International Conference on Control, Engineering & Information Technology (CEIT'13)
Proceedings Engineering & Technology - Vol.2, pp. 158-163, 2013
Copyright - IPCO

Knowledge Acquisition with system FrakaS-DL

Ouided.Hioual[#1], Mohamed Tayeb.Laskri[*2]
[#]Department of informatics, Abbas Laghrour university – Khenchla-Algeria
hioual_ouided@yahoo.fr
[*]Laboratory of Computer Science LRI -Badji Mokhtar university- Annaba-Algeria
laskri@univ-annaba.org

***Abstract*—In Case Based Reasoning (CBR), knowledge acquisition plays an important role as it allows to progressively improve the system's competencies. 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.**

I. Introduction

Case-based reasoning (CBR) is a reasoning paradigm which consists in solving new problems by adapting solutions of previously solved problems. This process is supported by various knowledge used to reason on cases. In particular, adaptation knowledge is of major importance: it is used during the retrieval step to retrieve a good source case (e.g. a case easy to adapt) and, of course, during the adaptation step to build the solution to the current problem. Unfortunately, knowledge management in CBR is still a difficult problem.

In Case Based Reasoning (CBR), knowledge acquisition plays an important role as it allows to progressively improve the system's competencies. One of the approaches of knowledge acquisition consists in performing it while the system is used to solve a problem. An advantage of this strategy is that it is not to constraining for the expert: the system exploits its interactions to acquire pieces of knowledge it needs to solve the current problem and takes the opportunity to learn this new knowledge for future use.

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 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].

The remainder of this paper is organized as following. In Section 2, we present 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.

II. 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 systems are knowledge-based systems (KBS) which, if we follow Richter's proposition [3], make use of four distinct knowledge sources: domain knowledge, cases, similarity knowledge and adaptation knowledge. But one can have an unified view of the knowledge involved in CBR systems as there exists close relations between the different knowledge containers.

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.

$$\begin{tikzcd}
\mathtt{srce} \arrow[d] & \mathtt{tgt} \arrow[l, "retrieval"'] \arrow[d] \\
\mathtt{Sol(srce)} \arrow[r, "adaptation"'] & \mathtt{Sol(tgt)}
\end{tikzcd}$$

Fig. 1 CBR classical paradigm.

Cycle of reasoning in CBR:
The principle of CBR, reusing a past problem-solving experience to solve a similar problem, is simple, but the implementation of this principle remains complex and raises a certain number of questions. How do we represent an experience? What is a similar problem? How do we reuse an experience and adapt it to the present situation? What can be retained from a specific problem-solving experience?
At the highest level of generality, a general CBR cycle may be described by the following four processes:

1. Retrieve similar cases to the problem description.
2. Reuse a solution suggested by a similar case.
3. Revise or adapt that solution to better fit the new problem if necessary.
4. Retain the new solution once it has been confirmed or validated.

A new problem is solved by retrieving one or more previously experienced cases, reusing the case in one way or another, revising the solution based on reusing a previous case, and retaining the new experience by incorporating it into the existing knowledge-base (case-base). The four processes each involve a number of more specific steps, which will be described in the task model. In figure 2, this cycle is illustrated

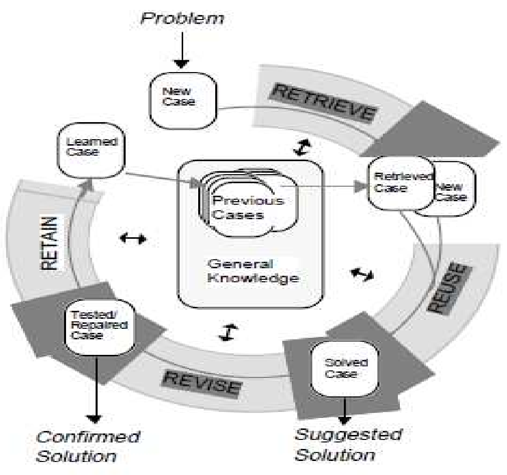

Fig. 2 A CBR cycle (Aamodt-Plaza-94)

An initial description of a problem (top of figure) defines a new case. This new case is used to RETRIEVE a case from the collection of previous cases. The retrieved case is combined with the new case - through REUSE - into a solved case, i.e. a proposed solution to the initial problem. Through the REVISE process this solution is tested for success, e.g. by being applied to the real world environment or evaluated by a teacher, and repaired if failed. During RETAIN, useful experience is retained for future reuse, and the case base is updated by a new learned case, or by modification ofsome existing cases[9],[20].

