Jörg Würtz: Object-Oriented Concurrent Constraint Programming in Oz
17 pages

RR-93-17
Rolf Backofen:
Regular Path Expressions in Feature Logic
37 pages

RR-93-18
Klaus Schild: Terminological Cycles and the Propositional μ-Calculus
32 pages

RR-93-20
Franz Baader, Bernhard Hollunder:
Embedding Defaults into Terminological Knowledge Representation Formalisms
34 pages

RR-93-22
Manfred Meyer, Jörg Müller:
Weak Looking-Ahead and its Application in Computer-Aided Process Planning
17 pages

RR-93-23
Andreas Dengel, Ottmar Lutzy:
Comparative Study of Connectionist Simulators
20 pages

RR-93-24
Rainer Hoch, Andreas Dengel:
Document Highlighting —
Message Classification in Printed Business Letters
17 pages

RR-93-25
Klaus Fischer, Norbert Kuhn: A DAI Approach to Modeling the Transportation Domain
93 pages

RR-93-26
Jörg P. Müller, Markus Pischel: The Agent Architecture InteRRaP: Concept and Application
99 pages

RR-93-27
Hans-Ulrich Krieger:
Derivation Without Lexical Rules
33 pages

RR-93-28
Hans-Ulrich Krieger, John Nerbonne, Hannes Pirker: Feature-Based Allomorphy
8 pages

RR-93-29
Armin Laux: Representing Belief in Multi-Agent Worlds viaTerminological Logics
35 pages

RR-93-30
Stephen P. Spackman, Elizabeth A. Hinkelman:
Corporate Agents
14 pages

RR-94-10
Knut Hinkelmann, Helge Hintze:
Computing Cost Estimates for Proof Strategies
22 pages

RR-94-11
Knut Hinkelmann: A Consequence Finding Approach for Feature Recognition in CAPP
18 pages

RR-94-12
Hubert Comon, Ralf Treinen:
Ordering Constraints on Trees
34 pages

RR-94-13
Jana Koehler: Planning from Second Principles —A Logic-based Approach
49 pages

RR-94-14
Harold Boley, Ulrich Buhrmann, Christof Kremer:
Towards a Sharable Knowledge Base on Recyclable Plastics
14 pages

RR-94-15
Winfried H. Graf, Stefan Neurohr: Using Graphical Style and Visibility Constraints for a Meaningful Layout in Visual Programming Interfaces
20 pages

RR-94-16
Gert Smolka: A Foundation for Higher-order Concurrent Constraint Programming
26 pages

RR-94-17
Georg Struth:
Philosophical Logics—A Survey and a Bibliography
58 pages

RR-94-18
Rolf Backofen, Ralf Treinen:
How to Win a Game with Features
18 pages

RR-94-20
Christian Schulte, Gert Smolka, Jörg Würtz:
Encapsulated Search and Constraint Programming in Oz
21 pages

RR-94-31
Otto Kühn, Volker Becker, Georg Lohse, Philipp Neumann:
Integrated Knowledge Utilization and Evolution for the Conservation of Corporate Know-How
17 pages

RR-94-33
Franz Baader, Armin Laux:
Terminological Logics with Modal Operators
29 pages

DFKI Technical Memos

TM-92-04
Jürgen Müller, Jörg Müller, Markus Pischel, Ralf Scheidhauer:
On the Representation of Temporal Knowledge
61 pages

TM-92-05
Franz Schmalhofer, Christoph Globig, Jörg Thoben:
The refitting of plans by a human expert
10 pages

TM-92-06
Otto Kühn, Franz Schmalhofer: Hierarchical skeletal plan refinement: Task- and inference structures
14 pages

TM-92-08
Anne Kilger: Realization of Tree Adjoining Grammars with Unification
27 pages

TM-93-01
Otto Kühn, Andreas Birk: Reconstructive Integrated Explanation of Lathe Production Plans
20 pages

TM-93-02
Pierre Sablayrolles, Achim Schupeta:
Conlfict Resolving Negotiation for COoperative Schedule Management
21 pages

