Université de Lyon **École doctorale Informatique et Information pour la Société**

Interactive and Opportunistic Knowledge Acquisition in Case-Based Reasoning

THÈSE

pour l'obtention du

Doctorat de l'Université de Lyon I

(spécialité informatique)

par

Amélie Cordier

Composition du jury

Laboratoire d'InfoRmatique en Images et Systèmes d'information — UMR 5205

Remerciements

Mon jury est un véritable honneur et je tiens à remercier chaleureusement et sincèrement chacun des membres d'avoir accepté d'y prendre part. Merci à Agnar Aamodt et David Leake d'avoir rapporté sur cette thèse. Leur enthousiasme et leurs remarques sur ce travail m'ont donné, s'il en fallait encore, une motivation supplémentaire pour poursuivre ma route. Merci à Eyke Hüllermeir d'avoir répondu avec tant de spontanéité à notre invitation. Je remercie Sylvie Després pour ses encouragements et sa confiance. C'est non seulement un honneur, mais un plaisir pour moi de la compter parmi les membres de ce jury, comme cela a été un honneur et un plaisir de travailler avec elle par le passé. Cette thèse n'aurait probablement pas existé si je n'avais pas rencontré Jean Lieber. Il m'a procuré ce fameux déclic qui change tout et a été un “maître” pour moi durant ces dernières années. Apprendre à faire de la recherche à ses côtés est une chance et je le remercie, probablement trop peu, de m'en avoir fait profiter avec tant de générosité.

Je remercie Béatrice Fuchs d'avoir encadré ce travail. Je remercie également Alain Mille pour son encadrement scientifique mais aussi pour sa confiance, pour avoir respecté mes choix et pour m'avoir soutenue dans mes projets. Dans les hauts et les bas, Alain m'a communiqué son enthousiasme et son dynamisme ; j'espère un jour suivre ses traces.

Cette thèse est un ouvrage collectif : nombreux sont ceux qui, parfois sans le savoir, ont contribué à sa réalisation. Quelques mots au début de cette thèse me semblent bien peu pour vous remercier, chers membres du collectif[1], pour tout ce que je vous dois ; cela dit, “je vais essayer”. J'ai une dette considérable envers Annie et Mike qui m'ont offert, sans compter, asile, relectures et traductions : pour tout cela, et pour le reste, mille mercis. Je remercie également Mauricio pour ses illustrations et Phil D. qui a gentiment accepté de poser pour nous. Les bons conseils et les encouragements d'Alphonse et de Léon doivent également être salués bien bas.

Mes pensées vont souvent aux membres de ma famille qui m'encouragent aveuglément et me témoignent leur confiance depuis toujours. L'enthousiasme de chacun, et en particulier de mon frère et de mes cousins et cousines a toujours été une source de motivation surprenante. Merci à vous, je suis heureuse d'appartenir à cette grande famille.

Mes amis m'ont supportée, dans les deux sens du terme, depuis plusieurs années et le plus surprenant, c'est que certains sont toujours mes amis ! Magali, Marie, Nathalie, Stéphanie, Séverine, Frank, Olivier, votre amitié est inestimable, merci d'être sur cette planète.

Aux meilleurs colocataires du monde, Léonardo et Elayne, je dois beaucoup plus que des remerciements ! Vous êtes des magiciens : en votre présence, l'année qui devait être la plus difficile s'est transformée en la plus belle des années.

Enfin, je tiens à remercier mes compagnons de café, et parfois de galère pour la bonne ambiance qu'ils font régner là où ils passent : Laure, Sophie, Thierry, David, Thierry (un autre), Lionel, Cathy, David (un autre aussi), Noël, Brice, Sébastien, Francois-Xavier, Marianne, et tous les autres.

Papa, maman, je ne vous ai pas oubliés, mais je n'ai pas assez de place ici pour dire tout ce que je vous dois.

[1]Si vous pensez avoir été oubliés du collectif, n'hésitez pas à m'envoyer un mail, je publierai un correctif dans mon HDR, ou dans ma prochaine thèse...

À Gros-Père, en espérant qu'il ne m'en veuille pas trop...

Contents

List of Figures

List of Taaables

List of Definitions

List of Hypotheses

List of Examples

Let us change our traditional attitude to the construction of programs. Instead of imagining that our main task is to instruct a computer what to do, let us concentrate rather on explaining to human beings what we want a computer to do.

Donald Knuth

1 Introduction

Have you ever used a recipe for duck à l'orange to make chicken with lemon? You have? Then you have already done some case-based reasoning. Reusing past experience and transforming it to adapt it to your needs is conducting case-based reasoning. To be precise, the expression "case-based reasoning" has been used for around thirty years and originates in studies on cognitive sciences and artificial intelligence. The simple idea of using existing problems as a basis for solving new ones seems much more attractive than starting from scratch on the basis of a set of rules and a complex theory. This is why, at a time when rule-based systems, which are well known tools of artificial intelligence, were beginning to show their limits, studies on case-based reasoning provoked a certain interest in the world of research. Case-based reasoning has been widely studied with a view to developing this reasoning paradigm in artificial intelligence tools, and several models were produced. In this paradigm, past experience is conventionally represented as an entity called a "case" which is divided into two sub-parts: "problem" and "solution". Faced with a new problem such as cooking chicken with lemon, one looks for a similar problem for which the solution is known, such as cooking duck with orange, and one adapts this solution, replacing missing ingredients with similar ones which are available.

The main problem of case-based reasoning is to know how to find relevant experiences and to make use of them in a given situation. The similarity between recipes for duck à l'orange and chicken with lemon may seem obvious to any human being with rudimentary cooking experience, but it does not appear so to a computer system. The system requires a considerable amount of knowledge to be able to infer that, in preparing a sweet and sour recipe, it is possible to replace oranges with lemons and that, from a cooking point of view, duck and chicken may be deemed to be similar and therefore interchangeable. There lies the critical point of the case-based reasoning paradigm: knowledge. No matter how efficient the reasoning mechanism, the results will be mediocre if the quality of the knowledge on which it is based is poor. Expert systems went out of favour in the mid 80s, not because of the quality of inference engines, but because of the great difficulty in building up bases of rules constituting the systems knowledge. Facts, however, were easier to acquire. This even constitutes an argument in favour of the relative easiness to acquire cases in case-based reasoning.

The purpose of a so-called artificial intelligence tool is to solve, or help solve, a problem as would an expert, i.e., a domain specialist. In order to reach this objective, one must therefore provide the systems with knowledge similar to that of the expert. Questions are then raised as to the nature of the expert: How does he solve a problem? What gives him this expert status? What is the knowledge he relies on? The expert's knowledge is not to be found in books describing a domain theory: it is not a set of rules. It is of a different nature. To make lemon chicken, one "knows" one should lower the cooking time in the initial recipe because one "knows" chicken takes less time to cook: it is practical knowledge, coming from experience. Research on knowledge acquisition aims to bring answers to the issue of the "transfer" of the expert's knowledge to the computer system. Beyond questioning the expert's knowledge, other questions are raised: the questions relating to tools and methods to be applied. How should the expert's knowledge be modelled in the system? What methodologies and tools must be developed to assist with the knowledge acquisition process? Is it possible, by adapting certain methods, to anticipate or predict obstacles which might impede the process? Case-based reasoning tools are often called "learning" tools, inasmuch as they acquire new cases, i.e., new experiences, as they solve problems. If the duck à l'orange you have just made proves to be good, you will quite probably keep the recipe to reuse it some day. But memorisation only is not sufficient to produce real "learning" systems. If you do not memorise the general knowledge that "chicken cooks faster than duck", you won't be able to reuse it in another context. This additional knowledge is expert knowledge and is used to reason over past experiences and to transform known solutions. A system which does not have the ability to acquire such knowledge will not be able to solve new problems (i.e., problems that cannot be solved with the initially available knowledge), since it will not be able to perform new reasoning (i.e., transform existing solutions): then it will only be a tool for remembering past experiences. Thus, to turn a case-based system into a real learning system, one must give it the ability to acquire cases, and also useful knowledge to reason on cases.

The thesis discussed in this work proposes to bring a contribution which may be considered from the point of view of acquisition of knowledge and that of case-based reasoning. It is based on two observations related to present case-based reasoning tools and to their use. The first observation is that methods and tools for the acquisition of knowledge other than cases are sadly lacking in present systems. The second observation is that this knowledge is difficult to grasp, relates mostly to experience, and must therefore be obtained from experts. On the basis of these observations, we propose a generic method and tools for interactive and opportunistic acquisition of knowledge applicable to case-based reasoning systems.

In observing the life cycle of a case-based reasoning tool, one notices that there is very often an initial phase of acquisition of knowledge where a first knowledge base is built in order to initialise the system. Later on, it is possible to acquire knowledge "as you go" to correct or to complete the initial knowledge base. In this work we are concerned with the second phase, i.e., the one that occurs as we go along.

At the core of the method and the tools proposed are the expert and the system which are constantly interactive. Considering that the system cannot acquire knowledge without interaction with its environment, one may consider that it cannot "learn" by itself. As for the expert, he has to explain, through interaction with the system, the knowledge which he is used to handling without being fully aware of it. In this way, he "learns", via the system, better to understand what he knows, which he would not have realised on his own. Thus, we are focusing on the "expert-system" pair which we consider as a system learning through interactions. The long and fastidious acquisition of knowledge from experts is a task fraught with great difficulties. Due to the experimental nature of this knowledge, there is no "catalogue" of knowledge for a given domain. When it comes to modelling it, one should ideally implement a methodology guaranteeing that any knowledge is taken into account. However, it is in fact impossible for it to be exhaustive. Indeed, this knowledge is precise, linked to a particular situation, to a given context, and it is often very difficult to elicit it "out of context". The opportunistic aspect of the approach which we are describing here proposes a solution to this problem of putting the expert in context during the knowledge acquisition process. When a case-based reasoning system solves a problem, it always proposes the "best" solution it can find (that is not necessarily the best solution for the problem) given the knowledge available at the time. If the solution does not satisfy the expert, this means that the knowledge used by the system to reach that solution is incorrect or incomplete and can therefore be improved. Let us suppose that the recipe for lemon chicken was proposed by a case-based reasoning tool which did not specify how to adapt the cooking time, then when following the recipe we would have obtained an overcooked and not very tasty chicken. New knowledge would have to be added to the system to avoid a re-occurrence of this unfortunate mistake. An unsatisfactory solution is called a "reasoning failure". We propose to use reasoning failures as triggers for a knowledge acquisition process. When a reasoning failure happens, the expert is already in a problem-solving situation and therefore in a context favourable to the acquisition of specific knowledge related to this particular problem. In this way, the opportunistic aspect of the approach places the expert in the right situation to facilitate the identification of knowledge to be acquired, without disturbing his main task or significantly increasing his workload.

In the next chapter (chapter 2) of this thesis, we conduct a study of case-based reasoning along two interlinked lines: reasoning processes and knowledge. A good understanding of the processes of case-based reasoning facilitates the identification of the knowledge involved and of the part it plays in the various reasoning phases. When studying case-based reasoning from the point of view of knowledge engineering, we will show how knowledge modelling constitutes a particularly delicate problem. This study also demonstrates that the quality of a case-based reasoning system is essentially linked to the quality of the knowledge base on which it rests and illustrates the usefulness of progressive knowledge acquisition approaches which go beyond a simple accumulation of cases.

The FIKA principles (Failure-driven Interactive Knowledge Acquisition) are described in chapter 3. FIKA provides a general strategy for the interactive and opportunistic knowledge acquisition which can be implemented in case-based reasoning systems. In this chapter, we describe the main ingredients characterising the originality of the approach (knowledge intensive, interactive, opportunistic, expert-centred) and we define the various concepts which will be used throughout the manuscript. We illustrate how FIKA may have an impact on the conception and implementation of case-based reasoning tools by anticipating the issue of knowledge acquisition on the way. Finally, we situate this work in the context of present research in this domain: we discuss the exploratory aspect of this work and show how it constitutes a step towards the building of a framework for the construction of generic tools to support knowledge acquisition in case-based reasoning systems.

IAKA and FRAKAS, presented in chapters 4 and 5 respectively, are two examples of the use of FIKA principles in two different contexts. IAKA proposes an interactive acquisition approach based on the decomposition of reasoning into reasoning steps. Through this decomposition, the knowledge involved in the reasoning is more finely identified and therefore easier to amend. In FRAKAS, it is the global solution produced by the system that is analysed by the expert. Therefore, the acquisition rests on expert analysing and checking its coherence with respect to his own knowledge. The system's reasoning process is seen as a black box (monolithic process). Both these approaches validate the applicability of the FIKA principles on different kinds of CBR systems. Furthermore, formalisations proposed in IAKA as well as FRAKAS remain at a fairly general level so that they can be reused in other applications without major constraints. IAKA and FRAKAS have both been implemented in prototypes developed in different domains: IAKA-NF and FRAKAS-PL. IAKA-NF illustrates the principles of adaptation knowledge acquisition by focusing on cases and adaptation knowledge, while FRAKAS-PL focuses on the acquisition of domain knowledge. IAKA-NF has been tested on several numerical functions and FRAKAS-PL has been tested on a knowledge base from the domain of breast cancer treatment.

FIKA, IAKA and FRAKAS principles have been implemented in experimental applications, in toy domains, but have not been tested with a proper expert. However, principles of FIKA were also used in the TAAABLE application, for knowledge acquisition in the cookery domain. TAAABLE is a case-based reasoning application aiming to propose cooking recipes in answer to a user request by adapting existing recipes from a cookery book. Developing TAAABLE involved building an important base of culinary knowledge. In order to constitute the initial knowledge base, several approaches of knowledge acquisition were used successfully, but the use of this application showed that there were significant gaps in the knowledge base. Several problems in the initial base were solved by the interactive acquisition method which may be called up at any time to improve the system locally. To amend the initial knowledge, an interactive knowledge acquisition strategy based on FIKA was first set up. Several experts worked on amending the knowledge base following these principles, thereby demonstrating in practice the validity of this approach and illustrating the possibility of transfer of the FIKA principles in an application which was not originally created for this purpose.

The main contribution of this work is the FIKA general strategy for interactive and opportunistic knowledge acquisition in case-based reasoning. IAKA and FRAKAS are models based on the FIKA principles in different contexts and are implemented in prototypical systems. The FIKA strategy is also used in a real-world CBR application called TAAABLE. The purpose of this manuscript is hence to describe the FIKA strategy and its application in three different contexts.

Document organisation

Each chapter begins with an abstract followed by a list of the main relevant research references and a list of our related publications. The following page consists of a table of content and an overview of the contributions described in the chapter.

Definition 1.1 (A definition)

When definitions are introduced in a chapter, they are placed in an environment such as this one. The list of definitions can be found at the beginning of this manuscript.

Hypothesis 1.1 (An hypothesis)

In the same way, hypotheses are set in environments like this one. The list of hypotheses can be found at the beginning of this manuscript.

An example

Important concepts are often illustrated by examples that are organised in environments like this one. As for definitions and hypotheses, a list of the examples is available at the beginning of this document.

Last, important terms appearing in the text are referred to in an index available at the end of this document (see appendix C). In the same way, appendix D lists the notations used in this manuscript.

Writing convention

In this manuscript, we often refer to "the expert" by the pronoun *he*; it is in no way an offence neither a hasty hypothesis on the expert's gender. We only apply a common English rule, as explained, for example, in the book *The elements of style*, pp. 60-61, [Strunk and White, 1979]:

> The use of *he* as pronoun for nouns embracing both genders is a simple, practical convention rooted in the beginnings of the English language. *He* has lost all suggestion of maleness in these circumstances. The word was unquestionably biased to begin with (the dominant male), but after hundreds of years it has become seemingly indispensable. [...] No one need fear to use *he* if common sense supports it. The furor recently raised about *he* would be more impressive if there were a handy substitute for the word. Unfortunately, there isn't —or at least, no one has come up with one yet. If you think *she* is a handy substitute for *he*, try it and see what happens.

We have decided not to try with *she* and keep using *he*. If you want to know more about duck à l'orange or about interactive knowledge acquisition in CBR, it is time to continue your reading.

Experience is the name everyone gives to their mistakes.

Oscar Wilde

2 A GENERAL OVERVIEW OF CASE-BASED REASONING

This study takes place within the framework of case-based reasoning, a reasoning paradigm which, in order to solve a problem, is based on the reuse of past experiences. This chapter outlines the principles which characterise this type of reasoning and presents an overview of work relating to this field of research. As an introduction, a general and historical description of the CBR paradigm is presented. Then, the study is conducted along two main axes: reasoning mechanisms on one side and the knowledge involved on the other. This focus of study has been chosen in order to underline the various CBR specificities that have to be taken into account when defining knowledge learning mechanisms for CBR systems. Hence, this chapter constitutes a background study that anchors this work in current CBR research. The synthesis of this chapter refers back to studies which have a particular link with the approach described in this document.

Related publications:
[Cordier, 2004], [Cordier et al., 2006a], [Cordier et al., 2006b], [Cordier et al., 2008b]

Main related work:
[Riesbeck and Schank, 1989], [Aamodt and Plaza, 1994], [Newell, 1982], [López de Mántaras et al., 2005]

Keywords:
Case-Based Reasoning, Knowledge Level

Contents

Contributions of this chapter

- State of the art in CBR
- Grounding of the knowledge acquisition approach
- Identification of the CBR specificities that have to be taken into account when defining a knowledge acquisition approach

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 summarised as in figure 2.1.

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

The classical CBR principle: to solve a target problem, retrieve a source case and adapt the solution to fit the target problem requirements.

Figure 2.1: CBR classical paradigm.

The main purpose of the approach described in this thesis is to provide an approach for knowledge learning in CBR systems. Defining such an approach requires a good understanding of the CBR process and an identification of its specificities. Indeed, both the reasoning process and the knowledge involved are to be considered when implementing a knowledge learning approach. This chapter aims at describing the CBR process and at giving the required background knowledge to better understand the research question we address in the remaining of this document.

The first section of this chapter recalls the historical foundations of CBR and positions it with regard to other paradigms of artificial intelligence. Section 2.2 describes the CBR process and its various phases, as they are usually presented in the specialised literature. Section 2.3 studies CBR as a knowledge-based system, from the knowledge engineering point of view. It also deals with present CBR research subjects and shows how the problematics of knowledge acquisition becomes important particularly when creating CBR systems *capable of learning how to solve problems*. The synthesis in section 2.4 draws links between the different studies introduced in this chapter and the thesis supported here. Finally, section 2.5 concludes this chapter.

2.1 Case-based reasoning foundations

How does one make a rhubarb pie when one does not know the recipe? A possible solution is to use the recipe for a similar pie, of which one knows the recipe, for example apple pie, and to adapt it by replacing apples with rhubarb. To adopt such an approach is to use case-based reasoning: to solve a particular problem, one tries to remember past experience and one modifies it so as to apply it to the present situation. This type of reasoning, often used instinctively by human beings, was highlighted by studies on cognitive sciences and was then studied more widely. The aim of this section is to present some founding studies of case-based reasoning and to position this approach with respect to other methodologies of artificial intelligence.

2.1.1 Historical foundations and cognitive aspects

In 1977, Schank and Abelson, researchers in cognitive science, were particularly interested in human memory and its part in understanding and explaining situations [Schank and Abelson, 1977], [Schank, 1982]. They showed the crucial part played by remembering past experience in the learning and problem-solving process. At that time they evolved the notion of *scripts*. Scripts allow us to structure conceptual memory. They are knowledge structures permitting the organisation of episodic information (in "cases"). Scripts come from experiences and are built up by repeated exposure to similar situations like "go to a restaurant". Depending on their approach, humans would have a large number of scripts available in their memory. It would then be easy enough to remember a relevant script and to apply it to react in a given situation.

The multiplicity and wide diversity of scripts make it difficult to organise them as a structured memory. Schank thus proposes *MOPs (Memory Organization Packets)* which enable the organisation of scripts into *packets* to store and find them more easily. It is then possible, via the MOPs, to generalise and specialise the scripts and thus to organise them hierarchically. The scripts existing in MOPs can be remembered and adapted to be reused in any situation. The notion of adaptation is very important, since it is rare to find a script which fits perfectly a given situation. Adaptation then allows us to perform the necessary adjustments to the script, taking into account the specificity of the present situation.

Schank observes that explanations play a central part in learning and that it is often easier to build an explanation by modifying an existing explanation than to build one from scratch. This is why he adds explanation models to the MOP models: *XPs (explanation patterns)*. This model proposes an explicit knowledge structure, used to generate, index and test the explanations in conjunction with episodic memory [Schank, 1986], [Schanck et al., 1994].

The researches initiated by Schank have been followed by numerous studies. The MOP model in particular has been used in the SWALE system [Kass et al., 1986]. This system enables the generation of explanations from descriptions of past situations (descriptions are called cases). In SWALE, MOPs are used to guide the understanding of phenomena and use the principles of CBR to find past experiences when a failure occurs in the explanation. Thanks to the remembered experience, the explanation is then adapted to the new situation. The first computer system which was qualified as "case based reasoning system" is called CYRUS [Kolodner, 1993] and was developed by Kolodner at Yale University. The model of case memory developed in CYRUS was inspired from Schank's dynamic memory model and served as a basis for other well-known CBR systems such as CHEF [Hammond, 1986] and CASEY [Koton, 1988]. The first studies explicitly dealing with CBR were then presented in the United States at the DARPA conference in 1988 [Kolodner, 1988]. Since then, this paradigm has matured. The fundamental principles have been more clearly defined and numerous industrial and experimental applications have demonstrated its usefulness. However, many perspectives for research are still open, with regard to implications in cognitive sciences and also possible applications in artificial intelligence.

2.1.2 Case-based reasoning and artificial intelligence

Case-based reasoning is a paradigm of artificial intelligence, like neural networks, genetic algorithms, multi-agent systems, Bayesian networks, etc. CBR is often considered as a sort of analogical reasoning. Analogical reasoning (*a* is to *b* what *c* is to *d*) is exploited in a number of fields of artificial intelligence [Gick and Holyoak, 1980, Gentner et al., 2001]. It is based on subtle notions such as similarity, relative importance of some data and the influence of the problem data over the solution [Py, 1994]: these are fundamental notions to be found in case-based reasoning likewise. In analogical reasoning, processes such as matching or retrieval considered as general cognitive processes applied to mental representations. By contrast, CBR systems are created in order to solve a specific task in a particular domain. Thus CBR differs from analogical reasoning because it applies to a particular domain while analogical reasoning considers transfers between several domains. Hence one of the assets of CBR is its ability to mobilise knowledge specific to the domain of application with the aim of improving the whole efficiency of the reasoning process.

