An Approach for Acquiring Domain Knowledge During the Case Based Reasoning: the System FrakaS-DL

Hioual Ouided[1], Laskri Mohamed Tayeb[2]
[1]Department of informatics
University Abbas Laghrour of Khenchla
Algeria
[2]LRI, Department of informatics
University of Annaba
Algeria
hioual_ouided@yahoo.fr

ABSTRACT: *As a young discipline at the junction of computer science, artificial intelligence and cognitive sciences, knowledge engineering aims at modelling knowledge of a specific domain to operationalise them in a computer system. To this end, it offers theoretical tools, models and empirical methodologies to support Knowledge sharing between the user and the system. The work developed here is related to knowledge engineering of a particular type of system: case-based reasoning systems (CBR) The case-based reasoning (CBR) is to solve a problem by remembering and adapting past cases already resolved. The CBR systems handle various kinds of knowledge: the case, the domain knowledge, knowledge of similarity and adaptation. The cases are collected in a gradual manner when using the system and the case base is enriched incrementally, while other types of knowledge are typically acquired when the system design. In particular the domain knowledge.*

This paper presents an approach for acquiring domain knowledge-based adaptation system failures. This approach has been implemented in a prototype, called FRAKAS, using the description logic.

Keywords: Conservative Adaptation, Description Logic, Reasoning from Case, Based Reasoning Domain Knowledge, Theory of Revision

Received: 27 February 2012, Revised 3 April 2012, Accepted 9 April 2012

1. Introduction

A system of case-based reasoning (CBR) is a system that relies on knowledge. Among these knowledge, there are cases sources, domain knowledge, and knowledge of similarity and adaptation.

This paper presents an approach to acquire domain knowledge of a CBR system. Specifically, this acquisition is done in sessions of case-based reasoning: when the target problem is solved by adapting the retrieved case, it is presented to the user who can demonstrate the fact that the solution is unsatisfactory and why it is, and it is the failure situations of interest here, the solution may be inconsistent with the knowledge of the expert or may be only partial (missing the user information in order to exploit this solution completely).

This new knowledge is used to repair and adaptations failed to prevent similar failures in future arguments. Therefore, this work

concerns the adaptation stage of CBR in [1], [2].

After having, in Section 2, presented a brief review of the case-based reasoning, we introduce and discuss the issues and objectives of our research in Section 3. In Section 4, these objectives were implemented in a prototype called FRAKAS. Finally, Section 5 concludes this paper.

2. Case Based Reasoning

In Case based reasoning, cases are generally represented by couples problem-solution, if a source will be denoted by srce -case = (srce; Sol (srce)) is the part where srce problem and Sol (srce) his party solution. The target case, where only the target problem is known if target = (target?), The adaptation consists in determining a solution Sol (tgt) from target srce-case for target-case =completed (target; Sol (tgt)).

This representation is based on the assumption that the representation of a source case in some problem and some solution can be uniquely independent of the target case.

Case based reasoning (CBR) is a paradigm of problem-solving which uses past experiences to solve new problems. Reuse of experience constitutes the main specificity and strength of CBR: reasoning bases itself on remembering and reusing past situations rather than on the exclusive use of formal knowledge of the domain. The exploitation of past situations is often profitable, particularly when knowledge of the domain is incomplete: experience still offers a "*basis*" for the solution. Of course CBR does not always give the ideal solution to the problem but, if it has the experience of this problem, it always offers a solution.

This solution, although imperfect, is nearly always satisfactory in real cases. The basic CBR principle, "*to solve a target problem, retrieve a source case and adapt it* ", can be summarized as in figure 1.

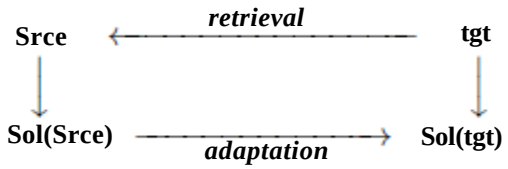

Figure 1. CBR classical paradigm

2.1 Cycle of reasoning in CBR

A cycle of case-based reasoning is to take as input a new problem target (pbcible) and to be able to infer a solution (solcible) from the other cases the base, known source case, and noted "*cassource*". The first to give an explicit description of characteristics of the CBR and a methodology for developing a CBR system are Agnar Aamodt and Enric Plaza [1]. For authors, the CBR is performed in a cycle four steps organized around a case base and knowledge: remembering ("*retrieve*"), adaptation ("*reuse*"), revision ("*revised* ") and memory ("*retain*"). the cycle Agnar Aamodt and proposed by Enric Plaza was a basis for reflection on the CBR for ensuing years. Variations of this cycle have been proposed, or adding some clarifying steps. We can identify such a preliminary stage that is developing, thus constituting the cycle shown in Figure 2.