We note SDK (system domain knowledge domain) : 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].

A. *Acquiring CBR knowledge*

Solutions produced by CBR systems may not be satisfactory because of either a lack of sufficient knowledge or imperfectly described knowledge, leading to reasoning failures.Thus, many research work address the learning component in CBR systems along several perspectives.
One of these perspectives characterizes the different knowledge containers targeted by the learning process [3]: case’s vocabulary, cases, similarity and solution transformation (i.e. adaptation knowledge). Some approaches consider similarity and adaptation knowledge as distinct and learn them separately [7]. We defend the idea that, ideally, only domain and adaptation knowledge should be learned and similarity knowledge should be deduced from adaptation knowledge.
Another perspective characterizes the knowledge source used by the learning process [8]. Some approaches use the content of the knowledge containers, in particular those who rely on machine-learning or "off-line" techniques in order to explicit knowledge. Other "on-line" approaches, by contrast, aim at acquiring new knowledge that is not already in the system through interactions with the environment . Learning takes place during the use of the system and aims at acquiring

domain knowledge. The evaluation of the adapted solution may highlight the fact that it does not meet the requirements of the target problem. In this situation, a reasoning failure occurs and is handled by a learning process. The expert is involved in the identification of faulty knowledge and a repair process is triggered to correct it.

III. 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.

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.[21]

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.

IV. FRAKAS (Failure Analysis for domain Knowledge AcquiSition)

FRAKAS is an illustration of the FIKA principles (Failure-driven Interactive Knowledge Acquisition, FIKA defines a general approach for interactive and opportunistic acquisition of knowledge in case-based reasoning.). 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][10]

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.

The FRAKAS approach aims to facilitate the acquisition of domain knowledge. This knowledge, although used to adapt cases, is not supposed to be linked to the cases. In FRAKAS, the identification of knowledge to be acquired is done, not by analysing the reasoning, but by analysing the solution. In the case of failure, the solution is analysed by the expert who must identify inconsistencies in the solution using his own knowledge. The system is able to infer, from the analysis of these inconsistencies, new knowledge which will allow it to avoid repeating the mistake in future.