One aspect that distinguishes CBR from other methodologies of the artificial intelligence domain is the fact that it is based on the reuse of experiences: solutions are reused rather than built up from theoretical knowledge. On this point, we may consider that this type of reasoning is closer to the way humans sometimes think in real life: we try to find an existing solution before building one of our own (via demonstrative reasoning, for instance). Case-based reasoning is particularly well adapted to the resolution of open problems for which the associated domain theory is weak or difficult to formalise; past cases bring specific knowledge to the solution of the problem. CBR is also an incremental and evolutive reasoning process since it acquires experience every time a new problem is solved. Contrary to other systems which, in order to solve a problem, must go through a huge research area, case-based reasoning systems are able to reason with a small number of experiences.

In the early 90s, CBR appeared as an interesting alternative to the rule-based systems which were beginning to show their limits. In rule-based systems, coherence of the base becomes more difficult as new rules are added to it. CBR adopts a contrasting approach: the addition of new cases to the base forms one of the very principles of this reasoning; the addition of knowledge therefore does not carry the risk of endangering the system. At that time, studies such as [Slade, 1991] showed how the CBR research paradigm could bring answers to some of the limitations of actual approaches of artificial intelligence. Since then, case-based reasoning has been used for various tasks (planning, synthesis, diagnosis, decision-making, etc.) in domains as varied as cookery, medicine, law, road safety, mechanical engineering, risk management, ecology, etc.

2.2 The CBR process

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? In order to bring answers to these questions, the first CBR studies focused on the identification of the knowledge being handled and the decomposition of reasoning into more specific steps, focusing on specific problems.

In [Richter, 1995], Richter describes four containers in which a CBR system can store knowledge. This knowledge may include domain knowledge as well as problem-solving knowledge which describes the "method of application" of the domain knowledge inside the container: vocabulary used to describe the domain, case base, similarity measure used for retrieval and solution transformation (used during the adaptation). Among these types of knowledge, the so-called principal elements are cases which represent experiences. Cases are generally composed of one problem part and one solution part. The problem part describes the problem context while the solution part describes the reasoning which led to the solving of the problem and gives the solution. The domain knowledge itself permits the modelisation of the theory available for the domain in question.

As for reasoning, it is often organised following a *cycle* which specifies the sequence of the various steps. A first version of this cycle was proposed in [Riesbeck and Schank, 1989], then a more elaborate version was described in [Aamodt and Plaza, 1994]. This cycle, composed of four main steps gravitating around a knowledge base (including cases), quickly became the reference cycle in the CBR domain. Little by little, the reasoning process filled out, new steps appeared and links between the different steps and the knowledge containers became more precise. The four diagrams presented below give an idea of the evolution of the representation of the case-based reasoning process.[2]

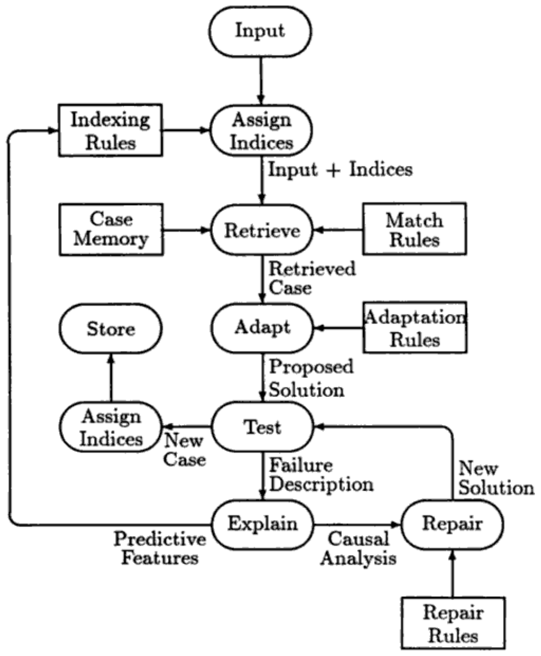


Figure 2.2: CBR flowchart according to Riesbeck and Schank - 1989.

The basic cycle of a case-based reasoner, according to Riesbeck and Schank, is "input a problem, find a relevant old solution, adapt it". Figure 2.2 shows the basic flow of control

[2]The diagrams are extracted from the original papers of the respective authors and are reproduced without modification (except for the last one).

described in *Inside Case-based Reasoning* [Riesbeck and Schank, 1989, p. 13]. A remarkable aspect of this flowchart is the test-explain-repair loop. The proposed solution is always tested. If it is satisfactory, it is indexed and stored for later retrieval. By contrast, if the test fails, two events occur:

- The indexing rules are updated (to avoid the re-occurrence of the failure),
- The solution is repaired, usually after an explanation of the causes of the failure, in order to provide a satisfactory solution to the input problem.

However, this diagram gives no details about who or what performs the solution repair. The authors assume that, depending on the domain, either the repair or the explanation has to come first. If the explanation comes first, then the repair can be guided by the explanation. In some situations, when the repair can be performed before the explanation. In such a situation, it is often useful to look at what has been repaired to find the correct explanation for the failure.

The role of explanations in CBR is still a major research issue that will not be discussed here. However, the interested reader could find additional information in [Bergmann et al., 1993], [Aamodt, 1994], [Massie et al., 2004], and [Sormo et al., 2005].

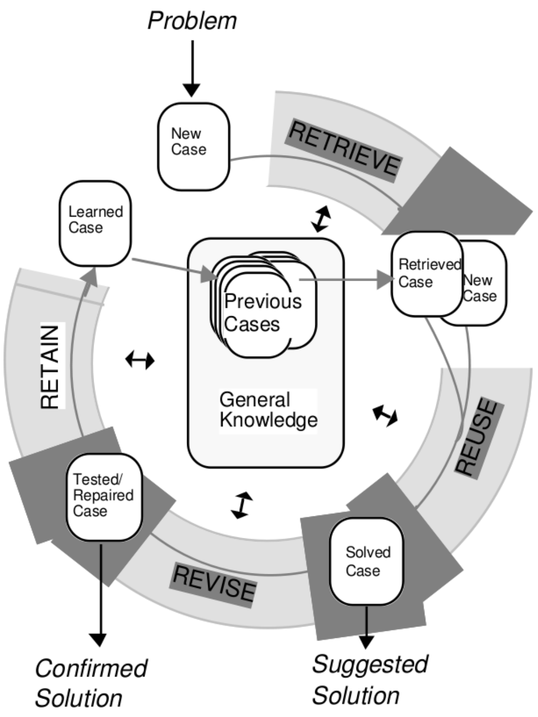


Figure 2.3: The CBR cycle according to Aamodt and Plaza - 1994.

Figure 2.3 presents the "classical" CBR cycle proposed by Aamodt and Plaza in 1994 [Aamodt and Plaza, 1994, p. 8]. The reasoning is broken down into four steps, known as the four "Re": Retrieve, Reuse, Revise and Retain. Each step of the cycle exploits the knowledge base central to the system, which contains previously solved cases as well as general knowledge. This cycle now serves as reference for most studies in this field.

In this cycle, a problem is represented by a case (*new case*). The retrieve step leads to the retrieval of a case found in the case base (*retrieved case*). During the reuse step, the solution of the retrieved case is transformed in order to propose a solution to the case under study. The *solved case* is then proposed as a solution to the problem. The revision step allows one to judge whether or not the solution is satisfactory and to amend it if need be. This is the *test/repaired* case which is considered as an acceptable solution to the problem. Finally, the retain step permits the addition of the *learned case* (i.e., the solved case) to the case base.

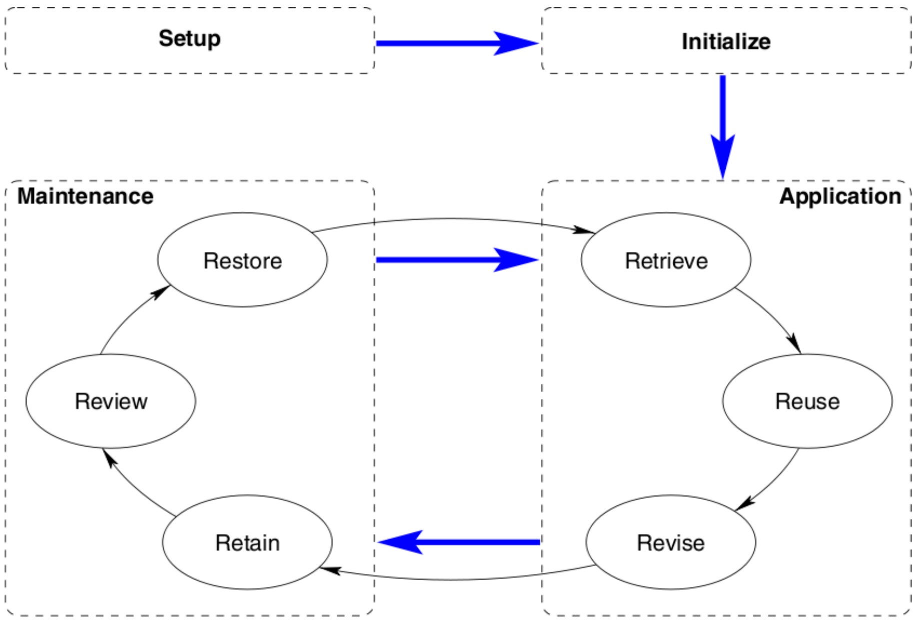


Figure 2.4: The CBR cycle according to Iglezakis et al. - 2004.

Figure 2.4 shows an extended variation of the classic four step cycle proposed in [Iglezakis et al., 2004, p. 4]. This diagram illustrates the importance of the knowledge maintenance phase in CBR systems. An interesting point of this approach is that the maintenance phase is distinct from the application phase. Nevertheless, the authors of this cycle defend the idea that the success of maintenance does not depend solely on maintenance strategies, but also on the way the steps preparatory to the maintenance are integrated into the main cycle. Another interesting aspect illustrated by this cycle concerns the knowledge which must be maintained. Indeed, even if a large number of studies focus on case retention and deletion strategies, maintenance must also concern the other types of knowledge handled by the reasoning.

Figure 2.5 presents another immediate evolution of the cycle proposed by Aamodt and Plaza. Two differences appear: the first one concerns the addition of the elaborate step to the traditional cycle. This step allows the system to evolve from a badly formulated problem to a problem which is “intelligible” by the system. The second evolution concerns the explicit representation of the expert in the centre of the cycle. This cycle illustrates the requirement for interaction between the system and the “outside world” to permit the acquisition of knowledge *during* the problem-solving process. Interactions may take different forms and

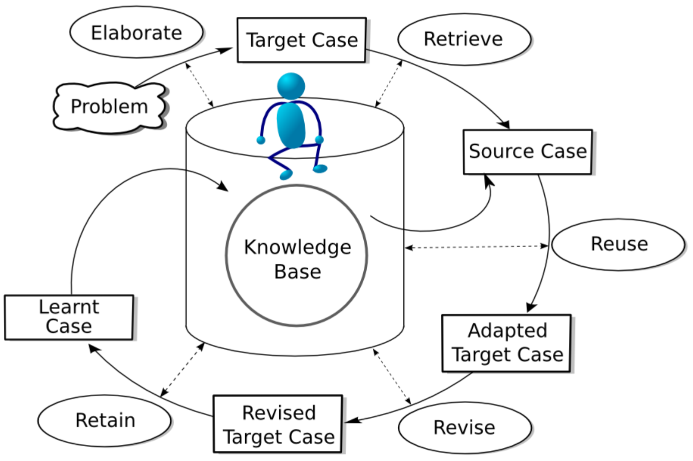


Figure 2.5: A CBR cycle centred on interactions according to Cordier et al. - 2006.

the "interlocutors" may vary: user, expert, "oracle", other system, etc. This cycle, initially proposed in [Cordier et al., 2006c], is characteristic of the way CBR systems are considered in this thesis:

- CBR systems are knowledge-based systems and the problematics of knowledge engineering of such systems must be one of the main concerns.
- Interactions must allow the system and its users to evolve together when solving problems.

The rest of this section mentions only briefly the studies linked to the various phases of reasoning presented above. The CBR paradigm has been the subject of many syntheses and the reader will be able to find additional information in work of reference such as [Riesbeck and Schank, 1989], [Aamodt and Plaza, 1994], [Watson and Marir, 1994] [López de Mántaras et al., 2005], and [Richter and Aamodt, 2005].

2.2.1 Elaboration

The object of a reasoning CBR session is to solve a *problem* expressed by a system user. In order to optimise the chances of success of the system, one must ensure that the problem is described in the best possible way, that is to say, so that it can be correctly handled by the system. The role of the *elaboration* step therefore consists in setting up the specification of the problem to be solved, called *target problem*. To that end, various mechanisms are set in motion to move from an often ill-specified problem to a correctly defined problem. Most of the time, when there is no elaboration, the system tries to infer the knowledge missing from the description of the problem which makes the search for similar cases more difficult.

Although it does not appear in the traditional CBR cycle, the elaboration step is essential and, even if it is rarely claimed as such, it exists in several CBR systems and particularly in

CHEF , a system for planning cooking recipes [Hammond, 1986] and in CARMA , a consultative system for the study of grasshoppers [Branting et al., 2001]. In CARMA, one of the main tasks is in fact to define the relevant characteristics of a problem which are essential to the reasoning process.

An initial formalisation of the elaboration step was recently proposed in [Fuchs et al., 2006b]. This study underlines the importance of elaboration in CBR, which had already been highlighted in the study by Herbeaux on ACCELERE [Herbeaux, 2000]. Elaboration is important because it permits the evaluation of present adaptation capabilities and the identification of missing knowledge, thereby ensuring a greater success during the adaptation step.

2.2.2 Retrieval

During the *retrieval* step, the system searches the case base (which contains all the solved problems) for a case which is *deemed similar* to the target problem outlined during the elaboration step. The retrieved case, called the *source case*, is then used as a basis for the construction of the solution to the target problem. The whole difficulty of the retrieval step rests in the choice of a good selection criterion for the case to be retrieved. The work described in [Lieber and Napoli, 1998] deals with the issues of correctness and completeness of case retrieval in CBR. The most traditional approach consists in using a *similarity measure*, but other techniques involving the organisation of the case base are often used to improve the quality of retrieval.

The notion of similarity measure has been widely studied and discussed in the field of CBR. Determining the similarity between two cases is far from being a trivial task, and the choice of a good similarity measure is crucial to the efficacy of the system [Bisson, 1994]. The most common similarity measures are so-called *surface* similarities which compare the description of the problems after a matching of the relevant descriptors. Beyond the CBR field itself, generic similarity measures have been proposed [Tversky, 1977], and a study has been made in [Rifqi, 1996]. Surface similarities are relevant when descriptions of the cases are represented by groups of attribute-value pairs, but become difficult to use in a relevant manner when case representation is more complex. In situations such as these, it becomes necessary to use structural similarity measures like those defined by [Gentner and Forbus, 1991], for example. The drawback of similarity measures based only on surface criteria is that they do not take into account case adaptability. They can therefore lead to the retrieval of cases which may appear similar but which are not, with regard to the adaptation problem. This is why some similarity measures take into account the criterion of case adaptability. Smyth and Keane, for example, consider that a case similar to a target must be a case which can be adapted in order to find a solution to this problem [Smyth and Keane, 1998]. They then propose an adaptability measure of the retrieved case used during the retrieval step to select the "best case" and described the adaptation-guided retrieval (AGR) approach. On the same lines, Leake describes a measure of the adaptation cost of a case with respect to a given problem [Leake et al., 1997]. Most of the time, similarity measures are encoded in the system during the design phase but some approaches such as the one developed in the PATDEX system offers mechanisms for classification and learning of similarity measures [Richter, 1992]. Such approaches enable the system to update the similarity measures it relies on progressively.

The organisation of the case base also plays an important part in the retrieval of a good source case. There are numerous studies on the organisation of the memory. Malek, for example, suggests the use of a neural network with incremental learning to organise the case base [Malek, 1996]. The memory is then divided into two levels: prototypes and indexes. Prototypes are general descriptions which include the common characteristics of a group of cases. As for indexes, they enable the organisation of prototypes and the quick retrieval of relevant cases. In [Arcos and De Mantaras, 1997], the authors use the notion of perspective for the retrieval and bring in the notion of points of view on retrieved cases. Points of view allow the system to retrieve relevant cases with regard to a sub-group of criteria defined for the target problem. In [Lenz and Burkhard, 1996], the authors present CRN (Case Retrieval Nets), a memory model in the form of a network organised for easy retrieval of similar cases. In this model case attributes are represented by nodes in the network and weightings corresponding to similarities are shown on the edges. Finally, in [McSherry, 2003], the authors introduce the notion of coverage of the solution space by a retrieved case and use this notion to assess the quality of retrieval.

At the present time, the retrieval step is probably the best formalised step of the CBR process. It benefits from all the studies on similarity measures which can be based on solid mathematical and logical foundations. However, in current studies, case similarity is too often considered independently from the adaptation problem and, when the adaptation fails, the knowledge used for the evaluation of similarity is rarely questioned. In fact, an adaptation failure should automatically lead to a revision of the knowledge involved in the reasoning process and this implies therefore the knowledge used during case retrieval. The adaptation-guided retrieval approach proposed in [Smyth and Keane, 1998] is interesting from this point of view: it shows that the knowledge used for the evaluation of similarity and that used for adaptation are linked and, hence, editing adaptation knowledge has an immediate impact on similarity knowledge. We shall demonstrate later that the studies presented in this thesis are partly based on these remarks.

2.2.3 Adaptation

The *adaptation* step is probably the most delicate and difficult step in CBR (see, for instance, [Smyth and Cunningham, 1993]). During the adaptation step, the case solution is modified in order to solve the problem under study. In some domains (such as design), the adaptation step is necessary since it is extremely rare that the solution of a known problem is directly applicable to the problem under study. This step thus transforms the solution by applying knowledge linked to the domain and the context of the new case. Generally, transformations reflect differences observed between the target problem and the retrieved source case. Adaptation thus plays two parts: detecting necessary changes and then applying them. The complexity of adaptation is variable: it may be a question of changing values in the solution (adaptation by substitution), of changing part of the solution (transformational adaptation) or even totally re-building the solution using additional knowledge.

The problems of adaptation have been studied by CBR pioneers whose approach was then memory-oriented. A transformational approach of adaptation was proposed by Carbonell in 1984 [Carbonell, 1984]. It consists in performing a transformation of the solution of the retrieved problem in order to adapt it into a solution to the problem being solved. This approach is based on the notion of adaptation paths. It presupposes that the representation of the solutions is sufficiently flexible to enable them to be transformed. It also requires knowledge of operators and of the conditions of application of the operators enabling the transformation. Carbonell then proposes a derivational approach [Carbonell, 1986, Carbonell and Veloso, 1988] which consists mainly in keeping track of the decisions taken to apply them to new problems. When transferred to the CBR domain, this approach results in applying the reasoning process used to build the source solution so as to find the target solution starting from the target problem. This presupposes the availability of a representation of this reasoning process, as well as the ability to adapt it to the target. Whereas the derivational adaptation requires a solid representation of the domain knowledge, transformational adaptation only requires knowledge of the differences between problems and the corresponding differences between solutions. Therefore transformational adaptation does not place any constraints on the completeness of transformation knowledge; however, the greater the availability of correct knowledge, the greater the chances of success of the adaptation. Due to these low knowledge requirements, transformational adaptation is the most typical adaptation approach in CBR, particularly in the fields where it is difficult to acquire a complete model of the domain. In 1993, Kolodner proposed a synthesis of the various proposals for transformational and derivational adaptations [Kolodner, 1993].

Adaptation is particularly important in problem-solving systems dealing with planning or design. Present adaptation methods possess variable complexity according to the way they alter the solution: substitution adaptation only alters certain parts of the retrieved solution whereas transformational adaptation modifies the structure of the solution. Examples of adaptation strategy are fairly numerous. In CHEF [Hammond, 1986], adaptation by substitution is used to replace ingredients in order to satisfy the demands of the menu. CHEF also uses transformation adaptation to modify the stages of the recipe. DÉJÀ VU [Smyth and Keane, 1995c] uses adaptation strategies and components called adaptation specialists. Adaptation specialists are used to perform localised changes to solutions but are blind to the changes made by other specialists. In consequence, adaptation strategies are designed to avoid conflicts between specialists. Adaptation specialists and adaptation strategies are described and illustrated in [Smyth, 1996a]. Model-based adaptation is another type of transformation adaptation which includes causal reasoning. This approach is illustrated in systems such as CASEY [Koton, 1988] and KRITIK [Goel and Chandrasekaran, 1989]. Generative adaptation approaches vary from substitution and transformation: they generate the target solution by reusing methods used to produce the retrieved solution. An illustration of this approach is shown in PRODIGY/ANALOGY [Veloso et al., 1995].

Recent studies aim to formalise the adaptation step: in [Fuchs et al., 2000], the authors proposed an adaptation approach based on the notion of influence of the problem on the solution which, combined with the matching made during the retrieval, permits the adaptation of the target case. This approach is rather new inasmuch as, even if the knowledge used in retrieval and adaptation have long been considered as distinct from each other, they seem here to be dual if not identical.

Apart from that, some studies are concerned more particularly with adaptation knowledge and not only with the process. A general study of techniques and knowledge used during adaptation is proposed in [Wilke and Bergmann, 1998]. In [Hanney et al., 1995], an adaptation-oriented taxonomy of CBR system is proposed. This taxonomy is established according to the part played by knowledge in the adaptation process, i.e. at various steps of the cycle: target elaboration operators, role substitution operators, sub-goaling operators and goal interaction operators. The representation of adaptation knowledge in the form of adaptation case has been studied, among others, in [Kinley, 2001], in the field of the DIAL system. The recourse to adaptation cases allows the exploitation of principles of case-based reasoning in order to acquire new adaptation knowledge. It is also a means of making up for the lack of formalisation of the knowledge the system is trying to acquire. In his studies, Kinley also shows that the acquisition of adaptation knowledge has an impact on the other types of knowledge in the system (this aspect will be discussed in the following chapter). In the same vein, in [Smyth and Keane, 1996], the authors show how the adaptation knowledge can be used to retrieve (and adapt) design cases.