We note SDK domain knowledge: SDK expresses knowledge believed to be correct but not necessarily complete. In particular, SDK gives necessary conditions for a case to be lawful. [3], [4].

3. Problematic and objectives

Between the domain knowledge (SDK) available to the CBR system, and knowledge of the expert, there is a big difference from [5], since the domain knowledge are available but are not sufficient, it is therefore necessary to acquire new ones. This problem is impossible to solve completely for most applications, but we can still learn new domain knowledge thanks to the expert.

The general principle here is to do reasoning and learning takes place when using the system and aims to acquire domain knowledge. When the solution is evaluated it may be unable-to solve the problem: it is then a failure of reasoning that is the subject of a process of learning from failure. The expert comes to identify parts of solution inconsistent.

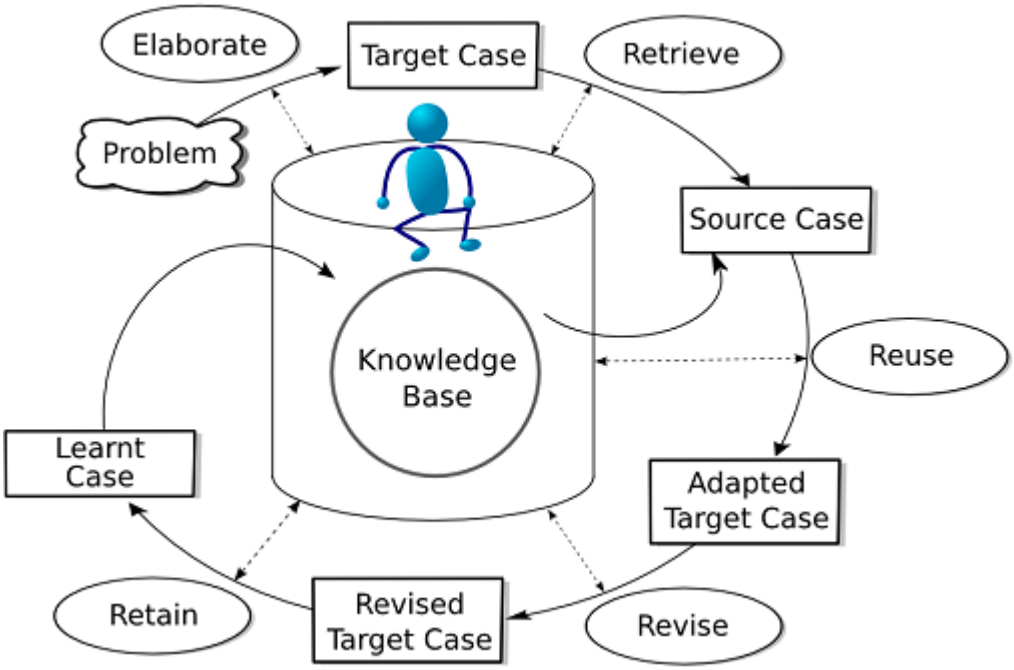

Figure 2. A CBR cycle (Cordier et al. - 2006.)

Two types of failures were identified in this paper and lead to acquisition of knowledge:

- Failed due to an inconsistency of the solution with the knowledge of the expert. The expert said that, given what he knows of domain knowledge, the affirmation of the solution of the target problem is inconsistent. This may mean that the solution itself is incoherent.

- Failure to have a solution that is only partial. If the solution proposed by the adaptation target is partial, and therefore not fully satisfactory, the interaction with the expert can help clarify it.

In this paper we used system FRAKAS, so enhancing the use of this system in a real situation and to reduce complexity and facilitate the work of the expert, it will be necessary to install a new version of FRAKAS, using the description logic. Thus, we proposed an algorithm for knowledge base revision in description logics. We chose the formalism of Description logic because of its ability to dual representation and reasoning about knowledge.

4. FRAKAS (Failure Analysis for domain Knowledge AcquiSition)

FRAKAS is an illustration of the FIKA principles. It defines strategies to interactively learn domain knowledge on-line, by exploiting reasoning failures and their correction. The learning process occurs during a CBR session. The target problem is automatically solved by adaptation of a retrieved case and then, the proposition is presented to the "*user*" who, depending on his expertise level, is supposed to highlight the part, in the proposition, that is not satisfactory.