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
```

Reasoning in FRAKAS.

An assumption is made that the CBR system is capable of performing consistent reasoning in the cases using the available domain knowledge. The proposed built by the

system is presented to the oracle who is able to decide if it is valid or not (i.e., if the proposition works or not). The role of the expert is then to highlight faulty knowledge if the proposition is not satisfactory, which amounts to highlighting the parts of the proposition that are not correct[10].
The knowledge acquisition in FRAKAS is:
- Opportunistic, as it exploits failures to trigger a learning process;
- Interactive, as it involves the user during the CBR session, through interactions;
- Incremental, as pieces of knowledge are added progressively to the domain knowledge.
The knowledge learned by a system implementing the FRAKAS principles is used to repair failed adaptations and to improve the quality of the solution proposed for the current problem. This knowledge is also stored and reused to prevent similar failures from occurring again in further reasonings.

FRAKAS principles
The FRAKAS principles are illustrated in figure 3

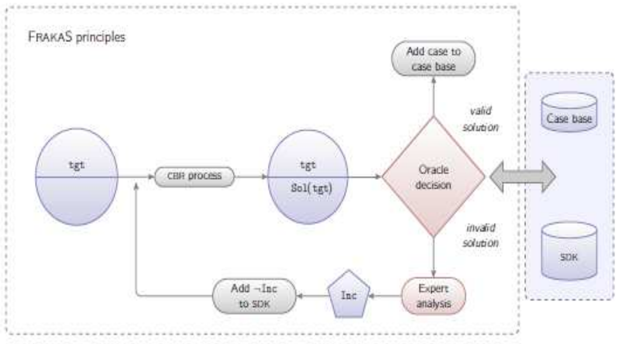

Fig. 3 FRAKAS principles

This figure describes the main FRAKAS principles and the links with the knowledge base (on the right of the figure). Circles represent cases, rounded rectangles are processes (the expert analysis involves interactions between the expert and the system). Inc is a piece of knowledge that is built during the reasoning cycle and that is going to be added to the knowledge base.
The CBR process exploits a knowledge base to produce a candidate solution. When the candidate solution is judged not valid (i.e. it does not work) by the oracle, the expert has to identify a subset of inconsistent knowledge (denoted Inc on the figure). From this subset of knowledge, the system is able to learn a new piece of knowledge. This new piece of knowledge is added to the knowledge base. The improved knowledge base allows the system to produce a new candidate solution for the current problem. The process is iterated until the expert validates a solution proposed by the systems, i.e. until the system finds a working solution[10].
The CBR process implemented in FRAKAS exploits a case base together with the system domain knowledge base (denoted by SDK). The cases contained in the case base are assumed to be consistent with SDK, they often contain pieces of knowledge coming from experience. These pieces of knowledge cannot always be explained by the domain knowledge but are nonetheless often very useful, that is why they are valuable.

A. The used Formalism : 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. [12]
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:

$\top$, $\bot$, $C \sqcap D$, $\neg C$, $C \sqcup D$, $\forall$ 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 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.

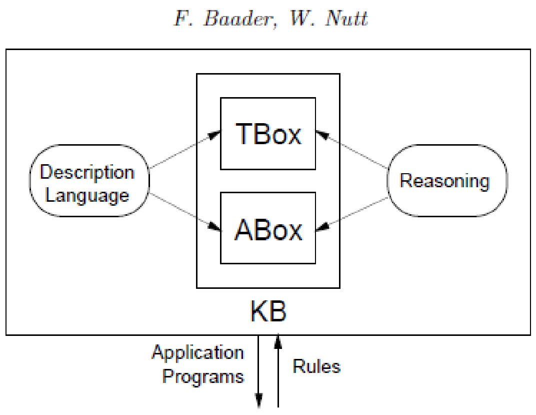

Fig. 4 Architecture of a knowledge representation system based on Description Logics.

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) Є r^I) and, to an instance has an element a^I of r^I. [12]

Given an interpretation *I*, we say that *I* satisfies a concept axiom $C \sqsubseteq D$ (respectively, a role inclusion axiom $R \sqsubseteq S$) if $C^I \subseteq D^I$ (respectively, $R^I \subseteq S^I$). An interpretation I is called a model of a TBox T , written I |= T , iff it satisfies each axiom in T . We use Mod(T) to denote all the models of a TBox T . Two TBoxes T_1 and T_2 are equivalent, written $T_1 \equiv T_2$, iff $Mod(T_1) = Mod(T_2)$. A named concept C in a terminology T is unsatisfiable iff, for each model I of T , $C^I = \Phi$. A terminology T is inconsistent iff it does not have a model, and it is incoherent iff there exists an unsatisfiable named concept in T . Incoherence is a kind of logical contradiction which has been widely discussed. When there is a concept in a TBox, if the TBox is inconsistent, then it must be incoherent.

Inferences:

DL system doesn't 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. 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 $C^I \neq Ø$; (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 $C^I \sqsubseteq D^I$ 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 ≡ 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 ≡ D

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

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 Є 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). [11]

A. *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[19] .

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 [14] for more 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],[17],[18],[15], [13]: a revision operator '$\circ$' combines two knowledge bases Ψ and μ knowledge base $\Psi \circ \mu$ which, intuitively , is obtained by minimal change on Ψ to be consistent with μ.[16]

In this paper, We consider only revision of terminologies in DLs and we have adapted the Dalal revision operator for revising terminologies