Although it is at the heart of CBR, adaptation is often considered as the most difficult step. Because of its complexity, it is often performed manually or by applying *ad-hoc* methods based on knowledge which has been pre-inscribed in the system. Now, case-based reasoning systems often have to treat cases which could not have been anticipated and for which adaptation knowledge has not been coded *a priori*. Consequently, methods permitting the acquisition of adaptation knowledge throughout the life of a system are of vital importance. These approaches will be studied in chapter 3 of this manuscript.

2.2.4 Test, explain, repair

The *test, explain, repair* phase, often called *revise* step, enables us to repair the problems identified in the adapted solution. There are several documented approaches with regard to this step. In systems such as CHEF or DIAL, knowledge allows us to anticipate some failures. To this end, the system makes sure that all expressed targets are reached in the proposed solution. If not, the solution is repaired to avoid foreseeable failures. Once it has been edited, the solution is proposed to the user.

Other systems exploit the interaction with the user to assess the quality of the proposed solution and eventually to edit it. In these systems, the solution resulting from adaptation is first studied by the user (who can, for example, confront it to the real world). He can then accept or refuse the solution, or even edit it if he has the ability. In these situations, revision is very important because the feedback from the user enables the assessment of the quality of adaption performed by the system as well as the usefulness of the newly solved case. As a general rule, a negative feedback from the user may highlight a failure in the reasoning process, i.e., during adaptation or, in some cases, in retrieval or elaboration.

Proportionately, few studies are centred on the revise step, though it allows the system to acquire easily additional knowledge. For example, in [Aamodt, 1991] or in [Fox and Leake, 1994b] and [Fox and Leake, 1994a], missing retrieval and adaptation knowledge is identified and acquired during the revise step. In [Smyth and Keane, 1995a], this step is used to evaluate the case usefulness and to define a retention or forgetting strategy according to its contribution to the system's general efficacy.

The revise step must not be used only to repair the solution issuing from the adaptation, it must also permit the acquisition of knowledge enabling the repair. When the revision is done punctually at the end of the reasoning process (i.e., when revision only implies an analysis of the final solution), it is not easy to acquire this additional knowledge. In the studies mentioned here, revision is omnipresent throughout the cycle: here the successive repairs to the intermediary solutions permit the progressive acquisition of new knowledge through the interaction between the system and its environment.[3] At the end of the process, the solution has been constructed jointly by the system and its user and the knowledge required for this construction has been learned by the system.

2.2.5 Retain

During the *retain* step, sometimes called learning step, the result of the problem-solving process is added to the system's general knowledge. The most naive approach to retention consists in storing in the case base successfully solved problems (in the form of cases). Other approaches retrieve solved cases with additional information indicating to which point the proposed solution was satisfactory for the problem or whether it answered certain objectives.

The problem of maintenance of the case base is also linked to the retrieval step and to the tradeoff "performance vs. efficiency". Indeed, as the number of cases added to the case base increases, so does the efficiency of the system. But the increase of the size of the case base also involves an increase in the time spent searching for cases. The performance of the system diminishes as its efficiency increases. This raises the problem of case usefulness [Minton, 1990]: how to maintain a reasonably sized case-base without altering the system's performance? Leake and Wilson highlight the necessity of considering the system's efficiency and performance together to study the optimisation of a case base. They also characterise in a general manner the process to be used to carry out this maintenance [Leake and Wilson, 2000]. Smyth and Keane [Smyth and Keane, 1995b] propose a case efficiency model. They set out four different categories of cases and classify the cases in these categories according to the competence they bring to the system. The deletion of cases is guided by competence criteria. Thus, when a case is deleted from the system according to a given deletion policy, the system's efficiency is not altered. This approach is a good way to keep a reasonably sized case base without risking a decrease of the system's problem-solving ability. Other studies use an introspective reasoning to carry out the maintenance of the case base: they try to improve case indexing by examining retrieval failures [Fox, 1995], [Cox and Ram, 1999].

In this way, most of the research related to the retrieval step is centred on learning from past solved cases, indexing methods and case base organisation, but there are few which focus on learning implicit knowledge such as similarity or adaptation knowledge [Aamodt, 1991]. In fact, cases are not the only knowledge units involved in case based reasoning. This type of reasoning is based on a domain theory as well as a quantity of similarity and adaptation knowledge. Studies focusing on these aspect will be discussed in the following chapter.

2.3 A CBR system is a knowledge-based system

The decomposition of the CBR cycle into steps enables us to grasp the aims of each phase and their sequence. This decomposition may give rise to the notion that knowledge containers used in each steps are distinct and independent, but in fact, they are closely linked. This is why, when one is interested in a knowledge-based system such as a CBR system, one must consider the whole process and adopt a unified vision of the knowledge it involves.

[3]The environment of the system includes everyone and everything the system is able to interact with, i.e., the user, the expert, another system, etc.

2.3.1 The knowledge of CBR

Reflections on the characteristics of CBR knowledge give rise to several questions:

- What knowledge is involved in each step?
- What role does the knowledge play?
- Where does it come from?
- How can it be represented?

Most of the studies seeking to answer these questions do so by positioning themselves with regard to knowledge containers as defined by Richter [Richter, 1995]. However, as mentioned previously, pieces of knowledge are in fact closely linked and should not be considered separately. For example, can we consider that adaptation knowledge, often used at the retrieve step, is distinct from similarity knowledge?

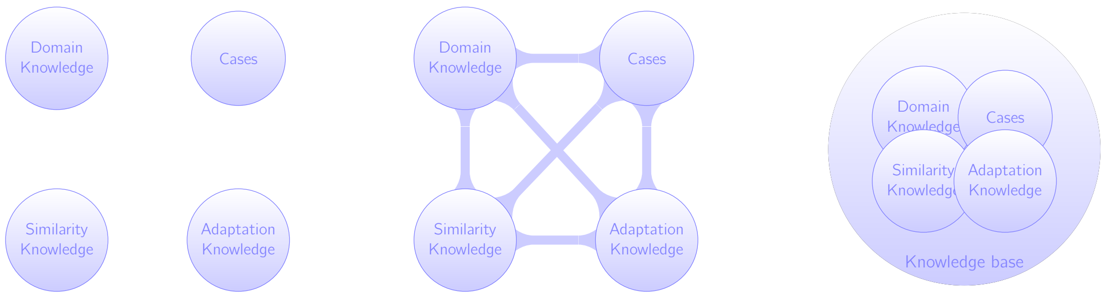
Table 2.1 brings together a few remarks about the knowledge involved in a CBR process. Remarks are ordered according to Richter's knowledge containers (first column). The second column summarises the main forms that the knowledge can take. The third column brings precisions on the role played by the different knowledge units and emphasises the steps on which the knowledge is used (in bold).

Knowledge containers	Knowledge units	Knowledge roles
Vocabulary used to describe the domain	Ontologies, rules, etc.	Guidance of **elaboration**, Control of the inferences during **retrieval** and **adaptation**, support of the **memorisation**
Case base	Vectors of attribute-value pairs, Structured representations, Textual Cases, etc.	Support of the reasoning process during **all the steps**
Similarity measure (similarity knowledge)	Similarity metrics, indexes, etc.	**Retrieve** a new case
Solution transformation (adaptation knowledge)	Adaptation rules, adaptation operators, adaptation cases, etc.	Support of the **elaboration**, guidance of the **retrieval**, realisation of the **adaptation**

Table 2.1: Towards a typology of CBR knowledge.

This table allows us to observe that there is no obvious correlation between the knowledge containers and the reasoning step in which the knowledge is used. One could remark that several types of knowledge can collaborate to accomplish a common task. Hence, knowledge containers are tightly interconnected and one could easily imagine that a modification on a knowledge container may have an influence on the others. As an illustration of this remark, the following paragraph goes back on the links between similarity and adaptation knowledge.

In this study, Richter argues that theoretically, each knowledge container can contain all the required knowledge, as in "pure CBR" where all the knowledge is contained in the cases. In practice, it is very useful to rely on knowledge containers to "organise" the knowledge but this organisation should not impact the way knowledge acquisition is performed. Figure 2.6 summarises the evolution of this idea.



On the left, the knowledge containers of the CBR such as defined by Richter. In the middle, the four containers are interconnected. On the right, the containers allows the organisation of the knowledge units but are considered together as a single knowledge base.

Figure 2.6: The knowledge of the CBR.

Duality of similarity and adaptation knowledge

The relation between similarity knowledge and adaptation knowledge constitutes an interesting topic of study. Indeed, these two types of knowledge which had long been considered as separate, are in fact very close and several approaches and applications tend to unify them or at least to link them tightly.

In [Smyth, 1996b], the author proposes a retrieval approach guided by adaptability. The argument is that the source cases most similar to the target case are not always the easiest to adapt, particularly when the similarity measure is based on surface characteristics. Retrieval must therefore search not only for similar cases but above all easily adaptable cases. On the same lines, Leake suggests that a good case retrieval facilitates the adaptation effort [Leake et al., 1997]. Indeed, traditional similarity measures may lead to mediocre results inasmuch as they sometimes enable one to retrieve source cases very similar to the target case, but are difficult or even impossible to adapt. This observation shows the limit of similarity measures with regard to the total reasoning process (when similarity measures do not take into account adaptation knowledge). Leake proposes therefore to include in the similarity measure a notion of "adaptation cost" to make it more relevant. Thus, in this approach, the evaluation of the similarity between the target case and the various cases in the case base is done in two steps: first, a classic similarity measure is made by comparing case descriptors, then the most similar cases retrieved at the end of the first step are prioritised according to their adaptability. A formal definition of the adaptation-dependent retrieval problem has been proposed in [Avesani and Blanzieri, 1999]. The main idea of this work is to define a metric that can be used by an adaptation-driven retrieval process. In order to define this metric, the authors propose a formalisation of an adaptability topology, allowing us to induce an adaptability metric.

For his part, Lieber proposes an adaptation approach through the use of similarity paths. Behind the notion of similarity paths lies the idea of splitting the adaptation into simpler adaptation sub-tasks. To highlight the similarities between two complex problems, one must possess knowledge of the domain. The approach proposed in [Lieber, 1999] aims to decrease the difficulty of adaptation by increasing the similarity of the problems, via the reformulation of a complex problem into several intermediary problems corresponding to simpler adaptations.

2.3.2 Formalisations of the CBR process at the knowledge level

Several studies have analysed the knowledge aspect of CBR, some by considering the whole process, others focusing of one particular aspect of it. In [Armengol and Plaza, 1994b], the authors analyse the knowledge aspect of the CHEF system (according to Newell's original definition [Newell, 1982]) following the Commet methodology [Steels, 1992]. The Commet methodology defines several components:

- An ontology of tasks (sets of goals that have to be solved)
- A set of models (relevant knowledge required to achieve a task)
- A set of methods (procedures organising and executing activities).

The analysis at the knowledge level permits explicit relations to be made between tasks and in particular between learning and problem-solving. Thus, the knowledge acquisition process can benefit from the formalisation at the knowledge level and such an analysis can be a first step towards the integration of different learning methods. In [Armengol and Plaza, 1994a], the authors conduct the same analysis on three systems: CHEF, PROTOS and CASEY.

In [Fuchs and Mille, 2000], a modelisation framework is also proposed. In this framework, the whole of the knowledge involved is broken up into knowledge models. A knowledge model links descriptions of concepts grouped according to the part they play in the problem-solving process. The main models are as follows:

- The domain's conceptual model, which describes the concepts used in an ontology of the domain;
- The case model, which describes what constitutes the experiences and particularly the problem, the solution and the reasoning path which led to the solution;
- The reasoning tasks models, which describe the CBR steps and allow each task to be broken down.
- Model supports which specify the knowledge required to realise the inferences on each reasoning task.

In [Fuchs and Mille, 1999], the same analysis is detailed for the adaptation step. In [Salotti and Ventos, 1998], the authors use a particular descriptive logic (C-CLASSIC) to model two special phases of the CBR process: retrieval and learning. Using this formalisation, the case search can be made through automatic concept classification and indexes taxonomy: subsumption calculations between concepts allow the introduction of partial order of cases and the definition of similarities (and dissimilarities) between cases. Indexes taxonomy is improved during the learning step. On the same lines, a modelisation of the CBR cycle via description logic is proposed in [Gómez-Albarrán et al., 1999], [Gonzalez-Calero et al., 1999]. This model proposes the structuring of all the knowledge required for reasoning, a structured case representation and inference mechanisms based on description logic in order to perform reasoning tasks.

Some studies focus on adaptation in particular. In [Bergmann and Wilke, 1998], the authors propose a formal model of transformational adaptation to "take a step" towards a more general formalisation of CBR adaptation. This model is based on the notion of quality (in the general sense, since quality may include various notions such as usefulness or correction) of the solution with regard to a given problem. This notion of quality permits the formalisation of adaptation knowledge in the form of adaptation operators. Adaptation operators are case transformation functions. They stretch the traditional concept of transformational adaptation, since they add to the classic notion the concept of quality of solution. In [Fuchs et al., 1999], the authors propose combining the retrieval and adaptation steps into a single planning process. The problem of case adaptation is then brought back to a problem of plan adaptation and this process may be accomplished through successive reformulations organised along similarity paths. Very recently, in [Sanchez-Ruiz et al., 2008], the authors propose a quite similar approach applied to case-based training system that generates a 3D scenario from a declarative description of the training case (which is used as a domain example though the approach is said to be domain-independent). An interesting part of this work is the use they made of description logics in planning problems. Though no explicit link is made with the work presented in [Fuchs et al., 1999], one can observe a lot of common ideas and, in our opinion, and interesting prospect could be to bring together these two approaches.

In [Fuchs et al., 2000], a formalisation of the differential adaptation strategy is introduced. Differential adaptation strategy relies on variation between problems and on relations between a problem part and a solution part (called dependencies) to perform adaptation. More information about differential adaptation can be found in [Fuchs et al., 2006a]. Moreover, in this work, we have made an implementation of the differential adaptation strategy in the prototype called IAKA-NF. Another formalisation of adaptation, conservative adaptation, is proposed in [Lieber, 2006]. This adaptation approach is linked to the revision theory. It consists in "keeping as much as possible" elements of the solution to be adapted while ensuring that it remains consistent with the domain knowledge. Conservative adaptation is therefore formalised via an operator, called revision operator. In this work, we have used the conservative adaptation strategy in the FRAKAS-PL application.

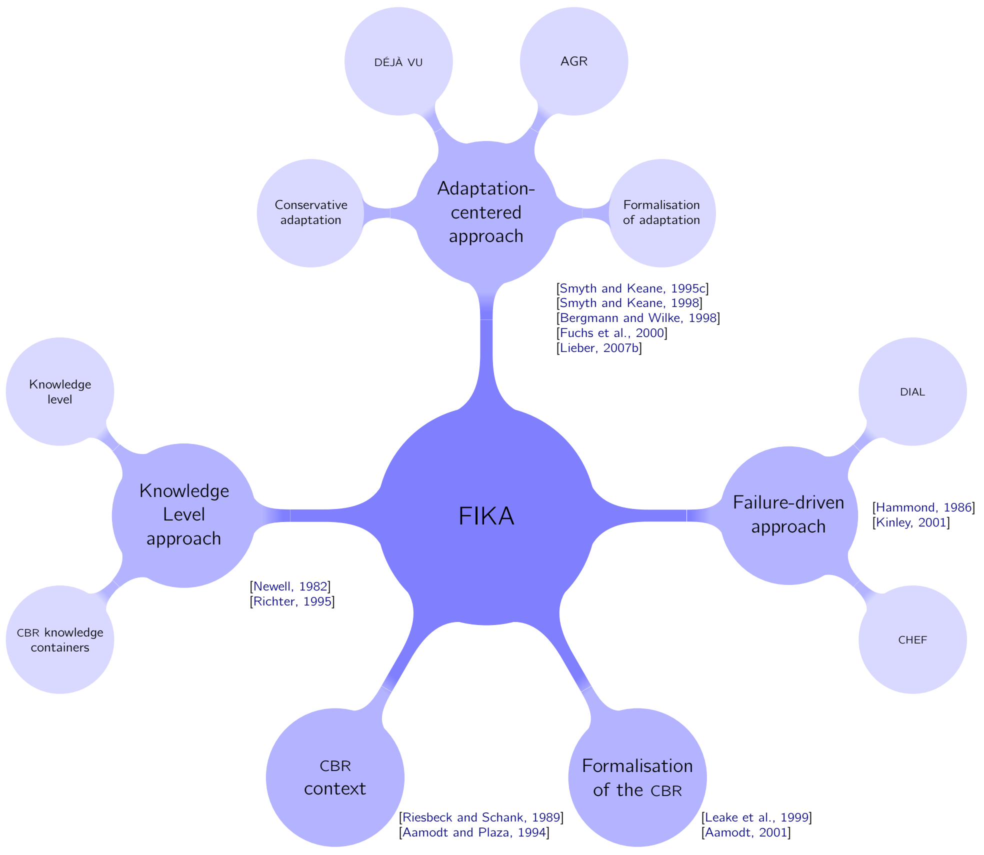
The need to consider the knowledge pieces in reference with the steps they are used in has been underlined in several studies. For example, in [Leake et al., 1999], it is argued that integration between knowledge management system and tasks that they serve should be much stronger and that systems should be able to learn unobtrusively by monitoring the user's tasks. Targeting the knowledge acquisition community, a review of methodologies and tools for analysis, modelling, and maintenance of knowledge components of CBR has been proposed in [Aamodt, 2001]. This review argues, among other things, that the knowledge level is the appropriate level for describing the behaviour of CBR systems and to identify the contents of its knowledge components.

CBR analysis at the knowledge level offers a number of advantages. First of all, the various studies show that it allows for a better understanding of the problematics and targets of the different phases of the reasoning process, and, consequently, to facilitate their integration. Proposing CBR models at the knowledge level facilitates the conception of systems. The learning process also benefits from the attempts to formalise reasoning at the knowledge level.

2.4 Synthesis

The background of this work is the CBR context. The overview of related work presented in this chapter provides us with some concepts that can ground our knowledge acquisition approach. These concepts are summarised in diagram 2.7. For each concept, a non-exhaustive list of relevant references is given. Several systems implementing these concepts are also referenced in the figure.

It must be remarked that the studies referring directly to the main problematics of this thesis, the acquisition of knowledge in systems, are not mentioned. They are the subject of a specific study in chapter 3.



The FIKA approach described in this thesis is related to several researches. The main approaches that are at the roots of FIKA or that can be compared to FIKA are illustrated in this figure.

Figure 2.7: A synthesis of the state of the art.

2.5 Conclusion

This chapter has presented a certain number of reference studies in the CBR domain. The state of the art shows that this reasoning paradigm, which has been in existence for the past twenty years, is based on well controlled mechanisms, even if it still is the subject of interesting researches. CBR systems are now used in numerous domains with varied applications (diagnosis, decision-support, planning, training, etc.). These systems have excellent results and keep improving through use (because they retain cases) and demonstrate the efficiency of CBR in concrete applications. However, the systems which actually "learn" to solve problems are rare.

The study of existing researches has also shown the important part played by additional knowledge in a CBR system. This knowledge is essential to remember past experiences or cases and to adapt them to the current situation. This knowledge gives the systems their ability to solve problems and hence they emphasise the difference between a good retrieval system and one that actually solves problems. Consequently, in order for a system to be able to solve a problem, it must have the ability to learn this knowledge.

FIKA, an approach for interactive knowledge acquisition based on reasoning failures, finds its justification in the conclusions of the analysis presented in this chapter. The following chapter is dedicated to its presentation. In a first part, a narrower review of the knowledge acquisition methods and tools is presented. Then, the chapter offers a description of the definitions and principles of FIKA.

Good tests kill flawed theories; we remain alive to guess again.

Karl Popper

3 FIKA: Failure-driven Interactive Knowledge Acquisition

FIKA defines a general approach for interactive and opportunistic acquisition of knowledge in case-based reasoning. This approach is at the heart of our work. In this chapter, we go one step further in the analysis of the related work started in the previous chapter by focusing on the specific issue of knowledge acquisition in the CBR context. Based on this study, we describe the benefits of interactive knowledge acquisition approaches to acquire knowledge related to experience and to "know-how". The chapter also gives a set of definitions that are frequently used in the remainder of this document. Last, we briefly discuss of different operationalisations of the FIKA approach we have worked on. These approaches are described at a very general level and are compared with regard to several criteria. The chapter ends with a synthesis of the main characteristics of the FIKA approach. This synthesis is used as a guideline for the studies conducted in the three following chapters.

Related publications:
[Cordier, 2004], [Cordier and Fuchs, 2005], [Cordier et al., 2006c], [Cordier et al., 2007d], [Cordier et al., 2008b]

Main related work:
[Hammond, 1989], [Leake, 1996]

Keywords:
Interactive and opportunistic knowledge acquisition, on-line, user-centred

Contents

Contributions of this chapter

- An approach for interactive and opportunistic knowledge acquisition in CBR
- A comparison of this approach with other knowledge acquisition strategies
- A discussion on the different operationalisations of FIKA

In order to solve a problem, a knowledge-based system performs a generic reasoning (logical, analogical, etc.) exploiting knowledge that is represented in a formalism adapted to the reasoning requirements. A system must be provided with knowledge acquisition methods allowing it to acquire and model this knowledge. Case-based Reasoning, as a reasoning paradigm, benefits from well-known knowledge acquisition methods. In the previous chapter, we have observed that CBR was often considered as a way to facilitate the knowledge acquisition process because of the ease of storage of solved cases in the case base. Indeed, by accumulating cases, a CBR system progressively increases its *experience base* and thus its ability to solve more problems. But we have shown that case retention is not sufficient to acquire all the knowledge a system needs to reason on cases. In particular, when domain knowledge, similarity knowledge and adaptation knowledge cannot be captured in cases, they must be acquired by other means. The definition of knowledge acquisition methods suitable to these different forms of knowledge is an important knowledge engineering issue and a major research field in CBR. The aim of the work described in this document is to address this issue by providing an approach adapted to the experimental nature of the knowledge to be acquired. This approach, called FIKA for *Failure-driven Interactive Knowledge Acquisition* is described in this chapter.

