Künstl Intell (2010) 24: 169–173
DOI 10.1007/s13218-010-0019-1

# Integration of Similarity-based and Deductive Reasoning for Knowledge Management

**Babak Mougouie**

Published online: 13 May 2010

**Abstract** Many disciplines in computer science combine similarity-based and logic-based reasoning. The problem is that the disciplines combine these independently of each other. For example in *Case-Based Reasoning* (CBR) (Aamodt and Plaza, AI Commun. 7(1):39–59, 1994; Bergmann et al., Künstl. Intell. 23(1):5–11, 2009; Bergmann, Experience Management: Foundation, Development, Methodology and Internet-based Applications, LNAI, vol. 2432, Springer, Berlin, 2002), the combination is applied in a sequential manner and not systematically as follows: a set of solutions is retrieved from a case-base using a similarity measure and then deductive reasoning is applied to adapt the retrieved solutions to a query. The aim of this dissertation (Mougouie, Ph.D. thesis, Trier University, Germany, 2009) is to integrate similarity-based and deductive reasoning in a unified manner within the context of Knowledge Management (KM).

**Keywords** Case-based reasoning · Knowledge management · Deductive reasoning

## 1 Introduction

Logic-based approaches for knowledge representation and reasoning have a long tradition in *Artificial Intelligence* (AI) and KM. However, recently the perspective of KM in real world problem solving scenarios is drastically extended in a sense that the output of knowledge-based systems is not simply considered correct or incorrect [8]. This is mainly due to the facts that first, ideal solutions of problems might not exist at all and second, the knowledge is often incomplete, imprecise and even contradicting since distributed knowledge gets expanded and changed very rapidly. Therefore the basic view on KM has already been shifted from pure logic-oriented to similarity-based approach. It has been recognized that knowledge-bases formalizing real world scenarios should be equipped with utility functions, fuzzy sets, probabilities or similarity measures to introduce *approximation* as a new paradigm.

Many disciplines combine similarity-based and logic-based reasoning. However these reasoning methods are combined independently of each other, one after the other and not systematically. Particularly in CBR [1–3], which is of the interest of this dissertation, reasoning with cases is usually done in a similarity-based manner. However it has always been recognized that additional *general knowledge* should be added to cases e.g. for the purpose of adapting cases. This general knowledge is often represented in rules, constraints or ontology definitions and is usually applied in a deductive reasoning process after similarity-based retrieval.

The main purpose of the thesis [7] is to present a way of combining similarity-based and deductive reasoning in a unified manner. Hence, we introduce similarity-based reasoning as search for the most similar elements (to a query w.r.t. some similarity measure) in the deductive closure of a domain theory. We elaborate the approach, finding *Most Similar Deductive Consequences* (MSDC), and introduce several search algorithms [4].

Our basic idea is to view general knowledge and cases together as a logical theory $\Sigma$ of a domain. In $\Sigma$, cases are usually encoded as facts. General knowledge is added to $\Sigma$

The dissertation was done under the supervision of Prof. Dr. Ralph Bergmann and Prof. Dr. Michael M. Richter at the Dept. of Business Information Systems II, Trier University, Germany.

B. Mougouie (✉)
Computer Science Dept., University of Geneva, Geneva, Switzerland
e-mail: babak.mougouie@unige.ch

in form of general logical sentences and possibly additional facts. $\Sigma$ must be constructed such that the deductive closure represents the knowledge we know with certainty to be true in that domain. Similarity-based reasoning is introduced as follows: for a given query $q$ and a similarity measure *sim*, we search for the $k$ most similar elements in the deductive closure and thus find the most similar deductive consequences.

*Main Scientific Contributions* (1) We provide a generalized view on several CBR approaches and integrate CBR similarity-based and logic-based reasoning methods in a unified manner. (2) We present search algorithms that integrate similarity-based search with well-known AI search processes performed during deduction. Our search algorithms show a different and unified way of performing the case-based reasoning tasks from the earlier CBR approaches. The developed algorithms differ in the heuristics used for pruning the huge overall search space. (3) Due to similarities between *Ontology-Based Knowledge Management* (*OBKM*) and *Structural Case-Base Reasoning* (*SCBR*) (see [5]), we show that MSDC can be used as a general framework for integrating OBKM and SCBR. (4) We combine *Optimization* methods with the heuristic search algorithms, extend approximation techniques of symbolic domains and integrate them with deductive reasoning methods.

The paper is organized as follows. In Sect. 2, we define MSDC and mention the search algorithms with experimental comparison. Section 3 is dedicated to the application of MSDC in implementing and re-interpreting CBR approaches and integrating SCBR and OBKM. Section 4 provides optimization algorithms to solve or approximate MSDC and their experimental comparison.