FRAKAS offers an interactive mechanism that aims at incorporating new pieces of domain knowledge. The new knowledge is then added to the system to prevent similar failures occurring in future reasonings and, especially, to perform a new adaptation with a more complete knowledge. As a result, the system progressively learns new pieces of knowledge and becomes more and more effective.[6]

FRAKAS uses a technique of guided retrieval adaptability. When a source case is remembered, it uses conservative adaptation to infer Sol (tgt) from the target problem and source case. The conservative adaptation is to modify the source case in a minimal way to be both consistent with the knowledge base and the target problem. The result of the adaptation is presented to the expert who can then detect an inconsistency of the proposed solution with personal knowledge.

Algorithm of FRAKAS.

```
Input: tgt, SDK, CB
 (srce; Sol(srce))  Retr ieval(SDK; tgt; CB)
Sol(tgt)  Adaptation(SDK; (srce; Sol(srce)); tgt)
{Taking into account type 1 failures}
while Sol(tgt) is inconsistent do
    The expert points out Inc {Inc: the inconsistency}
   The expert gives a textual explanation of the failure (stored
   for later use)
   'Inc is false' is integrated to SDK
   Sol(tgt)  Adaptation(SDK; (srce; Sol(srce)); tgt)
end while
{Taking into account type 2 failures}
 if Sol(tgt) is fully specified then
   Exit
end if
 while There is an inconsistent interpretation of Sol(tgt) do
   {Justification of this loop:}
   {The modification of the knowledge base can generate new
    inconsistent adaptations}
    for all inconsistent interpretation do
        The expert points out Inc
       The expert gives a textual explanation of the failure
        (stored for later use)
       'Inc is false' is integrated to SDK
    end for
    Sol(tgt)  Adaptation(SDK; (srce; Sol(srce)); tgt)
end while
```

4.1 Formalism used: the ALC Description Logic

Description Logics (DL) were first developed to provide a formal meaning, declarative semantic networks and frames, and to show how such structured representations can be provided with effective tools of reasoning. They form a family of knowledge representation formalisms that can be used to represent and reason about the knowledge of an application domain in a structured and formally well understood. They are increasingly important in knowledge representation. [7]

4.1.1 Syntax

The elements of the representation language ALC are the concepts, roles, bodies and forms. Intuitively, a concept represents a subset of the domain of interpretation. A concept is either an atomic concept (ie,d. A concept name), or a conceptual expression of one of the following form:

T, $\perp$, C$\sqcap$D, $\neg$ C, C$\sqcup$ D,$\forall$ r .C, $\exists$ r .C where C and D are concepts (atomic or not) and r is a role. In a concept can be associated with a first-order formula with one free variable x.

For BC 'Ψ' in LAC is a finite set of formulas ALC. The terminological part (TBox or terminology to box) of Ψ is the set of its

formulas terminology. The assertionnelle party (or for ABox assertional box) of Ψ is the set of its formulas assertionnelles.

An interpretation is a pair $I = (\Delta_I , \cdot^I)$ where Δ_I is a nonempty set (the domain of interpretation) and where $\cdot^I$ associated with a concept C a subset C^I of Δ_I , a role r in a relationship binary r^I on Δ_I (for x, y $\in \Delta_I$, x is related to y is denoted by r^I , (x, y) $\in r^I$) and, to an instance has an element a^I of r^I. [8]

Given an interpretation I, the different types of conceptual expressions are interpreted by:

$T^I = \Delta_I$ $(C \sqcap D)^I = C^I \cap D^I$ $(\neg C)^I = \Delta^I \backslash C^I$

$\bot^I = \Phi \sqcup (C \quad D)^I = C^I \cup D^I$

$(\exists r.C)^I = \{ x \in \Delta_I$ / there exists y such that (x, y) $\in r^I$ and y $\in C^I \}$

$(\forall r.C)^I = \{ x \in \Delta_I$ / for all y, if (x, y) $\in r^I$ then y $\in C^I \}$

a) Inferences:

DL system does'nt store only terminologies and assertions, but also offers the services of inference. Mainly dependent on the reasoning in a DL is to discover implicit knowledge from explicit knowledge by inference. The services are also inference made on all the TBox and as well as the ABox.

b) Basic inferences about the TBox:

Given a TBox T, C and D two concepts, then the typical tasks of reasoning on T consist of:

- Checking satisfiability of a concept: A concept C is satisfiable (or consistent) with respect to a TBox T if there exists a model I of the TBox T such that $C^I \neq \emptyset$; (I is a model C), we write I | = C.