Section 3.1 deals with knowledge acquisition issues in CBR. Then, a review of the related work is performed and the FIKA approach is compared to some of these researches. Section 3.2 is dedicated to the presentation of FIKA and to the definition of the notions used in the remainder of this document. Finally, section 3.3 describes how FIKA can be operationalised in concrete applications, and introduces IAKA and FRAKAS, two models based on the FIKA principles.

3.1 Knowledge acquisition issues in CBR

This section describes researches related to FIKA from a knowledge engineering point of view. The first part of the section describes the questions that have guided the analysis of the related work, then several approaches are compared following a set of selected criteria. Finally, FIKA is briefly introduced and compared to these approaches (a detailed description of the principles of FIKA is given in section 3.2).

3.1.1 Knowledge engineering in CBR

Several aspects are to be considered when examining the question of knowledge engineering. Some of these aspects, relevant to our study, are detailed hereafter.

Type of knowledge to be acquired. The typological study of CBR knowledge presented in chapter 2 has shown that systems are based on knowledge which is represented in such a way as to be easily used by reasoning engines. For practical reasons, this knowledge is also organised into knowledge containers: cases, domain knowledge, similarity knowledge and adaptation knowledge. These various knowledge units constitute acquisition targets. This knowledge is of a varied nature and thus requires different acquisition methods. It is therefore necessary to take into account the nature of the knowledge one seeks to acquire in order to select the appropriate method.

Knowledge representation. Knowledge is represented in different forms in different systems: cases (which may be structured in different ways), indexes, rules, ontologies, etc. These different structures imply different representation formats within the systems and hence different parameters to be taken into account by the acquisition method.

Knowledge origin. Several sources of knowledge can be examined. Knowledge which has already been acquired and modelled into the system can be manipulated by algorithms in order to cause new knowledge to "emerge", often in another form; this approach is called "knowledge light". On the contrary, other approaches try to acquire knowledge which is available within the system's environment, and in particular, the expert's knowledge. In so-called "knowledge intensive" systems, a lot of background and problem-solving knowledge is required to conduct the reasoning process. The approach we present here is a knowledge intensive approach.

Time of acquisition. The period of knowledge acquisition is an important criterium to be taken into account. Some knowledge can be acquired at the time of creation of the system, whereas other knowledge is easier to obtain "in context" while the system is being used. Generally, this knowledge acquired "as you go" has specific properties which make it difficult to formalise initially: linked to experience, constant evolution, particular situation, etc. Similarity or adaptation knowledge, for example, is often only valid in precise situations and cannot always be obtained by automatic processes. In the same way, in domains where theory is difficult to formalise, experience comes before theoretical knowledge when it comes to solving a problem. In these situations, it is not possible to acquire knowledge *a priori*, but only through experience. In addition, the maintenance phase can also constitute a knowledge acquisition period which may be appropriate in certain situations. According to the acquisition period under consideration, various acquisition approaches are possible. For example, when acquisition is made during the system design, the expert and the knowledge engineer can process, in one go, a large amount of knowledge. On the opposite, if the acquisition takes place while the system is being used, it is inconceivable to interrupt the expert's work and ask him to process a large amount of knowledge. It is nevertheless possible to ask him to validate a single piece of knowledge which might be acquired in this particular context.

Acquisition method. Among the acquisition methods which may be used, one differentiates between automatic and manual methods. Automatic methods apply algorithms (such as knowledge extraction from data) on important volumes of knowledge in order to extract new knowledge. These methods present the advantage of producing a large amount of knowledge in one go. This characteristic is an advantage inasmuch as the process is automatic, but it is also an inconvenient because the task of analysing the knowledge produced may be tedious for the expert. Manual acquisition methods aim to acquire and model into the system the knowledge of a domain expert. As a general rule, these methods are not based on knowledge already available in the system. Knowledge is modelled piece by piece into the system by the knowledge engineer by consultation with experts and the use of specific tools. The interactive acquisition methods, such as FIKA, lie half-way between manual and automatic methods. They enable the acquisition of single pieces of knowledge, in context, but are based on existing knowledge and a computer tool adapted to facilitate the acquisition process.

All these questions which arise when studying the issue of knowledge acquisition in case-based reasoning are linked together. The chosen acquisition method depends on the type of knowledge to be acquired, the acquisition period depends on the method, etc. The analysis of the studies linked to FIKA was conducted by taking into account the whole of these study criteria.

3.1.2 An overview of the related work

In this section, we describe several researches on knowledge acquisition (other than cases) in CBR. For the sake of simplicity, we have decided to adopt a synthetic presentation and to study the different approaches according to the following criteria:

Type of system	The type of system described: case-based planner, case-based design, etc.
Application domain	The application domain in which the system has been experimented
Knowledge source	The source for knowledge acquisition: from outside the system or from the knowledge already available in the system
Acquired knowledge	Form of the acquired knowledge: cases, adaptation rules, adaptation cases, etc.
Acquisition method	Acquisition strategy exploited: automatic learning, interactive approach
Acquisition period	Step of the CBR system life-cycle where the acquisition is performed: during the system design, during the system use, during the maintenance phase
Details	More information about the approach
References	Related bibliographical references

For further reading, a review focusing more specifically on approaches of adaptation knowledge acquisition is performed in [Lieber et al., 2004]. Three main criteria are used to compare the various approaches: knowledge sources used for acquisition, hypotheses on knowledge representation and type of acquired knowledge. More specific criteria are also used to go further on the comparison of a selected subset of approaches.

Hammond, CHEF

Type of system	Case-based planner
Application domain	Cooking domain (design of recipes)
Knowledge source	Causal model of the domain
Acquired knowledge	Adaptation rules, indexing rules
Acquisition method	Automatic: learning by remembering (storage of successfully adapted plans or repaired plans), failure-driven
Acquisition period	During the reasoning cycle (during the repair step)

Details	CHEF takes advantage of a failure to anticipate it in further reasoning cycles. When an adapted plan fails, it builds up a causal explanation of the failure in order to anticipate a future similar problem. This approach is an incremental repair process.
References	[Hammond, 1986], [Hammond, 1989], [Hammond, 1990]

Aamodt, CREEK

Type of system	General framework for CBR system
Application domain	Several application domains, including oil drilling applications
Knowledge source	Already available knowledge, system user
Acquired knowledge	Cases, domain knowledge
Acquisition method	Learning by remembering, building of domain model
Acquisition period	During the system use
Details	This work is motivated by the observation that, in order to perform its tasks, a CBR system needs a comprehensive model of the general domain knowledge. As argued by the author, this has led to the development of systems combining model-based and case-based approaches such as CREEK. CREEK provides CBR systems with strategies to perform particular learning tasks and to know how to combine these tasks. CREEK learns from every problem-solving experience and even if the learning process is mainly focused on the case base, learning is also possible through interactions with the user.
References	[Aamodt, 1990], [Aamodt, 2004]

Hastings, CARMA

Type of system	Case-based advisor
Application domain	Rangeland grasshopper infestation prediction
Knowledge source	Available knowledge, domain experts
Acquired knowledge	Adaptation parameters (parameters configure influence of each attribute on the solution)
Acquisition method	Automatic, using domain knowledge and a set of training cases given by the expert
Acquisition period	During the reasoning cycle
Details	The learning process consists in learning adaptation weights for a given situation
References	[Hastings et al., 1995], [Branting et al., 2001]

Leake, DIAL

Type of system	Case-based planner
Application domain	Disaster response planning
Knowledge source	Available knowledge (case base and adaptation knowledge) and expert
Acquired knowledge	Adaptation rules, adaptation cases
Acquisition method	Automatic and interactive when the automatic process fails
Acquisition period	During the reasoning cycle
Details	DIAL is inspired by previous case-based planners such as CHEF. The adaptation process is performed by a case-based process on the adaptation process itself: it is a combination of transformations with memory search processes in order to find the required knowledge. Adaptation knowledge consists of general adaptation rules and prior successful adaptation cases. The adaptation effort is stored and reused for an adaptation-guided retrieval approach.
References	[Leake et al., 1996a], [Leake et al., 1996b], [Kinley, 2001]

Hanney and Keane

Type of system	Test-bed applications
Application domain	Car prices and house-selling benchmarks
Knowledge source	Pairs of similar cases from the case base (and, possibly, other available knowledge)
Acquired knowledge	Adaptation rules, possibly generalised adaptation rules
Acquisition method	Automatic (machine-learning approach)
Acquisition period	Before the reasoning cycle
Details	Adaptation rules are generated by examining the differences between problems related to the differences between solutions.
References	[Hanney and Keane, 1996], [Hanney and Keane, 1997]

Jarmulak and Craw, Tablet formulation

Type of system	Case-based design
Application domain	Tablet formulation
Knowledge source	Cases of the case base
Acquired knowledge	Adaptation rules, adaptation cases
Acquisition method	Automatic (introspective learning)
Acquisition period	Off-line

Details	This work develops knowledge light methods for learning adaptation knowledge from the cases in the case base by applying introspective learning. The authors experiment further with several learning algorithm and compare results. The approach is suitable for substitutional adaptation and for both nominal and numerical values in decomposable design problems.
References	[Jarmulak et al., 2001], [Wiratunga et al., 2002], [Craw et al., 2006]

d'Aquin and Lieber, manual knowledge acquisition with experts

Type of system	Decision support
Application domain	Oncology (KASIMIR project)
Knowledge source	Experts
Acquired knowledge	Adaptation knowledge (rules, cases)
Acquisition method	Manual process with the expert
Acquisition period	During system design
Details	Use of informal meeting minutes with experts and knowledge engineers. Emphasis on the role of the decomposition of the global adaptation process into simpler adaptation steps.
References	[Lieber et al., 2001], [Lieber et al., 2005], [d'Aquin et al., 2006]

d'Aquin and Lieber, CABAMAKA

Type of system	Decision helping
Application domain	Oncology (KASIMIR project)
Knowledge source	Pairs of similar cases from the case base (and, possibly, other available knowledge)
Acquired knowledge	Adaptation rules
Acquisition method	Automatic
Acquisition period	Off-line, in a process supervised by the expert
Details	The approach uses a knowledge discovery process (frequent pattern extraction) to build association rules. In other words, data mining algorithms are applied to detect regularities which are candidates to become adaptation rules. A specificity of this system is that interactions with an expert are possible to manually validate the acquired rules.
References	[d'Aquin et al., 2004], [d'Aquin et al., 2007]

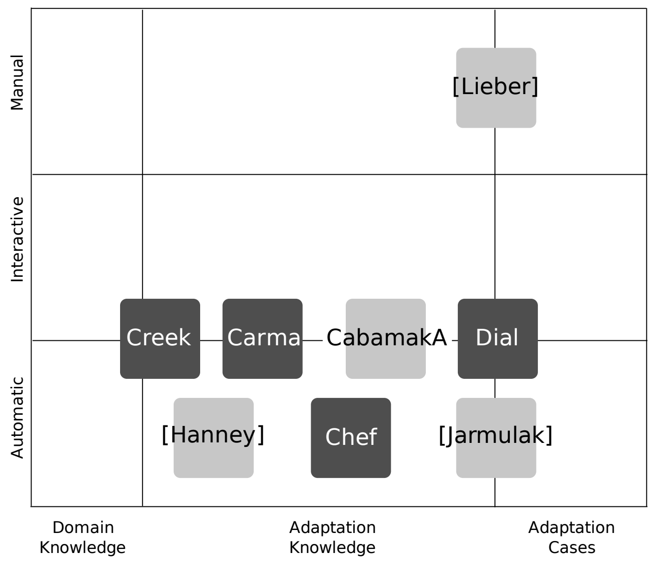
The following researches have not been studied in details although they are interesting approaches to knowledge acquisition. For example, the work described in [Cho et al., 1999] on the domain of travel and in [Lee, 2003] on the domain of car price estimation can be related to the approach developed by Hanney and Keane. In the same vein, but from a more general point of view, Wilke [Wilke et al., 1996] proposes a general framework for automatic adaptation knowledge acquisition that relies on a set of knowledge containers. On an other plane, in [Corchado and Lees, 2003], the authors exploit an artificial neuron network (RBF: radial basis function) to perform adaptation. The learning process occurs during the training of the neural network (it is trained by a set of source cases selected during the retrieval process) but the knowledge is not retained in the system, it is used only for the current problem-solving session. Zehraoui [Zehraoui, 2004] uses connectionist indexing techniques to add cases to a case base. The aim is to improve case retrieval by learning prototypical cases. Learning of similarity measures is also an active research topic. For example, in [Richter, 1992], the authors propose a method for classification and learning of similarity measures. Approaches such as [Cox and Ram, 1999], [Ram, 1993] propose to integrate multiple learning strategies in CBR system to give the user the ability to choose the more suitable learning strategy for the current task. A novel approach that brings together web mining and adaptation knowledge learning issues in proposed in [Leake and Powell, 2007]. In their work, the authors propose the WEB ADAPT system that implements these principles in the travel domain. In some way, the WEB ADAPT system can be related to DIAL and to CABAMAKA. In [Cheng and Hüllermeier, 2008], the authors propose a machine learning approach to similarity assessment that consist in combining local similarities measures in a global one. The originality of the approach rests on the nature of the training information: the method makes use of qualitative feedback in the form of similarity comparisons, revealing which one of two candidate cases is more similar to a third one. Incremental knowledge acquisition is also studied in [Khan, 2003]. In this work, the Ripple Down Rules (RDR) framework [Compton and Jansen, 1988], [Compton and Jansen, 1990] is extended to allow the interactive and incremental development of a knowledge base used for retrieval and adaptation. The author demonstrates how this approach allows the knowledge maintenance task of the knowledge engineer to be overcome. The approach has been implemented in a CBR system named MIKAS (Menu Construction using Incremental Knowledge Acquisition Systems) for the design of menus according to dietary requirements. For all these approaches, as for the one we propose here, an issue can be raised: are these methodologies likely to be useful in other problem-solving domains where experts have an important role to play?

3.1.3 Synthesis of the related work

Figure 3.1 shows the synthesis of studies mentioned previously which follow three lines of study:

- Type of acquired knowledge;
- Method of acquisition;
- Period in which the acquisition takes place.

With regard to the type of acquired knowledge, we focused on all forms of domain and adaptation knowledge. The study of acquisition of cases and knowledge specific to similarity measure was left out as it lays too far outside our field of interest. Concerning



In order to study the knowledge acquisition issue, 8 systems have been compared according to three criteria. The type of knowledge acquired is represented on the horizontal dimension. The method of acquisition is represented on the vertical dimension. The third criterium (period of acquisition) is represented by the colour information: off-line approaches are represented in light grey while on-line approaches are in dark-grey.

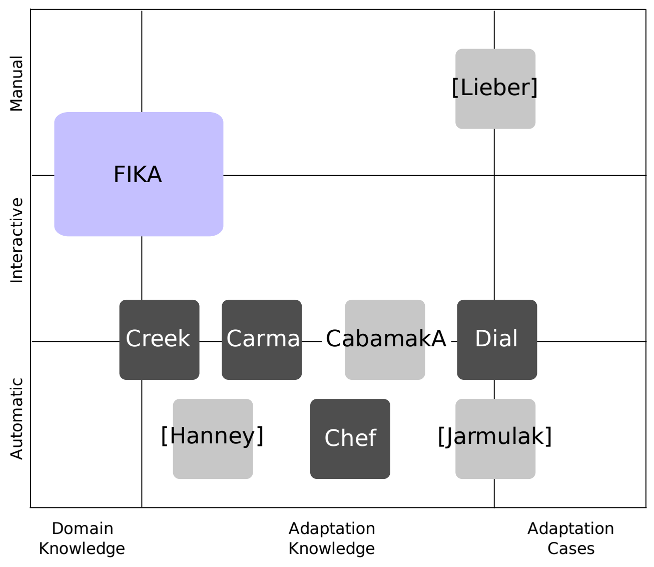
Figure 3.1: Knowledge acquisition in CBR: a review.

the method of acquisition to be used, we make a distinction between automatic, interactive, and manual methods. Some approaches can be called both automatic and interactive, either because they alternate according to the situation, or because they are automatic approaches producing knowledge which must be validated by an expert. Finally, we differentiate between two periods in which the acquisition process may take place: during the reasoning cycle or outside it. When the acquisition takes place during the reasoning cycle, knowledge is acquired while the problem-solving process is going on. Acquisition outside the reasoning cycle may happen either during the design of the system, to build an initial knowledge base, or during the system maintenance phase, to improve the knowledge base as needed.

As shown in figure 3.2, the FIKA approach, which we will describe later, fills an obvious gap by proposing a manual or interactive approach for the acquisition of various CBR knowledge during the reasoning process.

3.2 FIKA: Failure-driven Interactive Knowledge Acquisition

FIKA methodology focuses on the acquisition of CBR knowledge which is "difficult to grasp" and which must be captured "in context". To this end, it proposes an interactive and opportunistic approach of the knowledge acquisition process from the expert. This section explains in details the principles of FIKA.



As an on-line, interactive and opportunistic approach to knowledge acquisition, FIKA is complementary to the usual approaches of knowledge acquisition in CBR.

Figure 3.2: FIKA and knowledge acquisition in CBR.

3.2.1 Motivations

The building of FIKA principles originates from several observations using different experiences in knowledge acquisition, particularly within our research team. In a CBR system, the knowledge acquisition process is very difficult to model as it entails a lot of "test and error". It can however be summarised in three main steps:

- During the construction phase: building of the initial case base, which should reflect the domain under study, and modelling of a certain amount of similarity and adaptation knowledge to reason on cases;
- During the system testing phase: solving of a number of problems by experts with the help of knowledge engineers, increase of the case base and improvement of associated knowledge bases;
- As soon as the system is efficient enough, use in real life set-up by domain practitioners, evolution of the knowledge system, modification of the practitioner's understanding of his domain.

Even if the FIKA approach can be used efficiently to support the acquisition of initial knowledge, it was created to answer the interactivity requirements in the evolution process of existing knowledge bases. Our team's experience has shown that this type of approach could prove very useful and that it is missing in numerous systems, thus preventing their evolution. FIKA is not an automatic learning approach (in the machine learning sense). Indeed, in symbolic and/or numerical learning approaches, algorithms mobilise data or knowledge already available in the system to build new knowledge. FIKA is applicable in situations where automatic learning approaches cannot be used, particularly when already available knowledge is not sufficient. The knowledge in this case is often precise and issued from experience.

3.2.2 FIKA principles

The scenario illustrated on figure 3.3 corresponds a typical use of a CBR system by an expert.[4] It is to be observed that, in order to solve a particular problem, the expert is interacting permanently with the system and its environment. The expert tests the solutions in real life and if the test fails, he interacts with the system to point out the error and possibly to amend or add to the knowledge that caused the error. The FIKA approach is adapted for the acquisition of knowledge in situations like these. The various notions introduced in this example: the expert, the role of "real life", the mode of interaction with the system, the notion of knowledge, will be discussed in detail later in this section.

An interactive approach

The interactive aspect of the FIKA approach lies in the fact that the building of knowledge to be acquired is done through the interactions between the expert and the system. The knowledge acquisition process is part and parcel of the CBR process itself: in this way, one is assured of not diverting the expert from his main task. Then the solution of the problem is a process in which the reasoning capacities and the knowledge of the system and the expert are "blended" into a loop in order to reach the target. Knowledge acquisition can then benefit from this loop.

Another benefit of this approach is that the expert can go on learning in his own domain as he uses the system. Through interaction with the system, he improves his understanding of the way the system works, how its reasoning is built, which important piece of knowledge has not yet been modelled into the system.

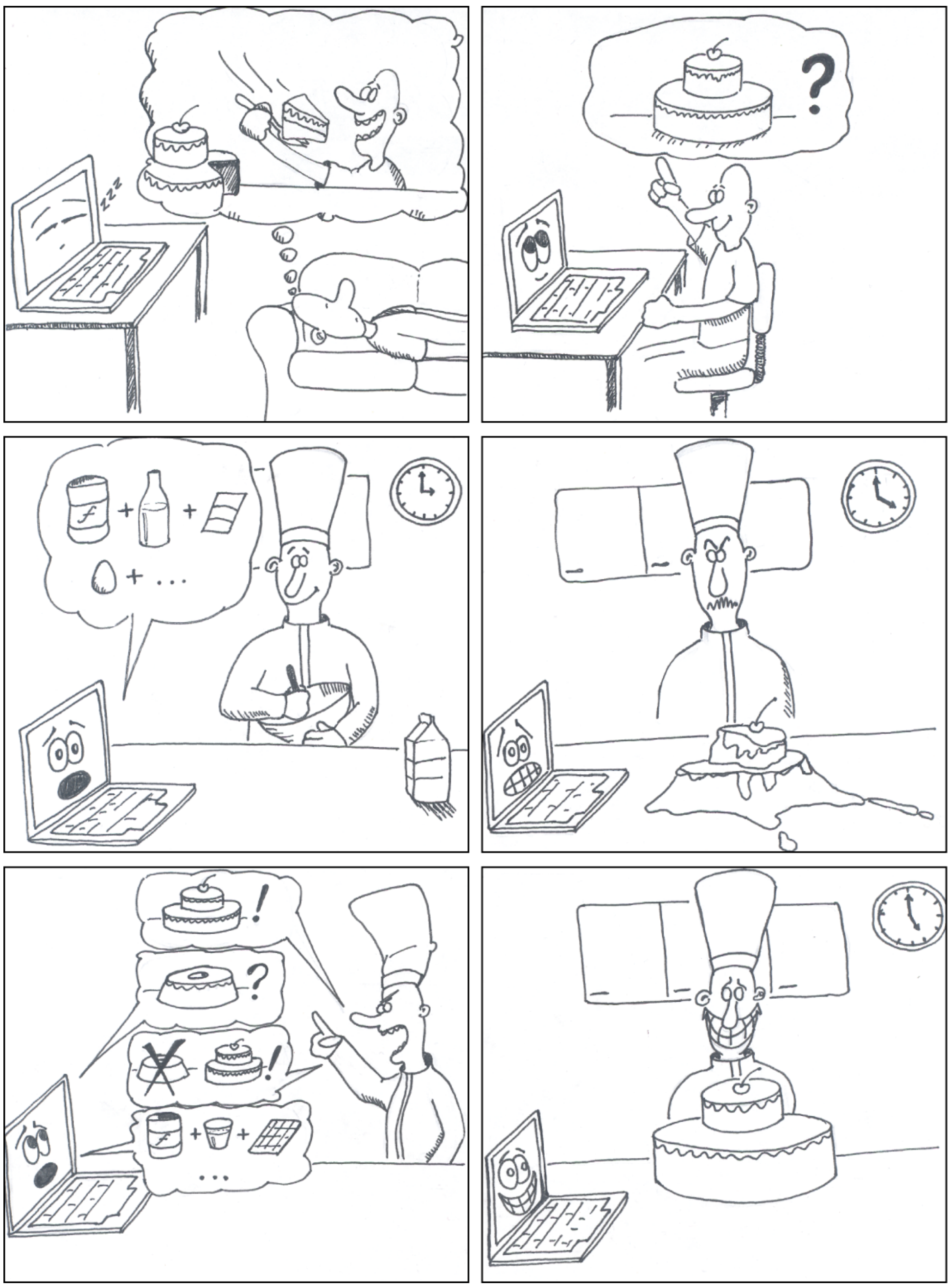
The pair "expert/CBR system" forms a system which we call "learning system". This idea is illustrated in figure 3.4. This ability to learn from each problem-solving experience is refered to as " sustained learning" in [Aamodt, 1989].