TM-93-03
Harold Boley, Ulrich Buhrmann, Christof Kremer:
Konzeption einer deklarativen Wissensbasis über recyclingrelevante Materialien
11 pages

TM-93-04
Hans-Günther Hein:
Propagation Techniques in WAM-based Architectures — The FIDO-III Approach
105 pages

TM-93-05
Michael Sintek: Indexing PROLOG Procedures into DAGs by Heuristic Classification
64 pages

TM-94-01
Rainer Bleisinger, Klaus-Peter Gores:
Text Skimming as a Part in Paper Document Understanding
14 pages

TM-94-02
Rainer Bleisinger, Berthold Kröll:
Representation of Non-Convex Time Intervals and Propagation of Non-Convex Relations
11 pages

RR-93-31
Elizabeth A. Hinkelman, Stephen P. Spackman: Abductive Speech Act Recognition, Corporate Agents and the COSMA System
34 pages

RR-93-32
David R. Traum, Elizabeth A. Hinkelman: Conversation Acts in Task-Oriented Spoken Dialogue
28 pages

RR-93-33
Bernhard Nebel, Jana Koehler: Plan Reuse versus Plan Generation: A Theoretical and Empirical Analysis
33 pages

RR-93-34
Wolfgang Wahlster: Verbmobil Translation of Face-To-Face Dialogs
10 pages

RR-93-35
Harold Boley, François Bry, Ulrich Geske (Eds.): Neuere Entwicklungen der deklarativen KI-Programmierung — *Proceedings*
150 Seiten
Note: This document is available only for a nominal charge of 25 DM (or 15 US-$).

RR-93-36
Michael M. Richter, Bernd Bachmann, Ansgar Bernardi, Christoph Klauck, Ralf Legleitner, Gabriele Schmidt: Von IDA bis IMCOD: Expertensysteme im CIM-Umfeld
13 Seiten

RR-93-38
Stephan Baumann: Document Recognition of Printed Scores and Transformation into MIDI
24 pages

RR-93-40
Francesco M. Donini, Maurizio Lenzerini, Daniele Nardi, Werner Nutt, Andrea Schaerf: Queries, Rules and Definitions as Epistemic Statements in Concept Languages
23 pages

RR-93-41
Winfried H. Graf: LAYLAB: A Constraint-Based Layout Manager for Multimedia Presentations
9 pages

RR-93-42
Hubert Comon, Ralf Treinen: The First-Order Theory of Lexicographic Path Orderings is Undecidable
9 pages

RR-93-43
M. Bauer, G. Paul: Logic-based Plan Recognition for Intelligent Help Systems
15 pages

RR-93-44
Martin Buchheit, Manfred A. Jeusfeld, Werner Nutt, Martin Staudt: Subsumption between Queries to Object-Oriented Databases
36 pages

RR-93-45
Rainer Hoch: On Virtual Partitioning of Large Dictionaries for Contextual Post-Processing to Improve Character Recognition
21 pages

RR-93-46
Philipp Hanschke: A Declarative Integration of Terminological, Constraint-based, Data-driven, and Goal-directed Reasoning
81 pages

RR-93-48
Franz Baader, Martin Buchheit, Bernhard Hollunder: Cardinality Restrictions on Concepts
20 pages

RR-94-01
Elisabeth André, Thomas Rist: Multimedia Presentations: The Support of Passive and Active Viewing
15 pages

RR-94-02
Elisabeth André, Thomas Rist: Von Textgeneratoren zu Intellimedia-Präsentationssystemen
22 Seiten

RR-94-03
Gert Smolka: A Calculus for Higher-Order Concurrent Constraint Programming with Deep Guards
34 pages

RR-94-05
Franz Schmalhofer, J.Stuart Aitken, Lyle E. Bourne jr.: Beyond the Knowledge Level: Descriptions of Rational Behavior for Sharing and Reuse
81 pages

RR-94-06
Dietmar Dengler: An Adaptive Deductive Planning System
17 pages

RR-94-07
Harold Boley: Finite Domains and Exclusions as First-Class Citizens
25 pages

RR-94-08
Otto Kühn, Björn Höfling: Conserving Corporate Knowledge for Crankshaft Design
17 pages

DFKI Documents

D-93-14
Manfred Meyer (Ed.): Constraint Processing – Proceedings of the International Workshop at CSAM'93, July 20-21, 1993
264 pages
Note: This document is available only for a nominal charge of 25 DM (or 15 US-$).