## 2 Finding MSDC

First, we need as basis a logical system with an inference calculus in which all knowledge, including cases, are represented. One can use predicate logic, Horn-logic, frame logic, or description logic. The latter appears to be very promising due to its use in ontology languages. However, for the scope of the dissertation, we restrict ourselves to Horn logic without recursive deductions.

Based on a domain theory $\Sigma$ consisting of formulas of the logic (a set of facts and rules in Horn logic) in which all knowledge of the domain including cases are represented, we look for the most similar case $q' = p(t'_1, \dots, t'_n)$ that can be deduced from $\Sigma$ to a query $q = p(t_1, \dots, t_n)$ ($t_1, \dots, t_n$, $t'_1, \dots, t'_n$ can be constants or variables) given a similarity measure $sim_p : AF_p \times AF_p \mapsto [0, 1]$ where $AF_p$ is the set of all atomic formulas starting with the $n$-ary predicate symbol $p$.

We further formalize the similarity measure following the local-global principle: $sim_p(q, q') = \Omega(sim_1(t_1, t'_1), \dots, sim_n(t_n, t'_n))$ such that $\Omega$ is an aggregate function that is monotonous in every argument and $0 \le sim_i(t_i, t'_i) \le 1$ for $i = 1, \dots, n$ are local similarities.

**Definition** The *most similar deductive consequence* is defined as follows:

$$MSDC(q) = \arg\max_{q' \in closure_p(\Sigma)} sim_p(q, q')$$

s.t. $q = p(t_1, \dots, t_n)$ and $closure_p(\Sigma) = \{p(t'_1, \dots, t'_n) \mid \Sigma \vdash p(t'_1, \dots, t'_n)\}$ is the deductive closure of $\Sigma$ restricted to atomic formulas starting with the $n$-ary predicate symbol $p$.

This is extended to $k$-MSDC which delivers the $k$-most similar deductive consequences: $k\text{-}MSDC(q) = \{q_1, \dots, q_k\} \subseteq closure_p(\Sigma)$ such that $sim_p(q, q') \le \min\{sim_p(q, q_i) \mid i = 1, \dots, k\}$ $\forall q' \in closure_p(\Sigma) - \{q_1, \dots, q_k\}$.

*Example* Consider the following domain theory denoted in traditional notation for Prolog:

```
q(X,Y) :- c(X,Y).
q(X,Y) :- c(X1,Y1), a(X,Y,X1,Y1).
c(2,5).
c(25,39).
a(X,Y,X1,Y1) :- D is X-X1, D>0, D<3,
Y is Y1+X-X1.
```

Consider the facts with the predicate `c` to be cases of a case base, with the first argument representing the problem and the second argument representing the solution. The first clause represents the fact that a query can be answered directly by a case (without adaptation) while the second clause describes that a case `c(X1,Y1)` is selected and then an adaptation operator is applied such as in transformational adaptation. The last clause represents the adaptation operator itself: if the difference in the problem is less than 3 then the solution is linearly adapted by the formula mentioned. Of course, a traditional Prolog interpreter, is able to answer queries of the kind `?- q(3,Y)` and in this case would return the answer `Y=6`. However, it would not be able to find an answer to the query `?- q(5,Y)`. An equivalent CBR system in this case would use a similarity measure to find the most similar case, which could be `c(2,5)` and apply the adaptation operator to this case and return the result `c(4,7)`. This is exactly what MSDC also does. For this purpose, assume that the following similarity measure $sim_q((q(X, Y), q(X1, Y1)) = 1 - (|X - X1|/100)$ is given. As usual in CBR, this similarity measure assesses the similarity of the problem attribute. With this similarity, $MSDC(q(5, Y)) = q(4, 7)$.

*Search Algorithms to Find MSDC* We developed several standard AI search methods and some of their varieties and combinations to compute or approximate $k$-MSDC or to deliver its approximation. They integrate similarity-based search with the search process performed during deduction to compute the closure. They differ in the heuristics used for pruning the huge search space. Our search space to find the elements of $closure_p(\Sigma)$ is a state space in which each state differs from its successor in one resolution step (we restrict ourselves to domain theories with finite state spaces).

The first obvious search methods, which deliver the exact solutions of $k$-MSDC, are complete search methods in a depth first (dfs) or breadth first (bfs) order. Thereby, all elements of the closure are constructed, evaluated w.r.t. the similarity measure and ordered according to the similarity value. These exhaustive search methods lead to high computational cost.