An opportunistic approach

The knowledge acquisition process in FIKA is triggered off by "reasoning failures": that is the opportunistic aspect of the approach. A reasoning failure occurs when the solution proposed by the system is deemed unsatisfactory by the expert. Supposing that the reasoning engine is correct (in other words, that the system does not go wrong), then we must admit in such a case that the knowledge used to solve the problem is incorrect, incomplete, or in-adapted to the given situation and must therefore be improved. An important problematic is to know then how to recognise a reasoning failure and when to start the interactive acquisition process.

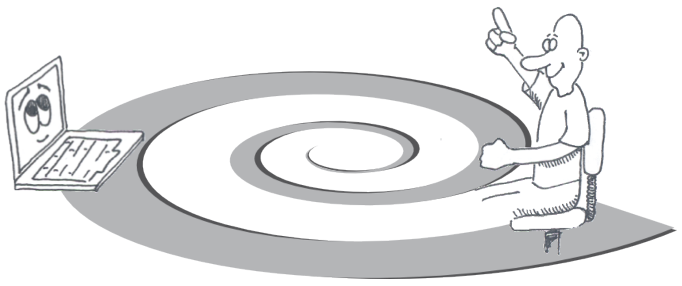
To sum up, the opportunistic aspect of the FIKA approach places the expert in favourable context for the acquisition of precise knowledge. In the previous example, the first cake was spoilt because of a problem in the recipe. In this particular context, the cookery expert was quite capable of identifying that the problem was linked to some ingredients used in the

[4]This comic has been drawn by Mauricio Meurer after a scenario written for the purpose of this work.



This scenario illustrates a use-case of a CBR system (such as TAABLE, described in chapter 6). The user asks the system for a recipe for a cake. When he tries the recipe suggested by the system, he realises that there is a problem (actually, the cake is spoilt). Through interactions with the system, the expert manages to explain why the cake is spoilt and the system is able to find a new solution. The system is able to learn new knowledge from this failure and to reuse this knowledge to avoid the failure occurring again.

Figure 3.3: A use-case of a CBR system.



Expert and system are involved in a learning “spiral” where each one is able to learn on the domain through interactions with the other. The issue for knowledge acquisition tools is to provide an adapted support for these interactions.

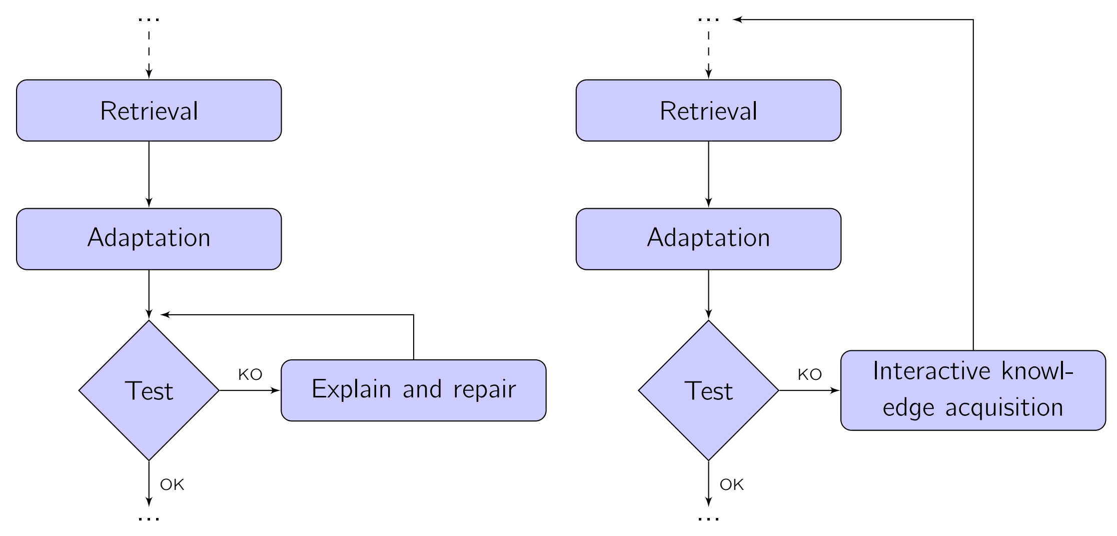
Figure 3.4: Expert and CBR tool, a learning system.

recipe, to repair the problem and to transmit the knowledge related to this situation to the system.

3.2.3 FIKA in the CBR process

The idea of using reasoning failures to improve system knowledge is not new. From the beginning of CBR research, certain systems used reasoning failures to acquire new knowledge. This is called “failure-driven learning”. For example, in CHEF [Hammond, 1986] (as well as in PERSUADER [Sycara, 1992] and MEDIATOR [Simpson, 1985]), when failure occurs, an explanation of the reason for the malfunction is given. The explanation is then used to repair the proposed solution to the case. The repaired case is stored; the explanation may also be stored. Then the case base is reorganised so that the repair which has just been made is found and reused in future if a similar situation occurs. CHEF is part of the systems which store repairs to avoid repeating errors in future situations. In systems producing failure-guided learning, the failure report comes from the real world: it is either reported by the user or it is noticed by the system itself, when it tries unsuccessfully to apply the solution it proposes. Reasoning failures, reported and treated during the test-explanation-repair stage, are at the root of a knowledge acquisition process.

Figure 3.5 compares the integration of the FIKA approach into the CBR cycle with the traditional approach proposed by Riesbeck and Schank [Riesbeck and Schank, 1989]. The left part of the figure (figure 3.5(a)) is a summarised version of the initial CBR flowchart (the original diagram is shown in chapter 2, figure 2.2). In this approach, explanations and repairs are memorised in order to be reused in later situations. The loop is made at the test-explication-repair stage until a satisfactory solution is found. Knowledge acquired during this process is then added to the various knowledge bases used by the system. On the contrary, in FIKA (cf. figure 3.5(b)), all that is learnt during the repair stage is immediately added to the knowledge bases and is used to produce a new solution to the problem. In fact, since the knowledge acquired is that used in remembering and adaptation, it must be reused during the reasoning process to test the relevance of the acquisition, and also to facilitate the solution of the problem under processing.



(a) Test, explanation, repair according to Riesbeck and Schank.

(b) Interactive knowledge acquisition in FIKA.

Figure 3.5: Failure-driven learning and FIKA: a comparison.

3.2.4 Definitions

Knowledge

The main focus of this work is *knowledge* acquisition, however, it is not in our scope to discuss the concept of "knowledge" nor of the meaning of "system's knowledge". Nevertheless, we would like to underline a work on this subject that is of major importance for us. This work is the one of [Bachimont, 2004].

Bachimont defends the notion of "knowledge inscriptions" as symbols representing knowledge in computer system, by contrast with the concept of "knowledge of the system" which is meaningless if not considered with regard to the system's user. According to this definition, knowledge engineering should be a discipline aiming at instrumenting the user's cognitive work associated to the building of knowledge inscriptions in the system. Given such a definition, we can only speak about "knowledge" when the symbolic manipulations performed by the system on the symbols "make sense" and have a justification to the user interacting with the system. Hence, the quality of a knowledge engineering system should be evaluated by its capability to be properly used in situation and to solve problems it has been conceived to solve.

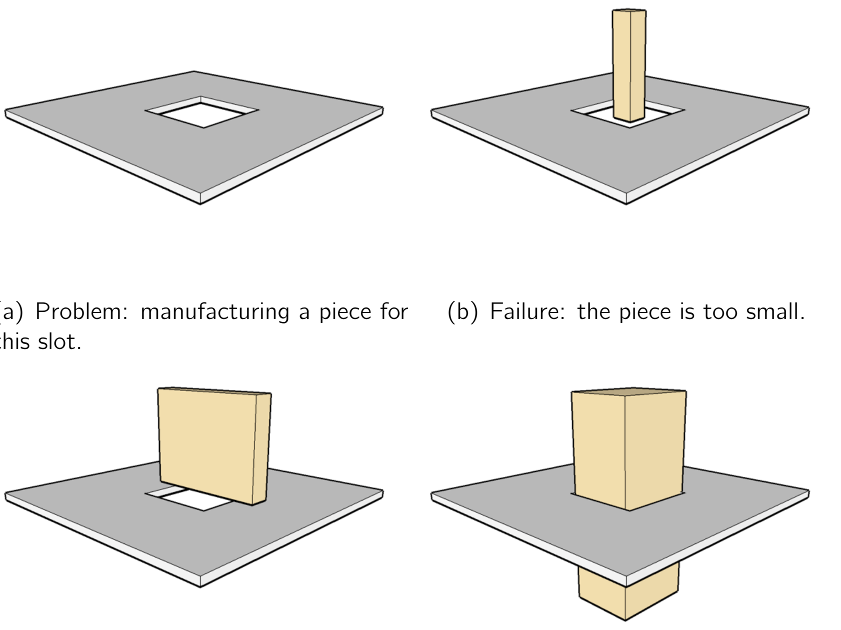
We want to insist on the fact that we adhere to this approach. When we speak about "knowledge" of the system, we in fact speak about "inscriptions" of knowledge that make sense for the user. We want to provide a support for the interactions between the system and its user in order to contribute to the development of applications making sense to their users and efficient to solve problems for which they were conceived.

Oracle

The oracle "knows" if a solution is correct or not. Moreover, the oracle never makes mistakes. We could compare the concept of oracle we use here to the notion of "outcome" introduced by Kolodner in [Kolodner, 1993]: the oracle gives a feedback of the reality.

For example, the oracle is able to say if the duck à l'orange he is eating is tasty or not. However, if the meal is not tasty, the oracle is not able to say why; for example he is not able to say that the meal misses salt and pepper. In addition, he is not able to explain how the recipe should be modified in order to make it tastier next time.

To better illustrate the oracle concept, consider an example from an other domain, the carpentry domain. The virtual CBR system considered in this example is responsible for the elaboration of wooden pieces used to assemble carpentry elements. In the case illustrated on figure 3.6(a), the goal is to manufacture a piece that is perfectly adjusted to the slot in the flat element.



(a) Problem: manufacturing a piece for this slot.

(b) Failure: the piece is too small.

(c) Failure: the piece is too large.

(d) Success: the piece is well suited to the slot.

Figure 3.6: Oracle and expert, an example from the carpentry domain.

In the cases 3.6(b) and 3.6(c), the manufactured pieces are not suitable, either because they are too large or too small. The only thing the oracle is able to say is that, in both cases, the piece is not adapted (this is the result of trying to put the piece in the slot), but he is not able to explain how to modify the pieces to make them adapted. By contrast, the oracle is able to say that the piece in situation 3.6(d) is adapted.

Expert

Usually, when an expert tries to solve a problem, he uses an approach based on a "know-how" more than on an explicit theory. Here, we consider the expert as a domain specialist. The expert has a knowledge that makes him able to understand his domain, he is able to explain why a solution given by a system is unsatisfactory and he is able to identify "errors" in the modelled knowledge or, in some cases, missing knowledge.

In the carpentry example introduced before, the expert, by contrast with the oracle is able to explain that in the first situation of failure, the solution is to reduce the width and to augment the length of the piece. The expert is also able to explain why the case 3.6(b) is also a failure that cannot be fixed.

Collaboration of the oracle and the expert to solve a problem

In summary, the oracle identifies faulty solutions and then, the expert is in charge of analysing the failure in order to identify the origin of the problem.

When a cook tastes the meal he has just prepared, he plays the role of the oracle: he knows if it is tasty or not. If he observes that the meal is not enough salted, then he plays the role of the expert. In the case of the carpentry example, it is the confrontation of the solution to the reality that plays the role of the oracle: the piece is tested in real conditions. The expert, i.e. the carpenter, can then repair the proposed solution if it is possible.

3.3 Implementations of FIKA

FIKA is a general approach for interactive and opportunistic knowledge acquisition. IAKA and FRAKAS are models implementing the principles of FIKA. IAKA-NF and FRAKAS-PL are prototypical applications that respectively illustrate the properties of the IAKA and the FRAKAS models (the development of IAKA-NF and FRAKAS-PL has lead us to make some implementation choices, they are discussed in the followings chapters). IAKA-NF(f) and FRAKAS-PL(ONCO) are application: they correspond to the use of IAKA-NF and FRAKAS-PL on specific domains and with specific knowledge bases.

Figure 3.7 gives a global vision of the organisation of our different contributions. It must be remarked that TAAABLE is an application that we have used to manually experiment with some FIKA principles.

The three following chapters are dedicated respectively to IAKA, FRAKAS, and TAAABLE. This section only gives a quick overview and a comparison of IAKA and FRAKAS. As it has not been formalised yet, the method developed in TAAABLE is not taken into account in this comparison.

3.3.1 IAKA, a method for acquiring cases and adaptation knowledge

IAKA provides a methodology for adaptation knowledge acquisition in CBR systems where the reasoning can be decomposed in reasoning steps. In this approach, the assumption is made that the knowledge to be acquired is often linked to the cases. The identification of the knowledge to acquire is performed through an analysis of the reasoning leading to the solution. One of the advantages of such an approach is that the decomposition of the

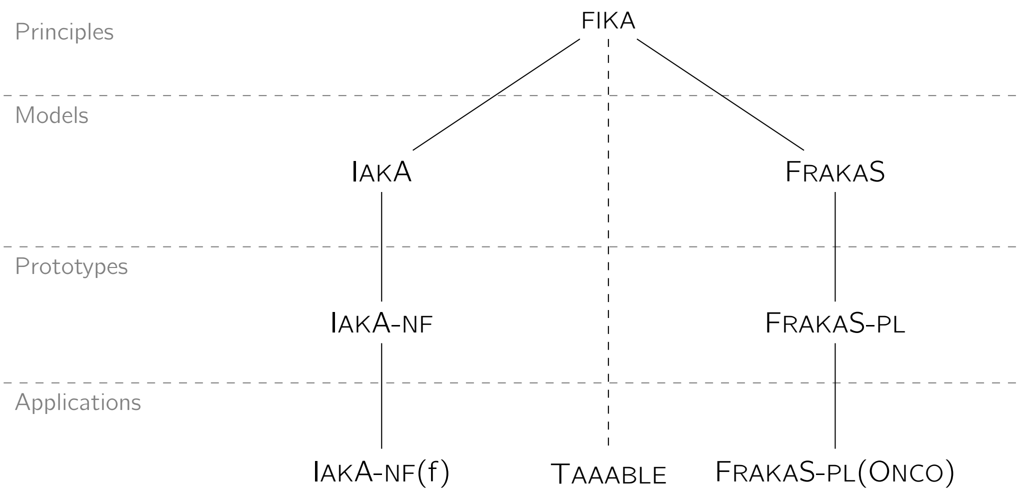


Figure 3.7: FRAKAS, IAKA and FIKA.

reasoning allows the used knowledge and the missing knowledge to be easily identified. Indeed, in order to solve a problem, the system builds intermediate problems, each intermediate problem corresponding to the application of a specific knowledge. When a failure occurs, the intermediate problems are shown to the expert who can, then, analyse each knowledge independently from the others.

IAKA-NF, the IAKA prototype, implements the differential adaptation strategy that appears to be well suited to an approach such as FIKA.

3.3.2 FRAKAS, a method for acquiring domain 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.

FRAKAS-PL, the FRAKAS prototype, is based on the implementation of the conservation adaptation paradigm, where domain knowledge is used to guide adaptation.

3.3.3 Comparison of IAKA and FRAKAS

The following tables show briefly the main characteristics of IAKA and FRAKAS (table 3.1), and compare them to associated prototypes (table 3.2). These tables are revised and completed in the discussion of this document (cf. chapter 7) in order to present a synthesis of the approaches developed in this study.

	IAKA	FRAKAS
Knowledge to acquire	Cases and adaptation knowledge	Domain knowledge
Type of solution (candidate solution, denoted $\widetilde{\texttt{Sol}}(\texttt{tgt})$)	Best operational solution according to the knowledge available in the system (this entails the notion of quality of a solution with respect to a given problem)	Operational solution consistent with the system domain knowledge
Reasoning failure	The solution is not "good enough" according to a predefined set of criteria	The solution is inconsistent with respect to the expert's knowledge
Interaction mode	The expert analyses each intermediate solution	The expert analyses the final solution
Reasoning mode	Decomposition of the reasoning process in several steps (intermediate problems)	Monolithic reasoning
Aim of the approach	Progressive improvement of an initial knowledge base, building of an initial knowledge base	Progressive improvement of an initial knowledge base, (the building of an initial knowledge base is possible too, but less relevant in this case)

Table 3.1: IAKA and FRAKAS: a comparison.

3.3.4 Synthesis

The table 3.3 presents a short synthesis of the main principles of the approach. This table is given at the end of each one of the three following chapters to show in what each of the approaches we describe is a variation of the FIKA approach.

3.4 Conclusion

Case-based reasoning is by now a fairly well formalised process and the major stake for researches in this field is the development of efficient knowledge acquisition methods and tools to feed such systems. The thesis described in this document proposes a generic knowledge acquisition approach (FIKA) adapted to case-based reasoning, which differs from other approaches in a number of characteristics: interactive, opportunistic (i.e., guided by reasoning failures), user centred and linked to the reasoning cycle. In FIKA, the expert domain user plays a major and active part, since he participates in the construction of knowledge which will be represented in the system. Thus, the knowledge used by the system makes sense to the expert and the "expert-CBR system" pair appears as a learning system in which the expert and the system solve —and learn to solve— problems together.

A criticism which might be levelled at FIKA is that this approach requires a lot of effort to acquire eventually only small knowledge units. FIKA is not in fact an approach designed

	IakA-nf	FrakaS-pl
Adaptation strategy	Differential adaptation	Conservative adaptation
Implementation details	Full system with textual interface, virtual expert and virtual oracle	Full system with graphical interface to support interactions
Implementation language	Java	Java
Experimentation domain	Numerical functions	Breast Cancer Treatment
Experimentations	Numerical experiments with various functions	Manual experimentations, but not with a real expert

Table 3.2: IakA-nf and FrakaS-pl: a comparison.

FIKA general principles	
Description	Principles
Knowledge intensive	The knowledge targeted by the knowledge acquisition process comes from outside the system (by contrast with knowledge-light approaches).
Interactive	The expert is strongly involved in the problem-solving process *and* in the knowledge acquisition process.
Opportunistic	The system exploits reasoning failures to trigger the knowledge acquisition process thus ensuring that the expert is already "in context" and consequently in "optimal conditions" to participate to the knowledge acquisition process.

Table 3.3: FIKA general principles.

for quantity, but for the acquisition of precise knowledge, in context. However, FIKA answers a particular need, that of acquiring knowhow, not theoretical knowledge. Because of its experiential nature, this practice-linked knowledge proves more efficient than other acquisition approaches used in CBR. The acquisition of knowledge as performed by FIKA is iterative: at each reasoning cycle, new pieces of knowledge are acquired and the system is therefore more efficient during the following cycle.

FIKA can be seen as a complement to traditional automatic approaches for knowledge acquisition in CBR. It is an additional brick in the construction of a global knowledge acquisition method covering all aspects of reasoning. The study of the various possible links between FIKA and other approaches constitutes a very important perspective for this work.

Following chapters are dedicated to the description of IakA and FrakaS which are two extensions of FIKA.

S'il n'y a pas de solution, c'est qu'il n'y a pas de problème.

Devise Shadok

4 IAKA, INTERACTIVE KNOWLEDGE ACQUISITION

This chapter presents the IAKA approach for interactive knowledge learning in CBR systems. A system implementing the IAKA principles is able to learn adaptation knowledge during the CBR cycle. The approach conveys the different aspects described in FIKA: opportunism, as IAKA is a failure-driven approach, and interactivity as it is performed on-line and exploits experience feedback.

This chapter also describes IAKA-NF, a CBR system prototype that carries out the IAKA principles. IAKA-NF is a prototype for testing IAKA in the domain of numerical functions where both the expert and the oracle can be simulated. The main advantage of IAKA-NF is that it shows how, under certain assumptions, the principles of IAKA can be implemented in a real-world application to "enrich" knowledge bases. This prototype has been implemented using a specific knowledge representation formalism (numerical functions). The adaptation is performed by the mean of differential adaptation strategy and both the expert and the oracle are simulated by an external program. Several experiments have been conducted with IAKA-NF. They allow us to validate the IAKA principles and to study several issues of the knowledge learning process in CBR.

Experiments with IAKA-NF, even if they were performed with a virtual expert and a virtual oracle, gave promising results and opened several research directions that are discussed at the end of the chapter.

Related publications:
[Cordier et al., 2007d], [Cordier et al., 2007b], [Cordier et al., 2008a]

Main related work:
[Kinley, 2001], [Bergmann and Wilke, 1998]

Keywords:
Adaptation knowledge acquisition, differential adaptation, adaptation cases

Contents

Contributions of this chapter

- A formalisation of a knowledge acquisition process which extends FIKA, focusing on adaptation operators
- An implementation of the differential adaptation strategy in a prototype
- Development of a prototype application (IAKA-NF) and experiments with it

Several ways of performing knowledge acquisition have been explored in CBR related research. For instance, knowledge engineers and domain experts can collaborate to model knowledge of the domain. This manual approach is efficient because it allows the acquisition of relevant knowledge coming from the expert but it is constrained by the availability of the expert and of the knowledge engineer. Other approaches rely on the knowledge already available in the system (often in the cases) to infer new knowledge, like adaptation rules. Theses approaches are efficient in the sense that they automate the acquisition process but they produce a large amount of knowledge that has to be validated by an expert. Moreover, this validation phase is performed off-line, out of a specific context, thus it may be felt by an expert as an irksome task. Hybrid approaches, such as the one presented in this chapter, combine the reasoning capabilities of the system with interactions with the user to acquire missing knowledge in context. Indeed, IAKA is an approach for interactive knowledge learning in CBR systems. It defines a knowledge representation formalism for cases and adaptation knowledge in CBR systems and a set of associated strategies and methods for knowledge acquisition. The representation formalism ensures a fine-grained representation of the available knowledge thus allowing facilitating the precise identification of the faulty pieces of knowledge through interactions.