D-93-15
Robert Laux:
Untersuchung maschineller Lernverfahren und heuristischer Methoden im Hinblick auf deren Kombination zur Unterstützung eines Chart-Parsers
86 Seiten

D-93-16
Bernd Bachmann, Ansgar Bernardi, Christoph Klauck, Gabriele Schmidt: Design & KI
74 Seiten

D-93-20
Bernhard Herbig:
Eine homogene Implementierungsebene für einen hybriden Wissensrepräsentationsformalismus
97 Seiten

D-93-21
Dennis Drollinger:
Intelligentes Backtracking in Inferenzsystemen am Beispiel Terminologischer Logiken
53 Seiten

D-93-22
Andreas Abecker:
Implementierung graphischer Benutzungsoberflächen mit Tcl/Tk und Common Lisp
44 Seiten

D-93-24
Brigitte Krenn, Martin Volk:
DiTo-Datenbank: Datendokumentation zu Funktionsverbgefügen und Relativsätzen
66 Seiten

D-93-25
Hans-Jürgen Bürckert, Werner Nutt (Eds.):
Modeling Epistemic Propositions
118 pages
Note: This document is available only for a nominal charge of 25 DM (or 15 US-$).

D-93-26
Frank Peters: Unterstützung des Experten bei der Formalisierung von Textwissen
INFOCOM:
Eine interaktive Formalisierungskomponente
58 Seiten

D-93-27
Rolf Backofen, Hans-Ulrich Krieger, Stephen P. Spackman, Hans Uszkoreit (Eds.):
Report of theEAGLES Workshop on Implemented Formalisms at DFKI, Saarbrücken
110 pages

D-94-01
Josua Boon (Ed.):
DFKI-Publications: The First Four Years
1990 - 1993
93 pages

D-94-02
Markus Steffens: Wissenserhebung und Analyse zum Entwicklungsprozeß eines Druckbehälters aus Faserverbundstoff
90 pages

D-94-03
Franz Schmalhofer: Maschinelles Lernen:
Eine kognitionswissenschaftliche Betrachtung
54 pages

D-94-04
Franz Schmalhofer, Ludger van Elst:
Entwicklung von Expertensystemen:
Prototypen, Tiefenmodellierung und kooperative Wissensevolution
22 pages

D-94-06
Ulrich Buhrmann:
Erstellung einer deklarativen Wissensbasis über recyclingrelevante Materialien
117 pages

D-94-07
Claudia Wenzel, Rainer Hoch:
Eine Übersicht über Information Retrieval (IR) und NLP-Verfahren zur Klassifikation von Texten
25 Seiten

D-94-08
Harald Feibel: IGLOO 1.0 - Eine grafikunterstützte Beweisentwicklungsumgebung
58 Seiten

D-94-09
DFKI Wissenschaftlich-Technischer Jahresbericht 1993
145 Seiten

D-94-10
F. Baader, M. Lenzerini, W. Nutt, P. F. Patel-Schneider (Eds.): Working Notes of the 1994 International Workshop on Description Logics
118 pages
Note: This document is available only for a nominal charge of 25 DM (or 15 US-$).

D-94-11
F. Baader, M. Buchheit, M. A. Jeusfeld, W. Nutt (Eds.):
Working Notes of the KI'94 Workshop:
KRDB'94 - Reasoning about Structured Objects: Knowledge Representation Meets Databases
65 Seiten

D-94-12
Arthur Sehn, Serge Autexier (Hrsg.): Proceedings des Studentenprogramms der 18. Deutschen Jahrestagung für Künstliche Intelligenz KI-94
69 Seiten

Issues in Concurrent Knowledge Engineering: Knowledge Sharing and KB Evolution

Manfred Meyer

RR-94-36
Research Report