- Checking subsumption relation between two concepts: C subsumes D (D is considered the concept more general than C), written C $\sqsubseteq$ D, with respect to TBox T iff $C^I \sqsubseteq$ DI for all models I of the TBox T In this case, we write C $\sqsubseteq_T$ D or T | = C$\sqsubseteq$D. For example, PARENT$\sqsubseteq$PERSON. The subsumption relation presents the service more complex classification: given a concept C and a TBox T, for all concepts D of T determine whether D subsumes C or D is subsumed by C. Intuitively, this determination research relationships implicit in the terminology. In particular, the classification, a basic task in building up a new terminology that expresses the concept in the appropriate place in the taxonomic hierarchy of concepts, can be accomplished by checking the subsumption relation between each concept defined in the hierarchy and expression of the new concept.

- Verification of equivalence between two concepts: Two concepts C and D are equivalent, written C $\equiv$ D, with respect to T iff $C^I \equiv D^I$ for all models I of TBox T. In this case, we write C $\equiv_T$ D or T | = C $\equiv$ D

- Verification of disjunction between two concepts: Two concepts C and D are disjoint, written C $\neq$ D, compared to a TBox T iff $C^I \cap D^I = \emptyset$, for all models I of TBox T.

In fact, checking the satisfiability of concept is a main inference. other inferences for concepts can be reduced to (in) satisfiability and vice versa.

c) Basic inferences about the ABox:

ABox reasoning about a focus on testing the correctness of a domain model. Must perform two tasks:

- Checking instance: whether an individual has an ABox A is an instance of a given concept description C (a $\in C^I$), written A | = C (a).

- The consistency check: An ABox A is consistent with respect to a TBox T, if there is an interpretation that is a model of both A and T.

Satisfiability of an ABox is to test whether, given a TBox T, ABox A has a model. Important inferences can be reduced to this inference, p. ex. T | = C$\sqsubseteq$D iff A = {(C$\sqcap\neg$ D)(a)} is not satisfiable modulo T, where a is a new instance (can't be found in (C$\sqcap\neg$ D), or in T). [9]

4.1 Conservative Adaptation

Adaptation is a step of some case-based reasoning (CBR) systems that consists in modifying a source case in order to suit a new situation, the target case. An approach to adaptation consists in using a belief revision operator, i.e., an operator that modifies minimally a set of beliefs in order to be consistent with some actual knowledge .

The idea is to consider the belief "*The source case solves the target case*" and then to revise it with the constraints given by the target case and the domain knowledge.

The adaptation performed by FRAKAS is conservative adaptation (CA) (see [10] for details). In this adaptation, the approach is to make changes "*minimum*" of the source case to be consistent with both the target problem and the domain knowledge. It is formalized through the notion of revision operator [11], [12], [13]: a revision operator 'o' combines two knowledge bases Ψ and ì knowledge base Ψ o μ which, intuitively , is obtained by minimal change on Ψ to be consistent with μ.[16]

In a description logic, revision of a knowledge base is through a revision algorithm on ABox (modulo a TBox).

4.2 Revision algorithm on ABox (modulo a TBox)

Be a revision algorithm between two knowledge bases ALC which TBox are identical, $\Psi = T \cup A_\Psi$ and $M = T \cup A_\mu$ which A_μ and A_Ψ are ABox and T is a TBox. Assertions are the same in both BC, they are not revised (see [14], [15] for details)

4.2.1 Parameters and results of the algorithm

The algorithm takes both ABox and A_Ψ A_μ and TBox $T = \{T \sqsubseteq K\}$ input. A cost function associated with a literal 'n' a numeric value cost (n) > 0, where a literal is either an atomic concept (positive literal) or a concept of the form ¬ A where A is atomic (negative literal). Intuitively, more cost (n), the more difficult it is to renounce the truth of an assertion n (a). The result of the algorithm is a disjunction of ABox D , representing the result of the revision of A_Ψ by A_μ modulo T, that is to say the revision of Ψ by M. Indeed, when two alternatives for the revision of A_Ψ by A_μ can't be separated by cost, the result can't always be expressed as a single ABox.