IAKA implements the FIKA principles: interactiveness and opportunism. IAKA is interactive in so far as it exploits interactions between an expert, an oracle and the system during CBR sessions. Its opportunistic aspect is due to the fact that reasoning failures trigger the acquisition process: the system seizes this opportunity to identify missing knowledge and to acquire it. One of the main advantages of this approach is that it focuses on knowledge known to be needed, which constitutes a strong guidance for the knowledge acquisition process and alleviates the effort required by the expert. IAKA contributes to defend a unified view of CBR steps and knowledge containers: the CBR process is considered in its global nature and the knowledge acquisition process focuses on *the knowledge of the system* (cases and adaptation knowledge are closely linked and are acquired at the same time). When a failure occurs, the applicability of the adaptation knowledge for this case has to be questioned. In this case, an interactive process involving the end user allows the system to precisely identify and correct the faulty knowledge. As the adaptation knowledge is corrected in the context of the case being solved, it stays linked with the case. The acquisition of new cases and adaptation knowledge allows the knowledge base of the system to be progressively improved.

This chapter is organised as follows. Section 4.1 is dedicated to the description of the IAKA principles. Notions and notations used in this chapter are presented and the implementation of the IAKA principles in a CBR system are discussed. Section 4.2 presents IAKA-NF, a prototypical CBR application that implements the IAKA principles with a particular formalism, that of numerical functions. This section illustrates how, under certain assumptions, the IAKA principles can be implemented in real world applications. Section 4.3 presents experimental results observed with IAKA-NF in several situations. The experiments are relevant both to validate the principles advocated in IAKA and to deal with more general issues of knowledge acquisition in CBR. Section 4.4 discusses the approach, compares it with related work and draws some prospects for this work. Finally, section 4.5 concludes this chapter.

4.1 The IAKA approach

IAKA stands for "InterActive Knowledge Acquisition". The main idea of this approach is to exploit reasoning failures and their correction to learn cases and adaptation knowledge. A reasoning failure occurs when the candidate solution given by the system is not "good enough" with respect to a given criteria. The occurrence of a failure highlights the fact that knowledge is missing. When correcting a failure (i.e., when improving the quality of the candidate solution), the required knowledge is added to the knowledge base and is reused in the following reasoning sessions to improve the solutions. The acquisition process is made possible through interactions with an *oracle* who is capable of pointing out that a given solution is incorrect and to an *expert* who is capable of providing the necessary adaptation knowledge.

The explanation of the principles of IAKA requires a set of definitions and hypotheses on the oracle, on the expert and on knowledge available in the system. These definitions and hypotheses are given hereafter.

4.1.1 Expert and oracle in IAKA

For being able to perform knowledge acquisition, a system needs a feedback on the solutions it produces. For this purpose, in IAKA, we make the assumption that an *oracle* and an *expert* are available. The system can interact with the oracle to know if a solution is correct or not, but the oracle knowledge remains inaccessible. The expert is capable of correcting an incorrect solution and of building (in collaboration with the system) missing adaptation knowledge when he is asked for.

The role of a CBR system is to produce an operational solution, i.e., a solution that is easily applicable and that will help the user on his problem-solving task. Let us assume that Léon wants to cook a dessert with rhubarb. He asks his favourite CBR system for a dessert recipe with rhubarb as a main ingredient. If the system provides him with a detailed rhubarb pie recipe, Léon will be satisfied. On the contrary, if the system provides him with a recipe such as "Take some rhubarb, some sugar and some pastry, make a tart", Léon will ask for a more detailed solution.

An important issue is to define how good a solution is. From a practical point of view, a good (i.e., satisfactory) solution is a solution that fulfils a set of satisfaction criteria defined by the expert. From a formal point of view, it is often possible to define a measure of satisfaction. For example, if there is a way to know the "ideal" solution, then, the candidate solution can be compared to the "ideal" solution. In IAKA, we consider that a candidate solution is a satisfactory solution when the gap between the candidate solution and the "ideal" solution is smaller than a given threshold (depending on the expert's level of demand and called "tolerance threshold"). This measure of satisfaction is given by the oracle. The roles of the oracle and the expert can be summarised as follows.

The oracle:

- Is able to say if the candidate solution is correct or not;
- Is able to give an estimation of "how good" the candidate solution is given the current problem.

The expert:

- Has a tolerance threshold denoted by ε, that defines his "level of satisfaction" and that is used by the pair "CBR system - oracle" to decide if a solution is satisfactory or not;
- Interacts with the system to improve knowledge when unsatisfactory solutions occur.

4.1.2 Definitions and hypotheses for IAKA

The purpose of this section is to introduce the vocabulary used in the IAKA approach. IAKA assumes that all the knowledge available in the system is represented by descriptors, cases, adaptation operators and adaptation methods. Each one of these elements as well as their relations are going to be defined. The hypotheses made in IAKA are generic enough to be suitable for a large number of CBR systems.

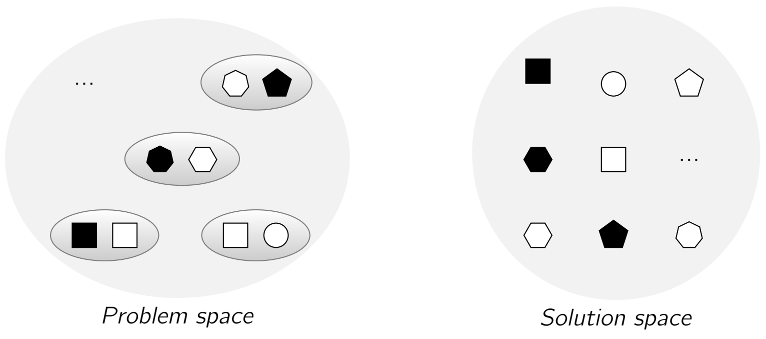
The Flatland domain

In order to illustrate the definitions described in this section, a fictive application domain has been chosen: Flatland. This domain is strongly inspired by the novel *Flatland* by Edwin A.Abbott [Abbott, 1884].

In this domain, problems consist of ordered pairs of shapes and solutions consist of single shapes. Shapes have two properties: number of edges and colour (figure 4.1 illustrates problem and solution spaces in Flatland).

It must be remarked that there is no available rule allowing the immediate computation of the solution given the knowledge on the problem.

Problem and solution spaces in Flatland



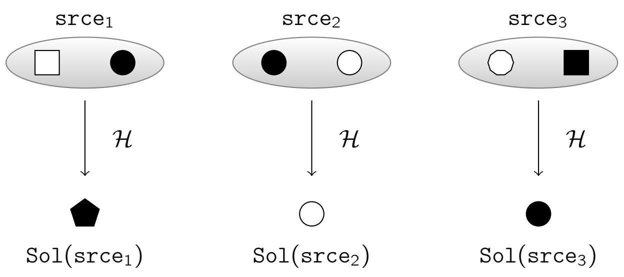
A problem is an ordered pair of shapes, a solution is a single shape.

Figure 4.1: A subset of the problem space and the solution space.

Cases

Notions of *problem* and *solution* are defined as follows. If pb is a problem (resp., sol is a solution), then pb (resp., sol) is an expression in a knowledge representation formalism representing a problem (resp., a solution). $\mathcal{L}_{\texttt{pb}}$ denotes the problem space and $\mathcal{L}_{\texttt{sol}}$ denotes the solution space. Moreover, a binary relation on $\mathcal{L}_{\mathtt{pb}} \times \mathcal{L}_{\mathtt{sol}}$ is assumed to exist with the semantics "has for solution". The pairs $(\mathtt{srce}, \mathtt{Sol}(\mathtt{srce})) \in \mathcal{L}_{\mathtt{pb}} \times \mathcal{L}_{\mathtt{sol}}$, such that $\mathtt{srce}$ has for solution $\mathtt{Sol}(\mathtt{srce})$, are called *source cases*. The relation denoted by $\mathcal{H}$ is a set of dependence relations indicating that there exist links (dependencies) between the problem part of a case and its solution part. With the knowledge of $\mathcal{H}$, the system is able to make a suitable modification on a solution part of a case when there is a modification on the problem part. The aim of the CBR process is to find a solution for the target problem denoted by $\mathtt{tgt}$ by adaptation of a source case. As there is no guarantee that the result of the adaptation gives a "valid" solution, the solution produced is called a *candidate solution* and is denoted by $\widetilde{\mathtt{Sol}}(\mathtt{tgt})$. A candidate solution becomes a solution once it is validated by an expert through interactions. Figure 4.2 shows examples of source cases from the Flatland domain.

Examples of source cases



A case consists of a problem part and a solution part. There exist links between problem descriptors and solution descriptors, they are materialised by the relation $\mathcal{H}$.

Figure 4.2: Examples of source cases in Flatland.

Adaptation knowledge

Adaptation knowledge is assumed to be organised in adaptation methods. An adaptation method consists of a set of adaptation operators. An adaptation method contains the necessary knowledge to compute a candidate solution $\widetilde{\mathtt{Sol}}(\mathtt{tgt})$ for the target problem $\mathtt{tgt}$ given $(\mathtt{srce}, \mathtt{Sol}(\mathtt{srce}))$. Each adaptation operator performs local changes on small parts of the problem and the solution (usually, on descriptors). The knowledge contained in an adaptation operator can be decomposed in two elements: a relation $\mathtt{r}$ and an adaptation function $\mathcal{A}_{\mathtt{r}}$. $\mathtt{r}$ can be seen as a (dis)similarity relation between problems.

Definition 4.1 (Adaptation operator —$\mathtt{AO}_{\mathtt{r}} = (\mathtt{r}, \mathcal{A}_{\mathtt{r}})$)

An *adaptation operator* $\mathtt{AO}_{\mathtt{r}}$ is a pair $(\mathtt{r}, \mathcal{A}_{\mathtt{r}})$ where $\mathtt{r}$ is a binary relation between problems ($\mathtt{r} \subseteq \mathcal{L}_{\mathtt{pb}} \times \mathcal{L}_{\mathtt{pb}}$).

$\mathtt{r}$ is a relation between problems meaning that $\mathtt{tgt}$ is related to $\mathtt{srce}$ and $\mathcal{A}_\mathtt{r}$ is an adaptation function allowing the computation of $\widetilde{\mathtt{Sol}}(\mathtt{tgt})$ from $\mathtt{Sol}(\mathtt{srce})$ given the knowledge of $\mathtt{r}$.

Definition 4.2 (Adaptation function —$\mathcal{A}_\mathtt{r}$)

$\mathcal{A}_\mathtt{r}$ is an adaptation function:

if $(\mathtt{srce}, \mathtt{Sol}(\mathtt{srce}), \mathtt{tgt}) \in \mathcal{L}_{\mathtt{pb}} \times \mathcal{L}_{\mathtt{sol}} \times \mathcal{L}_{\mathtt{pb}}$ and $\mathtt{srce}\ \mathtt{r}\ \mathtt{tgt}$
then $\mathcal{A}_\mathtt{r}(\mathtt{srce}, \mathtt{Sol}(\mathtt{srce}), \mathtt{tgt})$ is a candidate solution of $\mathtt{srce}$.

Given two problems related by $\mathtt{r}$, the adaptation function $\mathcal{A}_\mathtt{r}$ allows the computation of a solution for a problem by adaptation of the solution of another problem (and according to $\mathtt{r}$).

$$\mathcal{A}_\mathtt{r} : (\mathtt{srce}, \mathtt{Sol}(\mathtt{srce}), \mathtt{tgt}) \longmapsto \widetilde{\mathtt{Sol}}(\mathtt{tgt}) \qquad \text{(Adaptation function)}$$

$\widetilde{\mathtt{Sol}}(\mathtt{tgt})$ is a candidate solution for $\mathtt{tgt}$ obtained by adaptation of $\mathtt{Sol}(\mathtt{srce})$ and taking into account the differences between $\mathtt{srce}$ and $\mathtt{tgt}$ (relying on the knowledge of the relation $\mathtt{r}$).

To put it another way, an adaptation operator can be viewed as an adaptation rule:

if $\mathtt{srce}\ \mathtt{r}\ \mathtt{tgt}$
then $(\mathtt{srce}, \mathtt{Sol}(\mathtt{srce}))$ can be adapted by $\mathcal{A}_\mathtt{r}$ in a candidate solution $\widetilde{\mathtt{Sol}}(\mathtt{tgt})$ of $\mathtt{tgt}$

Figure 4.3 gives a graphical view of the concept of adaptation operator and example 4.1 illustrates this concept in the Flatland context.

$$\begin{tikzcd}
\mathtt{srce} \arrow[r, dash, "\mathtt{r}"] \arrow[d, "\mathcal{H}"'] & \mathtt{tgt} \arrow[d, "\widetilde{\mathcal{H}}"] \\
\mathtt{Sol}(\mathtt{srce}) \arrow[r, dash, "\mathcal{A}_\mathtt{r}"'] & \widetilde{\mathtt{Sol}}(\mathtt{tgt})
\end{tikzcd}$$

Figure 4.3: An adaptation operator in IAKA.

Example 4.1 - An example of adaptation operator

Figure 4.4 is a first illustration of the concept of adaptation operator. The retrieved source case (problem and solution) is represented on the left hand of the figure. The target problem and the candidate solution obtained by adaptation are represented on the right hand. The relation $\mathtt{r}$ between $\mathtt{srce}$ and $\mathtt{tgt}$ has the following meaning: *to go from $\mathtt{srce}$ to $\mathtt{tgt}$, an edge has to be added* *to the first shape of the pair.* In other words, the *difference* between `srce` and `tgt` is that there is one more edge on the first shape of `tgt`. Except this difference, all the relevant attributes of the shapes are identical. To `r` is associated the adaptation function $\mathcal{A}_\mathtt{r}$ which meaning is: *If there is one more edge on the first shape of the target problem, then the source solution must be adapted by adding one edge to it.* Hence, $\widetilde{\mathtt{Sol}}(\mathtt{tgt})$ is obtained by application of $\mathcal{A}_\mathtt{r}$ on $\mathtt{Sol}(\mathtt{srce})$.

Adaptation operator

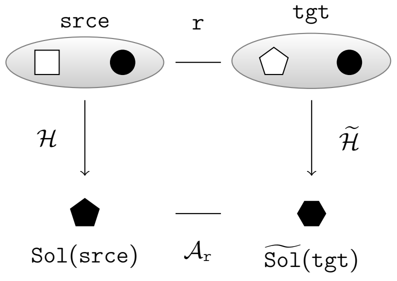


Figure 4.4: An adaptation operator in Flatland.

An adaptation method consists of a finite set of adaptation operators. An adaptation method is always associated to a case. When the purpose is to adapt a case, the adaptation operators that are to be used are chosen in one of the adaptation methods linked to the retrieved source case. The same adaptation method can be linked to several cases. A single case can be associated to several adaptation methods. No assumption is made on the way the association of the adaptation method to the source case has to be performed.

Definition 4.3 (Adaptation method —$\mathtt{AM}_\mathtt{srce}$)

The adaptation method $\mathtt{AM}_\mathtt{srce}$ associated with the case $(\mathtt{srce}, \mathtt{Sol}(\mathtt{srce}))$ is a finite set of adaptation operators $\mathtt{AO}_\mathtt{r} = (\mathtt{r}, \mathcal{A}_\mathtt{r})$.

An adaptation operator in IAKA is what is called a reformulation in [Melis et al., 1998]. The notions of adaptation operators and adaptation methods can be linked respectively to adaptation specialists and adaptation strategies in [Smyth and Keane, 1998].

Adaptation process

An adaptation method consists of several adaptation operators. In order to adapt a source case, adaptation operators have to be selected and organised. If there is strategic knowledge available in the system, it can be used for this purpose. This process leads to the building of a similarity path and an associated adaptation path.

Definition 4.4 (Similarity path —$\mathtt{SP}$)

A similarity path from a problem $\mathtt{srce}$ to a problem $\mathtt{tgt}$ is a set of q triples $(\mathtt{pb}_{i-1}, \mathtt{r}_i, \mathtt{pb}_i)$ with:

- $\mathtt{pb}_i$: i^{th} problem of the path;
- $\mathtt{pb}_0 = \mathtt{srce}$;
- $\mathtt{pb}_q = \mathtt{tgt}$;
- $\mathtt{pb}_{i-1}\ \mathtt{r}_i\ \mathtt{pb}_i$ (for $i \in \{1, \ldots, q\}$);
- $\mathtt{r}_i$ is such that $(\mathtt{r}_i, \mathcal{A}_{\mathtt{r}_i})$ is an available adaptation operator.

$\mathcal{P}(\mathtt{srce}, \mathtt{tgt})$ denotes the set of similarity paths that can be built from $\mathtt{srce}$ to $\mathtt{tgt}$. If $\mathtt{srce} = \mathtt{pb}_0$ and $\mathtt{tgt} = \mathtt{pb}_q$, then this similarity path can be written:

$$\mathtt{pb}_0\ \mathtt{r}_1\ \mathtt{pb}_1\ \mathtt{r}_2\ \mathtt{pb}_2 \ldots \mathtt{pb}_{q-1}\ \mathtt{r}_q\ \mathtt{pb}_q$$

A similarity path is a path from $\mathtt{srce}$ to $\mathtt{tgt}$ in the problem space $\mathcal{L}_{\mathtt{pb}}$ structured by the set of relations denoted by $\mathtt{r}$. The similarity path have to be built in accordance with the adaptation operators available in the selected adaptation method. These adaptation operators are applicable to the source case and to all the intermediate problems generated from this source case. Figure 4.5 illustrates the notion of similarity path, an example in the Flatland domain will be given later (see example 4.2). The question of how to choose a suitable similarity path will be addressed in definition 4.8.

$$\begin{tikzcd} \mathtt{srce} \arrow[r, dash, "\mathtt{r}_1"] & \mathtt{pb}_1 \arrow[r, dash, "\mathtt{r}_2"] & \mathtt{pb}_2 \arrow[r, dash, "\mathtt{r}_3"] & \mathtt{tgt} \end{tikzcd}$$

Figure 4.5: A similarity path.

The building of the adaptation path is straightforward: the adaptation operators $\mathcal{A}_{\mathtt{r}_i}$ matching the relations $\mathtt{r}_i$ are selected and applied.

Definition 4.5 (Adaptation path —$\mathtt{AP}$)

The adaptation path $\mathtt{AP}$ associated to a similarity path $\mathtt{SP}$ is a set of q triples $(\widetilde{\mathtt{Sol}}(\mathtt{pb}_{i-1}), \mathcal{A}_{\mathtt{r}_i}, \widetilde{\mathtt{Sol}}(\mathtt{pb}_i))$ with:

- $\widetilde{\mathtt{Sol}}(\mathtt{pb}_0) = \mathtt{Sol}(\mathtt{srce})$;
- $\widetilde{\mathtt{Sol}}(\mathtt{pb}_i) = \mathcal{A}_{\mathtt{r}_i}(\mathtt{pb}_{i-1}, \widetilde{\mathtt{Sol}}(\mathtt{pb}_{i-1}), \mathtt{pb}_i)$;
- $\widetilde{\mathtt{Sol}}(\mathtt{pb}_q) = \widetilde{\mathtt{Sol}}(\mathtt{tgt})$.

Figure 4.6 presents the previous similarity path (see figure 4.5) and its associated adaptation path. The adaptation operators involved are selected among all the adaptation operators available in the chosen adaptation method associated to the retrieved source case. It must be remarked that the intermediate problems are *virtual* in the sense that they are introduced for the purpose of reasoning and are neither previously solved problems (i.e., source problems), nor the problem being currently solved (i.e., the target problem). The relation $\widetilde{\mathcal{H}}$ is a copy of the relation $\mathcal{H}$. Indeed, we make the hypothesis that this relation holds for the intermediate problems generated from the source problem.

$$\begin{tikzcd}
\mathtt{srce} \arrow[r, dash, "\mathtt{r}_1"] \arrow[d, "\mathcal{H}"'] & \mathtt{pb}_1 \arrow[r, dash, "\mathtt{r}_2"] \arrow[d, "\widetilde{\mathcal{H}}"] & \mathtt{pb}_2 \arrow[r, dash, "\mathtt{r}_3"] \arrow[d, "\widetilde{\mathcal{H}}"] & \mathtt{tgt} \arrow[d, "\widetilde{\mathcal{H}}"] \\
\mathtt{Sol}(\mathtt{srce}) \arrow[r, dash, "\mathcal{A}_{\mathtt{r}_1}"'] & \widetilde{\mathtt{Sol}}(\mathtt{pb}_1) \arrow[r, dash, "\mathcal{A}_{\mathtt{r}_2}"'] & \widetilde{\mathtt{Sol}}(\mathtt{pb}_2) \arrow[r, dash, "\mathcal{A}_{\mathtt{r}_3}"'] & \widetilde{\mathtt{Sol}}(\mathtt{tgt})
\end{tikzcd}$$

Figure 4.6: An adaptation path

Example 4.2 - Adaptation steps

In this example, the aim is to solve the target problem "black pentagon, white circle" (on the top right side of figure 4.7). For that, the systems retrieves the source case "white square, black octogon → black pentagon" (on the left hand of figure 4.7) and its associated adaptation method. They have been retrieved because they allows us to build a "good" similarity path between the source case and the target problem (the meaning of "good similarity path" will be discussed with definition 4.8). The building of the similarity path implies the use of adaptation operators that are all available in the retrieved adaptation method. It must be remarked that the adaptation operators contained in the retrieved adaptation method $\mathtt{AM}_{\mathtt{srce}}$ are applicable not only on the source case but also on the intermediate cases generated during the problem-solving process.