To adapt Dalal's revision operator to DLs, we need to define the "difference set" between two models. By treating each concept name as a propositional variable, we can define the difference between two models in DLs in a similar way as the difference set between two models in propositional logic. Suppose we want to revise a TBox T_1 using another one T_2. Following the idea of Dalal's revision operator, in our revision operator, we revise some models of T_1 to make them as models of T_2.(see [22,23] for more details)

We consider postulates for revision operators in DLs given in [22], which are reformulated from Katsuno and Mendelzon's postulates (KM postulates) in [24].

(G1) $Mod(T1 \circ T2) \subseteq Mod(\varphi)$ for all $\varphi \in T2$.

(G2) If $Mod(T1) \cap Mod(T2) \neq Ø$, then $Mod(T1 \circ T2) = Mod(T1) \cap Mod(T2)$.

(G3) If $T2$ is consistent, then $Mod(T1 \circ T2) \neq Ø$.

(G4) If $Mod(T) = Mod(T_1)$ and $Mod(T') = Mod(T_2)$, then $Mod(T \circ T') = Mod(T_1 \circ T_2)$.

(G5) $Mod(T_1 \circ T_2) \cap Mod(T_3) \subseteq Mod(T_1 \circ (T_2 \cup T_3))$.

(G6) If $Mod(T_1 \circ T_2) \cap Mod(T_3)$ is not empty, then

$Mod(T_1 \circ (T_2 \cup T_3)) \subseteq Mod(T_1 \circ T_2) \cap Mod(T_3)$.

(G1) guarantees that every axiom in the new TBox can be inferred from the result of revision. (G2) says that we do not change the original knowledge base if there is no conflict. (G3) is a condition preventing a revision from introducing unwarranted inconsistency. (G4) says the revision operator should be independent of the syntactical forms of knowledge bases. (G5) and (G6) together are used to ensure minimal change.

V. 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 Implementation choosing a scope and make it generic

REFERENCES

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

[2] A. Cordier, B. Fuchs, and A. Mille. Engineering and Learning of Adaptation Knowledge in Case-Based Reasoning. In Proceedings of the 15th International Conference on Knowledge Engineering and Knowledge Management (EKAW-06), pages 303–317, 2006.

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

[4] B. Fuchs, J. Lieber, A. Mille, and A. Napoli. A general strategy for adaptation in casebased reasoning. Technical Report RR-LIRIS-2006-016, LIRIS UMR 5205 CNRS/INSA de Lyon/University Claude Bernard Lyon 1/University Lumiere Lyon 2/Ecole Centrale de Lyon, 2006.

[5] B. Smyth and M.T. Keane. Retrieving Adaptable Cases. In S.Wess, K.-D. Althoff, and M.M. Richter, editors, Topics in Case-Based Reasoning – First European Workshop (EWCBR’93), Kaiserslautern, LNAI 837, pages 209–220. Springer, Berlin, 1994.

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

[7] D.B Leake, A. Kinley, and D. Wilson. Multistrategy learning to apply cases for case-based reasoning. In Third International Workshop on Multistrategy Learning, pages 155–164, Menlo Park, CA, 1996. AAAI Press.

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

[9] J. McCarthy. “Epistemological Problems of Artificial Intelligence”. In Proceedings of the *5th InternationalJoint Conference on Artificial Intelligence (IJCAI’77), Cambridge (Massachussetts)*, pages 1038–1044,1977.

[10] A. Cordier. “Interactive and Opportunistic Knowledge Acquisition in Case-Based Reasoning” .Thèse University de Lyon (2008).

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

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

[13] Meyer, T., Lee, K. et Booth, R. Knowledge integration for description logics. In Proceedings of the AAAI'05, pages 645_650 (2005).

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

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

[16] M. d’Aquin, J. Lieber, et A. Napoli. “Adaptation Knowledge Acquisition; a Case Study for Case-Based Decision Support in Oncology”. *Computational Intelligence (an International Journal)*, 22(3/4) :161–176, 200

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

[18] J. Cojan. “Application of the theory of knowledge revision in case-based reasoning”. Thèse University Henri Poincaré – Nancy 1 2011

[19] 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*, pages 386_398. Springer.

[20] Wilke, W., Vollrath, I., Althoff, K.-D., and 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), pages 235–242, Budapest.

[21] Cordier, A., Fuchs, B., Lieber, J., and Mille, A. (2007). Acquisition de connaissances du domaine d’un système de RàPC : une approche fondée sur l’analyse interactive des échecs d’adaptation - le système FrakaS. In Cordier, A. and Fuchs, B., editors, Actes du 15ème atelier de raisonnement à partir de cas (RàPC’07), pages 57–70, Grenoble. Plateforme AFIA.

[22] G. Qi,W. Liu, and D. Bell. Knowledge baserevision in description logics. In Proc. of JELIA, pages 386–398, 2006.

[23] M. Moretto Ribeiro and R. Wassermann. Base revision in description logics – preliminary results. In *Proc. of IWOD*, pages 69–82, 2007.

[24] Katsuno and A.O. Mendelzon. Propositional knowledge base revision and minimal change. *Artif. Intell.*, 52(3):263–294, 1992.