To improve the search, we defined heuristics knowing that the similarity measure can be used to decide whether a branch in the search tree can still contribute to the computation of $k$-MSDC. Therefore, we can apply the similarity measure to a node in the search tree to compute an upper bound of the similarity that can be achieved by the solutions in the sub-tree.

dfs_MAS is a depth-first search that reduces the size of the explored search space by introducing a minimum acceptable similarity value $\mu$ and prunes the solutions whose similarities to $q$ cannot anymore exceed $\mu$. This algorithm cuts the search space at the cost of not assuring to finding the optimal solution of the $k$-MSDC problem. It only finds up to $k$ solutions with a similarity higher than $\mu$. To use this approach it is of course required to know in advance a good value for $\mu$. Therefore, we developed a method for efficiently computing an approximation for $\mu$ by use of optimization methods (refer to Sect. 4).

The second algorithm implements a beam search maintaining some states with the highest similarity.

The third algorithm, look-ahead pruning (lap), improves the pruning approaches by introducing a branch-and-bound technique. The idea is to find for each open state a lower and an upper bound for its similarity to the query. Here we use this information to prune states whose upper bound similarities are lower than any lower bound similarity. This technique is combined with beam search, leading to the algorithm called lap_beam.

*Experimental Comparison of Search Algorithms* To evaluate the performance of the search algorithms with respect to their computation time and similarity error caused by the pruning heuristics, we implemented the algorithms in SWI-Prolog.[1] As test domains we employed: a case-based configuration scenario and a car buying scenario in electronic commerce.

For the domain theories of both scenarios, the algorithms showed more or less the same behavior in terms of execution time and accuracy of the results although some small differences can not be overseen. dfs was the slowest algorithm in the majority of experiments and beam and dfs_MAS were the fastest, but at the cost of producing a much higher similarity error. lap_beam and lap produced a fairly lower error in average, but were significantly slower than beam. Both lap_beam and lap were faster than dfs in the majority of experiments. However, in some experiments, they got very time-consuming, even worse than dfs.

## 3 Application

MSDC can be used to implement and re-interpret existing CBR approaches and go even beyond CBR.

*Completion Rules* The formulation of completion rules is a very straightforward use of general knowledge [2], which is widely used in CBR applications. Rules are applied for inferring additional properties not explicitly represented in the cases or the query, but which are necessary for the similarity assessment. Then each completion rule is transferred into a Horn clause of the following schema:

$$c(X_1,\dots,X_{k-1},XNew_k,X_{k+1},\dots,X_n) :- c(X_1,\dots,X_n),$$
$$pre(X_1,\dots,X_n),\ XNew_k \text{ is } action(X_1,\dots,X_n).$$

The head of the clause represents the case that results by the completion; the variable $XNew_k$ gets a new value. $pre$ represents the rule's precondition defined over the values $X_1,\dots,X_n$ of the attribute of the selected case. If the precondition is fulfilled, the expression action computes the new value. In a regular deduction process, several completion rules can be chained.

*Transformational Adaptation* As shown in the example before, transformational adaptation can be implemented in the MSDC approach by encoding adaptation operators as Horn rules. The general form of these clauses is as follows:

$$c(X_1,\dots,X_n) :- c(Y_1,\dots,Y_n), pre(Y_1,\dots,Y_n),$$
$$X_1 \text{ is } act_1(Y_1,\dots,Y_n),\dots,X_n \text{ is } act_n(Y_1,\dots,Y_n).$$

Again, the head of the clause represents the result of the adaptation and $pre$ is the operator's precondition based upon the selected case. Then the value of each attribute $X_i$ is adapted by a separate $act_i$.

*Generalized Cases* In CBR, a generalized case (when a case is not a point in the problem-solution space but a subspace of it) can be represented by a set of constraints that specify its subspace. Generalized cases are particularly useful to reduce the size of a case base, which can ease case authoring

[1] www.swi-prolog.org.

and retrieval. Such generalized cases can be easily encoded within the MSDC approach by a Horn rule:

$$c(X_1,\dots,X_n) :- cons_1(X_1,\dots,X_n,X_{n+1},\dots,X_{n+m}),\dots,$$
$$cons_k(X_1,\dots,X_n,X_{n+1},\dots,X_{n+m}).$$

In this schema $cons_i$ are the constraints which relate the variables in the case and possible local variables to one another. Of course the used constraints must be specified in Horn logic by additional clauses.

*Configuration* Recommender systems for configurable products can also be implemented using the MSDC approach. Here, a product should be configured from several components, while various constraints must be maintained. The query represents the desired properties of the overall product; the result will be some configurations as similar as possible to the query. This problem is encoded as a domain theory as follows: Individual components are represented as facts in the domain theory. Several components are then combined to form a product or sub-product while the combinations of components with related constraints are encoded in Horn rules.

*Integration of SCBR and OBKM with MSDC* MSDC provides a general framework for integrating SCBR and OBKM. An ontology can be represented as an SCBR system and therefore as a domain theory in MSDC. Thus MSDC can be interpreted as an OBKM system equipped with similarity. The important point is that MSDC possesses many of the characteristics of both SCBR and OBKM which are missing in either of them. For example some important issues are the notions of mapping, assignment and semantic unification which can also be provided by MSDC (see [7]).