In this example, three adaptation operators (pairs $\mathtt{r}, \mathcal{A}_{\mathtt{r}}$) are used. The first one, $\mathtt{AO}_1$, says "when the second shape changes colour from black to white, the solution changes colour from black to white." The second one, $\mathtt{AO}_2$, says "when the first shape wins an edge, the solution wins an edge". The third one, $\mathtt{AO}_3$, says "when the second shape changes from any shape to a circle, the solution wins an edge".

Adaptation operator

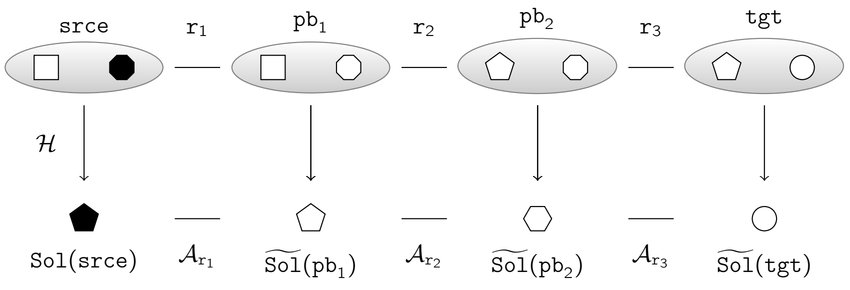


Figure 4.7: An adaptation in three steps.

The adaptation process is performed following one or several adaptation steps. An adaptation step allows the computation of the solution $\widetilde{\mathtt{Sol}}(\mathtt{pb}_i)$ for a problem $\mathtt{pb}_i$ given

$(\mathtt{pb}_{i-1}, \widetilde{\mathtt{Sol}}(\mathtt{pb}_{i-1}))$. Hence, an adaptation step corresponds to the application of an adaptation operator.

Definition 4.6 (Adaptation step)

Each triple $(\widetilde{\mathtt{Sol}}(\mathtt{pb}_{i-1}), \mathcal{A}_{\mathtt{r}_i}, \widetilde{\mathtt{Sol}}(\mathtt{pb}_i))$ is called an adaptation step.

In the previous example (example 4.2), the adaptation process is decomposed in three steps, each step corresponding to the application of an adaptation operator. Given a target problem and a source case, several similarity paths can be built. To select between all the possible similarity paths the "best one" with regard to the adaptation task, the notion of length of a similarity path has to be introduced. Before going further on this definition, the concept of "best similarity path" has to be explained. As the adaptation path is built after the similarity path, the best similarity path is the path that gives the best adaptation result, i.e., the path that minimises the probability of adaptation error. Therefore, the notion of length of a similarity path makes reference to the concept of adaptation error and estimated adaptation error.

Definition 4.7 (Adaptation error —e_r and its estimation —$\widetilde{\mathsf{e}_r}$)

Each adaptation operator $\mathtt{AO}_\mathtt{r}$ introduces a numerical error e_r, function of the problems $\mathtt{srce}$ and $\mathtt{tgt}$ related by $\mathtt{r}$: $\mathsf{e}_r(\mathtt{srce}, \mathtt{tgt}) \in \mathbb{R}_+$. The system is assumed to have an estimated value $\widetilde{\mathsf{e}_r}(\mathtt{srce}, \mathtt{tgt})$ of the adaptation error (the exact value of the adaptation error cannot be known by the system). Moreover, $\widetilde{\mathsf{e}_r}$ is assumed to have the following property: $\widetilde{\mathsf{e}_r}(\mathtt{srce}, \mathtt{tgt}) = 0$ if $\mathtt{srce} = \mathtt{tgt}$.

The estimation of the adaptation error is a value known by the system and used to estimate the "risk" of using the corresponding adaptation operator (it is a kind of weight associated with the adaptation operator).

Once the estimated value of the adaptation error is defined, the length of the similarity path is defined as the sum of the adaptation errors implied by the adaptation operators used in this path. Hence, the (dis)similarity measure (i.e., the notion of distance between problems) is related to the estimation of the adaptation error.

Definition 4.8 (Length of a similarity path —$\ell(\mathtt{SP})$)

$$\ell(\mathtt{SP}) = \sum_{i=1}^{q} \widetilde{\mathsf{e}_r}(\mathtt{pb}_{i-1}, \mathtt{pb}_i)$$

The length of a similarity path allows the notion of distance between problems to be defined: the distance between two problems is defined as the length of the shortest similarity path between the problems.[5]

[5]Technically, an inf should be used instead of a min: it is possible to find a series of similarity paths $(\mathtt{SP}_n)_n$ such that $\ell(\mathtt{SP}_n) > 0$ and $\lim\limits_{n\to\infty} \ell(\mathtt{SP}_n) = 0$. To avoid this theoretical problem, it is assumed that the number q of steps in a similarity path is bounded by a constant (eg, $q \leq 10^{100}$).

Definition 4.9 (Distance between problems —$\texttt{dist}(\texttt{srce}, \texttt{tgt})$)

$$\texttt{dist}(\texttt{srce}, \texttt{tgt}) = \min\{\ell(\texttt{SP}) \mid \ell(\texttt{SP}) \in \mathcal{P}(\texttt{srce}, \texttt{tgt})\}$$

It can be observed that the distance between problems is not symetrical, i.e., $\texttt{dist}(\texttt{srce}, \texttt{tgt})$ is not necessarily equal to $\texttt{dist}(\texttt{tgt}, \texttt{srce})$

Example 4.3 - Adaptation steps

Figure 4.8 shows a subset of similarity paths that can be built between srce and tgt using a given adaptation method. The problem part of the source case is shown on the left hand and the target problem is shown on the right hand. All the other problems are intermediate problems and are built by application of adaptation operators. The numerical values on the edges correspond to the estimation of the adaptation errors involved by the application of the adaptation operators. This figure shows that each similarity path has a different cost (sum of the estimated adaptation errors). According to definition 9, the distance between srce and tgt is the length of the shortest similarity path, i.e., in this example, 0.4.

Similarity paths

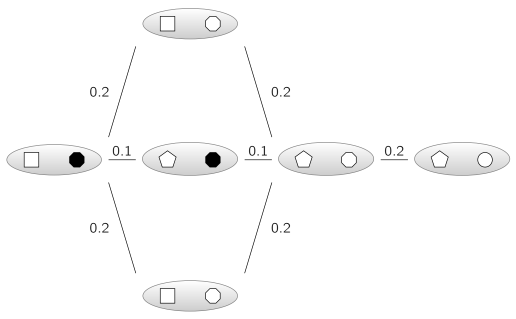


Figure 4.8: Length of a similarity path.

The computation of $\texttt{dist}(\texttt{srce}, \texttt{tgt})$ for each source case of the case base allows the retrieval of the “best case” among the cases of the case base.

4.1.3 IAKA: main principles

The idea behind the IAKA approach is to exploit failures to acquire cases and adaptation knowledge. In approximate reasoning, a failure occurs when the distance between the solution of the system and the “ideal solution” is too large. IAKA relies on the availability of an *expert* and an *oracle* who are able to say if a solution is satisfactory or not, to correct a non-satisfactory solution and to give adaptation operators for a case. The oracle is able to give an estimation of “how far” the candidate solution is from the real solution and to compare it to the expert’s *tolerance threshold* denoted by ε ($\varepsilon > 0$): if the distance is larger than ε, the solution is not satisfactory. Interactions with the expert and the oracle provide IAKA with methods to learn additional knowledge and to improve existing knowledge.

The occurrence of a failure means that a piece of knowledge that was used during adaptation has to be corrected or made precise. In the framework of IAKA, adaptation methods, adaptation operators and adaptation errors can be questioned. A main advantage of the IAKA approach is that the different pieces of knowledge are separated and tested independently thus enabling the faulty knowledge to be identified more easily. Indeed, when a solution is not satisfactory, the adaptation operators involved are tested one after the other. If a faulty adaptation operator is identified, it is corrected through interactions. The new piece of knowledge is added to the knowledge base of the system and a new CBR cycle is performed in order to find a better solution for the current problem. Adaptation operators are corrected and solution are re-adapted until a satisfactory solution is found.

Justification of the IAKA approach

The CBR inference is based on the following principle (see, eg, [Dubois et al., 1997]):

Similar problems have similar solutions. (CBR principle)

The similarity between problems is the knowledge of the retrieval step, often in the form of a similarity measure or a distance between problems. The similarity between solutions is linked with the adaptation: the higher the error involved by adaptation is, the less the solutions are similar.

Therefore, a failure in the CBR inference indicates:

(a) Either that $\mathtt{srce}$ and $\mathtt{tgt}$ are not (enough) similar;
(b) Or a failure in the CBR process.

The failure (a) can also be split into two sub-situations:

(a1) There is no source case similar to the target problem;
(a2) There is at least a source case $(\mathtt{srce}', \mathtt{Sol}(\mathtt{srce}')) \neq (\mathtt{srce}, \mathtt{Sol}(\mathtt{srce}))$ that is similar to $\mathtt{tgt}$ but it has not been retrieved.

Each of the failure of type (a1), (a2), and (b) leads to a knowledge acquisition from the oracle and with the expert

When a failure of type (a1) occurs, the expert may provide a new source case with its associated adaptation method, that is similar to the target problem (for instance a case $(\mathtt{tgt}, \mathtt{Sol}(\mathtt{tgt}))$ and an adaptation method $\mathtt{AM}_{\mathtt{tgt}}$).

When a failure of type (a2) occurs, this questions the similarity between problems that constitutes the retrieval knowledge: $(\mathtt{srce}, \mathtt{Sol}(\mathtt{srce}))$ is closer to $\mathtt{tgt}$ than $(\mathtt{srce}', \mathtt{Sol}(\mathtt{srce}'))$ and it should not be. With a similarity based on the estimated adaptation errors, the interactions with the expert should lead to a modification of these estimated errors.

When a failure of type (b) occurs, the similar problems $\mathtt{srce}$ and $\mathtt{tgt}$ have no similar solution. In other words, in a neighbourhood of $\mathtt{srce}$, the solution varies in an irregular manner. This situation can be interpreted with the notion of (dis)continuity of numerical functions $f : \mathbb{R}^n \to \mathbb{R}$. Indeed, if $\mathcal{L}_{\mathtt{pb}} = \mathbb{R}^n$, $\mathcal{L}_{\mathtt{sol}} = \mathbb{R}$, and $\mathtt{Sol}(\mathtt{pb})$ solves $\mathtt{pb}$ if $f(\mathtt{pb}) = \mathtt{Sol}(\mathtt{pb})$, then the continuity of f is defined intuitively with the CBR principle: if x_1 is close to x_2 then $f(x_1)$ is close to $f(x_2)$. A type (b) failure means that there is a discontinuity close to $\texttt{srce}$. The interactions with the expert may be useful to better locate the discontinuity points. It may occur that these discontinuity points involve a partition of the problem space in several places. For example, if $\mathcal{L}_{\text{pb}} = \mathbb{R}$ and 4 is a discontinuity point highlighted by the expert, then $\mathcal{L}_{\text{pb}}$ is partitioned in $\{]-\infty, 4[, \{4\},]4, +\infty[\}$. This implies that two problems of two different parts of this partition should never be considered as similar. With the previous example, 3.99 is dissimilar to 4.01. Therefore, the knowledge of this discontinuity point can be used as pieces of retrieval knowledge.

This justification of the IAKA approach based on the CBR principle and the proximity of this principle to the notion of continuity suggests it should be tested in domains where continuity is well-defined. The numerical functions constitute such domains and that is why it has been chosen for our experimentations and for the development of IAKA-NF (see section 4.2.

4.1.4 IAKA in the CBR process

The basics of the CBR process consist in finding a suitable source case and to adapt it to fit the requirements of the target problem. In most situations, solving the target problem $\texttt{tgt}$ amounts to follow these steps:

- *Retrieval* of a source case $(\texttt{srce}, \texttt{Sol}(\texttt{srce}))$ deemed to be similar to $\texttt{tgt}$.
- *Adaptation* of $\texttt{Sol}(\texttt{srce})$ using adaptation knowledge to provide a candidate solution $\widetilde{\texttt{Sol}}(\texttt{tgt})$ for $\texttt{tgt}$.
- *Test and repair* of the candidate solution and of the adaptation knowledge used.

The knowledge acquisition principles described in IAKA are applicable in CBR systems implementing, at least, the three previous steps. As the knowledge acquisition process is performed during the problem-solving process, it is advisable to understand the reasoning process. This section aims at describing the modelling of a problem-solving episode in CBR and at showing how knowledge acquisition is made possible through the process.

Retrieval

The retrieval process consists in selecting the best case of the case base by computing distance between problems (i.e., problem part of the case and problem part of the target problem) according to the notion of distance between problems introduced in definition 4.9. Hence, during the retrieval process, several similarity paths are built and the "best" one is selected. It must be remarked that the length of the considered similarity paths depends on the selected adaptation method.

According to the *adaptation guided retrieval* principle [Smyth and Keane, 1998], the retrieval process takes into account the adaptation knowledge. For instance, adaptation strategies, that are strategical knowledge that can be used to choose the operators and their order, are often applied to improve the retrieval process. Moreover, adaptation operators play important roles: their availability is a determining factor for the building of the similarity path and their associated adaptation error is involved in the computation of the distance between problems.

The result of the process is the retrieval of a case $(\texttt{srce}, \texttt{Sol}(\texttt{srce}))$ (together with its adaptation method, denoted by $\texttt{AM}_{\texttt{srce}}$) that minimises $\texttt{dist}(\texttt{srce}, \texttt{tgt})$.

Adaptation

The aim of the adaptation process is to provide a candidate solution $\widetilde{\mathtt{Sol}}(\mathtt{tgt})$ for $\mathtt{tgt}$.

In this modelling of the CBR process, performing adaptation simply consists in following the adaptation path that corresponds to the selected similarity path. Adaptation is performed in several steps, each step consisting in applying an adaptation operator. Each adaptation step leads to the building of an intermediate problem (that is a "virtual" problem).

At the end of the adaptation phase, a set of intermediate problems together with their candidate solutions is available in addition to the candidate solution for $\mathtt{tgt}$: $\widetilde{\mathtt{Sol}}(\mathtt{tgt})$. Each intermediate problem corresponds to the application of a specific adaptation knowledge and thus, ease the analysis of the knowledge unit in case of failure.

Test and repair

At the end of the adaptation step, the candidate solution proposed by the system has to be tested to check if it is satisfactory or not. The test is performed by the expert who can require help from the oracle. In practice, when the expert wants to test a solution, he asks the oracle how close the candidate solution is to the ideal solution. If the distance (given by the oracle) between the candidate solution and the ideal solution is larger than the expert tolerance threshold, that he judges it not satisfactory.

If the expert judges the solution satisfactory, the system can process to the memorisation step. If the solution is not satisfactory, it has to be corrected and the involved adaptation knowledge has to be revised. Adaptation knowledge involved in the computation of the solutions can be located at several levels:

- One or several adaptation operators can be incorrect;
- The adaptation method can be wrong (incorrect order of the adaptation steps, missing adaptation steps);
- Some of the problem descriptors should not have been ignored.

It must be remarked that adaptation knowledge is not "wrong" in itself, it might be correct in other situation but not applicable to the current problem. One specificity of IAKA is that when the adaptation operators are to be questioned, they are tested one after the other thus allowing to easily identify which operators are responsible for the failure.

4.2 IAKA-NF: A prototypical CBR engine

IAKA-NF is a prototypical CBR engine implementing the principles of IAKA in the application domain of the numerical functions ($f : \mathbb{R}^n \to \mathbb{R}$), hence the name: IAKA-Numerical Functions. The aim of the reasoning process in this prototype is to "solve problems by approximation", i.e., given n values $x_1, ..., and x_n$, the goal is to find an approximate value of $f(x_1, ..., x_n)$ by CBR.

One could think that IAKA-NF is a CBR system that has been built to provide a new approach for solving approximation problems, but this is not the case. IAKA-NF has been developed to experiment with some CBR principles and with some knowledge acquisition hypotheses. The numerical domain is only an easily accessible playground for such experimentations and that is the main reason why it has been chosen. We use mathematics to illustrate our purpose and we do not develop a CBR system to solve mathematical problems. To perform adaptation, IAKA-NF relies on an implementation of the differential adaptation strategy [Fuchs et al., 2000]. This section recalls the principles of the differential adaptation strategy, then it shows how this strategy is used to perform adaptation in IAKA-NF and how the IAKA model is implemented to support knowledge acquisition. The last part of the section introduces a working example to better understand the main principles of IAKA-NF.

4.2.1 Principles of differential adaptation

Differential adaptation strategy is a general strategy for adaptation in CBR inspired from the well-known notion of differential calculus. It has been firstly introduced in [Fuchs et al., 2000] and an extended description of the formalisation is available in [Fuchs et al., 2006a].

The differential calculus is the study of how functions change when their inputs change. In other words, it is the study of how a variation of a "variable" entails a variation on the result. Differential calculus entails the notions of "derivative" and of "differential". The derivative of a function at a chosen point describes the behaviour of the function near that point. Intuitively, we could say that differential represents a very small change in the value of a function under certain conditions.

If we take a closer look at some CBR specificities, the analogy with the differential calculus becomes obvious. As it is illustrated in figure 4.9, two main relations exist between a source case and a target problem:

- Relations between problems: *dissimilarities*;
- Relations between problem part and solution part in a case: *dependencies*.

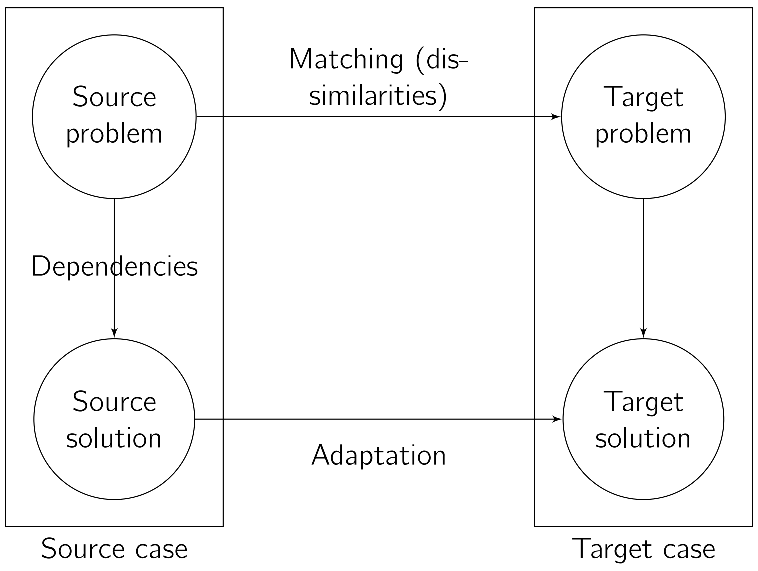


Figure 4.9: General adaptation process.

In the differential adaptation formalisation, problems and solutions are supposed to be represented by sets of descriptors. Then, dissimilarities represent variations between problem descriptors (for example, I have a Pavlova recipe for 6 servings and I want to prepare a Pavlova for 9 servings). Dependencies characterise links between problem descriptors and solution descriptors inside a case (for example, the number of servings has an influence on the number of eggs I will put in my Pavlova). Several problem descriptors may have an influence on a solution descriptor. To overcome this problem, the formalisation provides us with a set of operators that are used to combine "local" influences. A dependency not only indicates that a problem descriptor has an influence on a solution descriptor; in addition, it also contains a piece of knowledge indicating "how" the influence works. This piece of knowledge is called the influence function (in the above example, the influence function says that I have to multiply the number of eggs by a value, but the issue is to know how to determine this value).

The dissimilarities are variations on the problem (i.e., the differential) and influence functions can be seen as a kind of partial derivatives. In differential calculus, the formula that relates the differential value $dx = (dx_1, ..., dx_m)$ of $x \in \mathbb{R}^m$ to the differential value $dy = (dy_1, ..., dy_n)$ of $y \in \mathbb{R}^n$ relies on the use of partial derivatives and is given by:

$$dy_j = \sum_i \frac{\partial y_j}{\partial x_i} dx_i$$

To illustrate the analogy between differential adaptation strategy and differential calculus, let us continue with the Pavlova example. I have a recipe for a 6 servings Pavlova and I want to prepare a 9 servings Pavlova. The dissimilarity between the two problems is that there is 3 more servings in the target problem. The influence function (a kind of partial derivative) indicates that I have to multiply the number of eggs to reflect the variation (i.e., the dissimilarity) in the problem, then I will multiply this number by $9/6 = 1.5$.

To sum up, in the differential adaptation strategy, the adaptation process is decomposed in two main steps.

- Similarity assessment (or matching process): this step consists in studying the relationships between `srce` and `tgt` and aims at pointing out what makes them similar and, most of all, what makes them dissimilar. At the end of this step, we obtain a matching between a source case and a target problem, denoted by $\mathcal{M}(\texttt{srce}, \texttt{tgt})$;
- Solution modification (or adaptation process): this step consists in studying the relationships between the solution parts of the cases, i.e., between $\texttt{Sol}(\texttt{srce})$ and $\texttt{Sol}(\texttt{tgt})$. This study is supported by the knowledge of $\mathcal{M}(\texttt{srce}, \texttt{tgt})$.

Differential calculus is defined on $\mathbb{R}^m$ domains, which is not always suitable for CBR applications. One of the main contributions of the formalisation of the differential adaptation strategy is that is defines operators for similarity assessment and adaptation process to generalise the approach to other problem and solution spaces.

4.2.2 Differential adaptation strategy in IAKA-NF

IAKA-NF implements a numerical version of the differential adaptation strategy for experimental purposes. The notions proposed in the differential adaptation formalisation can be fitted to the notions we used in IAKA. Table 4.1 summaries these correspondences. In the following, we will use the IAKA notions and notations.

Differential adaptation strategy	IAKA	General CBR concept involved
Local variation (dissimilarity)	relation $\mathtt{r}$	Dissimilarity
Dependencies (partial derivatives)	Dependencies $\mathcal{H}$	Links between problems and solutions
Influence function	Adaptation operator $\mathtt{AO}$	Adaptation knowledge
Similarity assessment	Building of a similarity path	Retrieval process
Solution modification	Following the similarity path	Adaptation process

Table 4.1: Correspondences between IAKA and the differential adaptation strategy.