4.2.2 Outline of the algorithm by reviewing A_Ψ by A_μ modulo T

The algorithm tables (for details see [14]) is applied to $A_\Psi \cup A_\mu$, it can test its satisfiability modulo T. If $A_\Psi \cup A\mu$ is satisfiable, that is to say consistent, $(T \cup A_\Psi) + (T \cup A_\mu)$ is equivalent to $T \cup (A_\Psi \cup A_\mu)$, there is no need to change A_Ψ. However, if $T \cup A_\Psi \cup A_\mu$ is not satisfiable, we must modify A_Ψ to make it consistent with A_μ modulo T. For this, we will "*fix*" the conflicts generated by the application of rules from $A_\Psi \cup A_\mu$, A_μ without change. For this " *repair* " it is not enough to remove conflict, these conflicts derive formulas which must also be deleted. for a more detailed review, A_Ψ and A_μ are complemented by the method tables before being combined.

5. Conclusions

A system of case-based reasoning (CBR) is based on domain knowledge, in addition to the base case. The acquisition of new domain knowledge should improve the accuracy of such a system.

This paper presents an approach to acquire domain knowledge based on failures of a CBR system. This approach has been implemented in FRAKAS.

FRAKAS proposed a new way to perform knowledge acquisition in CBR systems producing solutions that are consistent with the domain knowledge. This prototype is based on a description logic representation, conservative adaptation is based on the principle of minimal change to a knowledge base that makes this change by revising the bases case in our work we propose an algorithm to use for revision on ABox (modulo a TBox) for revising a knowledge base.

In future work we plan to work on our Implementable choosing a scope and make it generic.

References

[1] Aamodt et E., Plaza, A. (1994). Case-based Reasoning : Foundational Issues, Methodological Variations, and System Approaches. AI Communications 7(1) 39–59.

[2] Blansché, A., Cojan, J., Dufour-Lussier, V., Lieber, J., Molli, P., Nauer, E., Skaf-Molli, H., Toussaint, Y. (2010). Taaable 3 : Adaptation of Ingredient Quantities and of Textual Preparations. *In*: *Workshops of the 18th International Conference on Case-Based Reasoning (ICCBR*-10).

[3] Badra, F. (2009). Extracting adaptation knowledge in case-based reasoning . These, University Henri Poincaré - Nancy I.

[4] Lieber, J. (2008). Contributions to the design of systems-based reasoning case. These, University Henri Poincaré - Nancy I.

[5] McCarthy, J. (1977). Epistemological Problems of Artificial Intelligenc. *In*: Proceedings of the *5th InternationalJoint Conference on Artificial Intelligence (IJCAI'77), Cambridge (Massachussetts)*, p. 1038–1044.

[6] Cordier, A. (2008). Interactive and Opportunistic Knowledge Acquisition in Case-Based Reasoning. These University de Lyon.

[7] Baader, F., Calvanese, D., McGuinness, D., Nardi, D., et Patel-Schneider, P., editors. (2003). *The Description Logic Handbook.* Cambridge University Press, cambridge, UK.

[8] Baader, F., Calvanese, D., McGuinness, D. L., Nardi, D. et Patel- Schneider, P. F. (2003). The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press.

[9] Meyer, T., Lee, K., et Booth, R. (2005). Knowledge integration for description logics. *In*: Proceedings of the AAAI'05, p. 645-650 .

[10] Lieber, D. (2006). A Definition and a Formalization of Conservative Adaptation for Knowledge-Intensive Case- Based Reasoning – Application to Decision Support in Oncology (A Preliminary Report). LORIA.

[11] Alchourrón, C. E., Gärdenfors, P., et Makinson, D. (1985). On the logic of theory change : partial meet functions for contraction and revision. *Journal of Symbolic Logic*, 50, 510–530.

[12] d'Aquin, M., Lieber, J., et Napoli, A. (2009). Adaptation Knowledge Acquisition ;a Case Study for Case-Based Decision Support in Oncology". *Computational Intelligence (an International Journal)*, 22 (3/4) 161–176.

[13] Katsuno, H., et Mendelzon, A. (1991). Propositional knowledge base revision and minimal change. *Artificial Intelligence*, 52 (3) 263–294.

[14] Cojan, J. (2011). Application of the theory of knowledge revision in case-based reasoning. Thèse University Henri Poincaré – Nancy.

[15] Qi, G., Liu, W., et Bell, D. A. (2006). Knowledge base revision in description logics. *In*: Fisher, M., van der Hoek, W., Konev, B. and Lisitsa, A., éditeurs : JELIA, volum 4160 de Lecture Notes in *Computer Science*, p. 386_398. Springer.

[16] Wilke, W., Vollrath, I., Althoff, K.-D., Bergmann, R. (1996). A Framework for Learning Adaptation Knowledge Based on Knowledge Light Approaches. *In*: Proceedings of the workshop of Adaptation in Case-Based Reasoning (at ECAI'96), p. 235–242, Budapest.