## 4 Optimization Methods

We proposed a formulation $\mathcal{OP}$-MSDC($q$) of MSDC in [6], which is a mixed integer optimization problem. Although such problems are exponentially solvable in general, our experimental results show that $\mathcal{OP}$-MSDC($q$) is solved surprisingly faster than our heuristic algorithms. Based on this observation, we provide several algorithms to find or approximate $k$-MSDC.

In order to apply optimization methods for real domain theories with variables bound to symbols, it was necessary to extend approximation techniques and transform symbols of a domain theory into integer numbers by replacing each symbol with a unique integer number. Furthermore, similarity measures with symbols were transformed into numerical similarity functions. As a consequence, $\mathcal{OP}$-MSDC($q$) was an approximation of MSDC. However, our experimental results show that the gaps between the results of $\mathcal{OP}$-MSDC($q$) and MSDC are very small.

For solving $\mathcal{OP}$-MSDC($q$), we apply some preprocessing techniques such as *domain reduction* and *constraint generation* (refer to [7] for discussion on this issue and our further *pruning* and *relaxation* techniques).

We use the package *lp_solve*[2] to solve optimization problems. Assuming that lp_solve($\mathcal{OP}$) is a program that provides an optimal solution of an optimization problem $\mathcal{OP}$, we developed several algorithms to find or approximate $k$-MSDC among them the following two.

*$\mathcal{OP}$-k-MSDC* The idea of this algorithm is straightforward. Given a query $q$, we first solve lp_solve($\mathcal{OP}$-MSDC($q$)) and retrieve the optimal solution of $\mathcal{OP}$-MSDC($q$). Then we add a pruning constraint to $\mathcal{OP}$-MSDC($q$) to prune the retrieved optimal solution and solve the new optimization problem. Keeping a list $k$-MSDC of the retrieved optimal solutions, we continue the same procedure until no further solution is found or $k$-MSDC contains $k$ solutions.

*dfs_MAS_$\mathcal{OP}$* This algorithm is the same as dfs_MAS in which $\mu$ is set to $sim_p(q,q') - \epsilon$ such that $q'$ is found by $\mathcal{OP}$-$k$-MSDC with $k = 1$ and $\epsilon$ is a small deviation e.g. 2%, 4%,.... An upper bound for $\epsilon$ is retrieved using statistics, e.g. applying several experiments and make $\epsilon$ bigger and bigger until $k$ solutions are found.

*Comparison of Optimization Algorithms* $\mathcal{OP}$-$k$-MSDC is the fastest algorithm among all implemented algorithms (the ones in this section and those in Sect. 2) with very small errors. dfs_MAS_$\mathcal{OP}$ is less precise and a bit slower than $\mathcal{OP}$-$k$-MSDC, but generates also small errors. Surprisingly, its errors are a lot less than those of dfs_MAS. The reason for this is the particular choice of the parameter $\mu$ which is very close to the real similarity for the best solution. We can conclude, this approach is useful in general if a good and fast approximation algorithm to estimate $\mu$ is at hand.

The only disadvantage of $\mathcal{OP}$-$k$-MSDC is that it should be run until termination whereas the other algorithms can be interrupted whenever a stopping criterion is satisfied.

## References

1. Aamodt A, Plaza E (1994) Case-based reasoning: foundational issues, methodological variations, and system approaches. AI Commun 7(1):39–59
2. Bergmann R (2002) Experience management: foundation, development, methodology and internet-based applications. LNAI, vol 2432. Springer, Berlin
3. Bergmann R, Althoff K-D, Minor M, Reichle M, Bach K (2009) Case-based reasoning—introduction and recent developments. Künstl Intell 23(1):5–11

[2]lp_solve: http://tech.groups.yahoo.com/group/lp_solve/.

4. Bergmann R, Mougouie B (2006) Finding similar deductive consequences—a new search-based framework for unified reasoning from cases and general knowledge. In: Proceedings of the 8th European conference, ECCBR 2006
5. Bergmann R, Schaaf M (2003) Structural case-based reasoning and ontology-based knowledge management: a perfect match? J Univers Comput Sci 9(7):608–626
6. Mougouie B (2008) Optimization algorithms to find most similar deductive consequences (MSDC). In: The 9th European conference on case-based reasoning, ECCBR 2008, Trier, Germany, September 2008
7. Mougouie B (2009) Integration of similarity-based and deductive reasoning for knowledge management. PhD thesis, ISBN 978-3-86853-091-9, Trier University, Germany
8. Richter MM (2004) Logic and approximation in knowledge based systems. In: Lenski W (ed) Logic versus approximation. LNCS, vol 3075. Springer, Berlin, pp 33–42