In order to better understand what is done in IAKA-NF, we are going to consider a very simple example of adaptation. Let f be a numerical function on $\mathbb{R}$ ($f : \mathbb{R} \to \mathbb{R}$). The CBR system does not have any knowledge about this function, it only knows "points" of the function. However, it relies on a case base, a case being a pair $(x, y) = (x, f(x))$. For each case, the system has also the knowledge of the local variations of f around x (i.e., a knowledge on the derivative). This is illustrated by figure 4.10(a).

When a new problem occurs, a CBR system retrieves a source case and adapt it. We focus here on adaptation. The value x is known and the goal is to find an approximation of this solution. This is performed by the differential adaptation of the retrieved source case, i.e., by applying the adaptation operator on the source case. Figure 4.10(b) illustrates this principle. The analogy with differential adaptation is obvious.

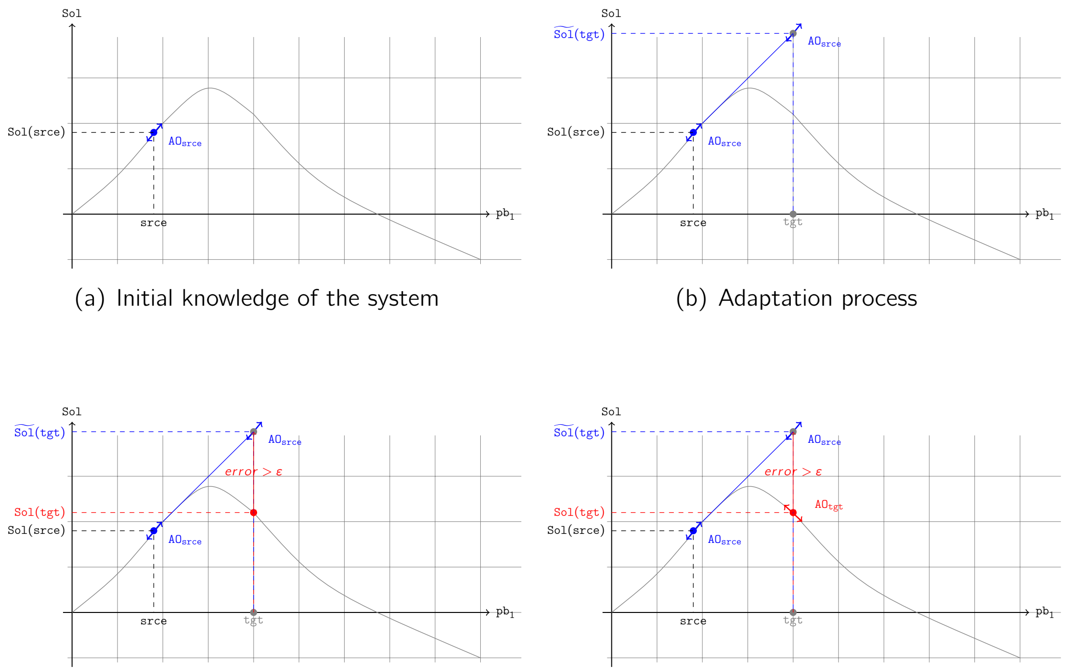
As it can be remarked in this figure, the result obtained is not good, the candidate solution y is very far from the "real" solution. This occurs either because the source case was not similar enough to the target problem or because the adaptation operator was not correct in this situation. In any case, a knowledge acquisition process has to be triggered to improve the system knowledge base and to avoid this failure to occur again.

4.2.3 Knowledge acquisition

In the general model defined in IAKA, knowledge acquisition is performed through interactions with the expert. In IAKA-NF, the expert is simulated by a computer program. He has two properties: he is able to give the correct solution and he is able to help the system to build a correct adaptation operator.

Interactions with the "virtual" expert are illustrated in the following figures. The knowledge acquisition process is triggered because the difference between the ideal solution and the candidate solution is larger than ε, the tolerance threshold of the expert. In figure 4.10(c), the expert gives the correct solution for the target problem. Then, in figure 4.10(d), he builds the corresponding adaptation operator.

The new case and its corresponding adaptation method (consisting of only one adaptation operator) is then added to the case base. This example is very simple and there is only one



(a) Initial knowledge of the system

(b) Adaptation process

(c) Failure detection

(d) Knowledge acquisition

Figure 4.10: Adaptation and knowledge acquisition in IAKA-NF.

adaptation operator to test. When a failure occurs, in case several operators are involved in the adaptation process, they have to be isolated and tested one by one on intermediate problems. Following the IAKA principle of decomposition of the adaptation in adaptation steps, IAKA-NF includes an algorithm for selecting and testing adaptation operators one by one.

Acquiring knowledge by reducing the scope of the cases

The "scope" of a case is the problem space where the case can be used as a source case for adaptation, i.e., where the case is "competent". This can be linked to the notions of *coverage* and *reachability* defined in [Smyth and McKenna, 1998]. Hence, the scope of a case defines the set of target problems that can be solved by adaptation of this case. A main issue is then to explain how to define this scope. Should it be related to the adaptation error? To the expert tolerance level?

An intuitive way to define this scope is to say that a case is competent while no other case of the case base is strictly more competent than it is. In such a conception, if there is only one case in the case base, its scope is the whole problem space. When a new case is added to the case base, the scope of the first case is automatically reduced: problems that are closer to the first case (with respect to a well chosen similarity measure) will be solved with it whereas other problems will be solved with the second case.

This principle is applied in IAKA-NF. The knowledge base is progressively improved by the addition of new cases together with their adaptation method. Each time a new case is added to the case base, the scope of its neighbour cases is reduced. This principle is illustrated in figure 4.11 in a very simple example (in more complex situation, the scope diagram would be a Voronoï diagram). This figure shows the scope of the cases before and after the acquisition of a new case.

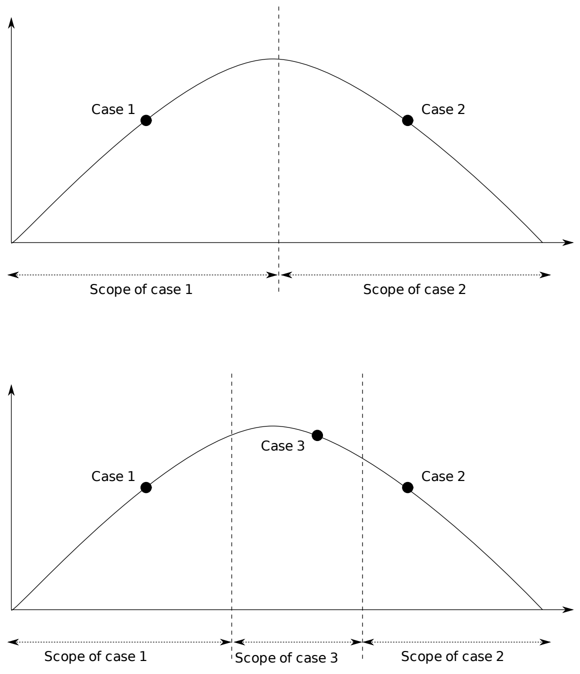


Figure 4.11: Reduction of the scope of the cases.

The algorithm that has to be implemented to reduce the scope of an adaptation method can be discussed. Indeed, in some situations, it is not relevant to divide equally the scope of the neighbours cases. This issue should be studied carefully.

Experimentations on the scope of the adaptation methods and on the way of handling them have been conducted with IAKA-NF, but they remain at an early stage. An investigation on the formalisation of this notion is still under process. In particular, we think that the notion of scope of a case should be related to the intuition of adaptation class, where an adaptation class is defined as a set of problem that can be adapted using the same adaptation method. We are convinced that this is an important research issue in addition to be an interesting prospect to IAKA. A proper formalisation of the notion of adaptation classes could provided us with useful tools to support better the adaptation process and most of all, the adaptation knowledge acquisition process.

This section has presented a very simple example as an introduction of how IAKA-NF works. One should notice that IAKA-NF is able to handle multidimensional problems in the field of numerical functions. Next section introduces a mono-dimensional concrete example to illustrate the basic principles of IAKA-NF before entering into details in the experimentation section.

4.2.4 Knowledge representation and reasoning process

In IAKA-NF, a problem is defined as a n-tuple of values and a solution is an approximation of the value of the function f for these values. The function f is called the "objective function". It is used to build the case base and to support the knowledge acquisition process, but it is not known by the system. Indeed, as the expert does (i.e., he tries out a solution and receives a feedback indicating if the solution works or not), in IAKA-NF, the expert can try out the solution on the objective function to check if it is correct or not. f_{CB} denotes the approximation of the function f by the case base. To each case is associated at least an adaptation method containing n adaptation operators. As we are in the numerical functions domain, adaptation operators are implemented by partial derivatives.

The retrieval is performed according to the distance defined in definition 4.9. The adaptation consists in applying the different adaptation operators to the retrieved adaptation method. The solution is obtained by adding to the solution of $\mathtt{srce}$ the variations involved by the different variables of the problem (calculated using the partial derivatives).

The knowledge acquisition process is performed according to the principle introduced before: a candidate solution produced by the system is always tested by the pair expert-oracle (simulated by the objective function f and the tolerance threshold ε). If the solution is not satisfactory, the involved adaptation operators are tested and corrected if needed, until a satisfactory solution is found. Then, the newly solved case $(\mathtt{tgt}, \mathtt{Sol}(\mathtt{tgt}))$ is added to the case base together with its adaptation method, given by the expert.

The following example illustrates the mechanism of IAKA-NF with a function $f_a : \mathbb{R} \to \mathbb{R}$. The example below also indicates how cases are defined.

Example 4.4 - A working example of IAKA-NF

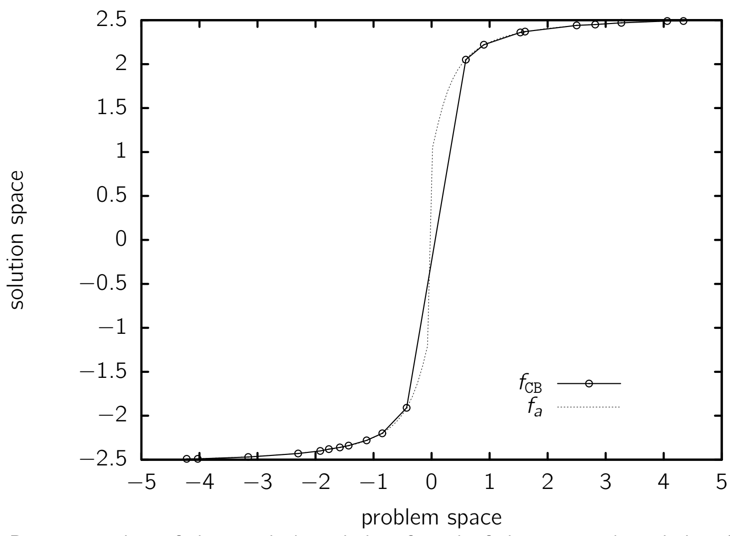
$$f_a : \mathbb{R} \to \mathbb{R} \qquad f_a(x) = \begin{cases} 1 + \mathrm{Arctan}(3x) & (\text{if } x \geq 0) \\ -1 + \mathrm{Arctan}(3x) & (\text{if } x < 0) \end{cases}$$

$$\mathtt{srce} = x^s \qquad \mathtt{tgt} = x^t$$

$$\mathtt{Sol}(\mathtt{srce}) = y^s \qquad \widetilde{\mathtt{Sol}}(tgt) = \widetilde{y^t}$$

Moreover, there is only one adaptation operator $\mathtt{AO}_r$ in the adaptation method $\mathtt{AM}_{\mathtt{srce}}$. It is defined such as $x^s \; \mathtt{r} \; x^t$ holds for any x^s and x^t, and $\widetilde{y^t} = \mathcal{A}_{\mathtt{r}}(x^s, y^s, x^t) = y^s + \frac{\partial y^s}{\partial x}(x^t - x^s)$.

Figure 4.12 illustrates the state of the system after the initialisation of the case base with 20 cases. The two dimensions are the problem space (x values) and the solution space (y values). f_a, represented with the dotted line, is the objective function (not known by the system, but represented for illustrative purpose). The small circles are the cases available in the case base and the plain line, denoted by f_{CB} is the approximation of the objective function provided by the CBR system given the current case base (the line is drawn by a simple approximation).



Representation of the oracle knowledge f_a and of the system knowledge f_{CB} for an initial case base of 20 cases (circles represent source cases).

Figure 4.12: A simple example on use of IAKA.

4.3 Experiments

Several experiments aiming at validating the IAKA principles have been conducted with IAKA-NF. Three of them are described and commented in this section. The first experiment has been chosen as an illustration and is presented to make clear how IAKA-NF works. The two other experiments are more complex, they illustrates the various aspects of the approach.

4.3.1 A simple example

This simple example is called the *M-problem* because of the shape of the function used to describe the domain (see figure 4.13). In the *M-domain*, the problem space is characterised by a single problem descriptor and the solution space is characterised by a single solution descriptor.

Cases. A source case `srce-case` is a pair (x, y) where x is the unique problem descriptor and y is the solution descriptor. To each source case is associated an adaptation method $\texttt{AM}_{\texttt{srce}}$ containing one (and only one) adaptation operator $\texttt{AO}_x$.

$$\texttt{srce-case} = (x, y)$$

Oracle. The oracle is denoted by $\mathcal{O}_{f_m}$. The oracle's knowledge can be simulated by the function $f_m(x)$ defined as:

$$f_m(x) = x + 10\sin(5x) + 7\cos(4x)$$

Figure 4.13 gives a graphical view of the oracle's knowledge about the *M-domain*. By the means of this function, the oracle knows the solution of any problem, but it must be remarked that he does not have any knowledge about the way to perform adaptation. In other words, he does not have any adaptation knowledge.

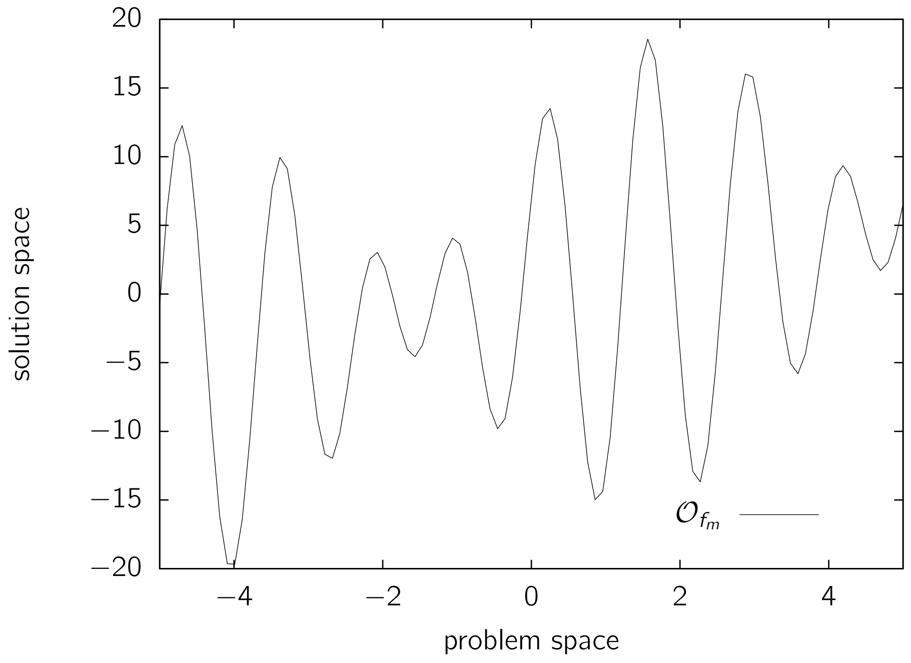


Figure 4.13: The oracle's knowledge on the *M-problem*.

It must be remarked as well that the function f_m is only known by the oracle, and not by the system.

Expert. For each problem, the expert is able to give an adaptation operator ($\mathtt{AO}_x$) when it is needed. How is it done in IAKA-NF? The virtual expert knows the formula of the partial derivatives of f_m and is able to compute its value for a given value of x. In this simple example, the partial derivative $\frac{df_m}{dx}(x)$ of f_m is:

$$\frac{df_m}{dx}(x) = 1 + 50\cos(5x) - 28\sin(4x)$$

It must be remarked that the expert could also use any other technique that allows him to approximate the partial derivative.

Experimental protocol

The experimental protocol defined here is quite simple. It can be decomposed in two main steps: building of the initial case base and utilisation of the system.

- Building of the initial case base.
    - The number n of cases to be generated is defined.
    - A domain for the x values is chosen.
    - The n values for x (problems parts) are generated by a random selection in the predefined domain.
    - For each x value, the oracle gives the corresponding solution thus leading to the definition of a source case.
    - For each source case, the expert gives a valid adaptation method (composed of an adaptation operator $\mathtt{AO}_x$).
- Utilisation of the system to solve a problem.
    - A problem is randomly generated (random selection of a value x in the domain).
    - A candidate solution is computed by the system.
    - The candidate solution is evaluated by the oracle.
    - If it is required, the solution and the involved adaptation operator are corrected.

The evaluation of the candidate solution by the oracle is performed through a comparison with the oracle tolerance threshold ε. The tolerance threshold is defined *a priori*. When a candidate solution $\widetilde{\mathtt{Sol}}(\mathtt{tgt})$ is proposed by the system, the oracle compares this solution with the correct solution $\mathtt{Sol}(\mathtt{tgt})$ (he knows it). If the difference $\left|\widetilde{\mathtt{Sol}}(\mathtt{tgt}) - \mathtt{Sol}(\mathtt{tgt})\right| > \varepsilon$ is higher than the tolerance threshold, then the solution is said to be unsatisfactory.

The pseudo-algorithm below (algorithm) illustrates the utilisation of the system to solve a problem.

Algorithm 1 A simple interaction in IAKA-NF.

```
1: A problem $tgt$ is randomly generated
2: The system returns the candidate solution $\widetilde{\mathtt{Sol}}(\mathtt{tgt})$
3: The oracle tests $\widetilde{\mathtt{Sol}}(\mathtt{tgt})$ (thanks to the oracle)
4: if the solution is satisfactory then
5:     The adaptation method is linked to the case
6:     The case is added to the case base
7: else
8:     The expert corrects the solution
9:     The expert corrects the adaptation method
10:    The adaptation method is linked to the case
11:    The corrected case is added to the case base
12: end if
```

It must be remarked that because the adaptation knowledge involved is also very simple, the repair step is very simple. Indeed, in the *M-domain*, an adaptation method contains only one adaptation operator.

System's use case. In order to illustrate how the system works, we are going to consider a simple example. In this example, we assume that the systems knows the case $\mathtt{srce\text{-}case}_1 = (x_1, y_1) = (0, 7)$ and the adaptation operator $\mathtt{AO}x_1 = 1 + 50\cos(5x_1) - 28\sin(4x_1) = 51$ (because $x_1 = 0$). In order to find a solution for the target problem $\mathtt{tgt} = (x_t, y_t)$ (with

$x_t = 0.1$), the system performs the following calculation: $y_t = y_1 + \mathtt{AO}x_1(x_t - x_1) = 7 + 51 * (0.1 - 0) = 12.1$. Thus, the system's candidate solution is $y_t = 12.1$. As for the oracle, he knows that the real solution is $y_t = 11.34$. There is a difference of 0.76 (in absolute value) between the two solutions.

If the tolerance threshold of the expert is $\varepsilon = 0.5$, then the candidate solution is not satisfactory (because $0.76 > 0.5$). The adaptation operator has to be corrected by the expert. As it was said before, several methods can be implemented to permit the correction of an adaptation operator (computation by the system after interactions with the expert, proposition of candidate adaptation operators and selection by the expert, etc.). In IAKA-NF, the expert simply gives the correct adaptation operator $\mathtt{AO}x_t$ to the system. This adaptation operator is linked to the newly solved case. The adaptation operator $\mathtt{AO}x_1$ linked to the case $\mathtt{srce\text{-}case}_1$ is not altered (it was not incorrect, only not appropriate in this situation).

Parameters

Several parameters have to be taken into account when performing experiments with IAKA-NF. Among them, we can cite:

- Size of the initial case base
- Tolerance threshold of the expert
- Memorisation of the adaptation knowledge
- Precision of the adaptation knowledge

Several of these parameters are going to be discussed in the other experiments.

4.3.2 Influence of the tolerance threshold of the oracle

The aim of this experiment is to analyse the impact of ε (the tolerance threshold of the expert) on the quality of the results produced by the system. The hypothesis is that the smaller ε is, the better the results are (for a constant number of solved problems).

This experiment is conducted in the *M-domain* described in section 4.3.1. In order to conduct this experiment, an initial knowledge base is built; it is composed of 20 cases randomly generated (and solved by the oracle $\mathcal{O}_{f_m}$) and their associated adaptation methods (provided by the expert).

$\mathcal{O}_{f_m}$ knowledge: f_m	$f_m : \mathbb{R} \to \mathbb{R}$
Tolerence threshold: ε	$f_m(x) = x + 10\sin(5x) + 7\cos(4x)$

Moreover, 70 target problems are randomly generated. The same initial knowledge base and set of problems are used for all the tests in this experiment.

On this experiment, two systems are run in parallel: the control system and the IAKA-NF system. The goal is to solve the 70 problems of the set of problems. In both systems, problems are solved according to the IAKA approach (test and repair of the knowledge of the system). The difference is that solved cases are not added to the case base in the control system whereas they are in the IAKA-NF system. The results presented in the following are obtained by comparison, for each case, of the results of the two systems.

The purpose of the experiment is to make ε vary, thus the experimental protocol described above is made 10 times with 10 different values for ε. For each experiment, we compare, for each case, the difference between the error made by the control system and by the IAKA-NF system.

Two statistical tests can be performed on the gathered data: the *Z-test* [Kendall and Stuart, 1969] and the *Wilcoxon test* [Wilcoxon, 1945] to measure the efficiency of the knowledge acquisition process. The value ρ, determined in each test, is the probability of obtaining the same results in a system performing knowledge acquisition as in a system without knowledge acquisition. For a IAKA-NF system, the smaller ρ is, the lower the chances of obtaining such results with the control system are. In other words, the smaller ρ is, the better the system is. The Z-test is a parametric test for two paired samples. As there is no guarantee that the initial distribution of cases and problems follows a normal law, the Wilcoxon test, a non-parametric test for two paired samples, is used to confirm the results of the first test. Figure 4.14 shows a graphical interpretation of the results of the Wilcoxon test (the results of the Z-test are similar).

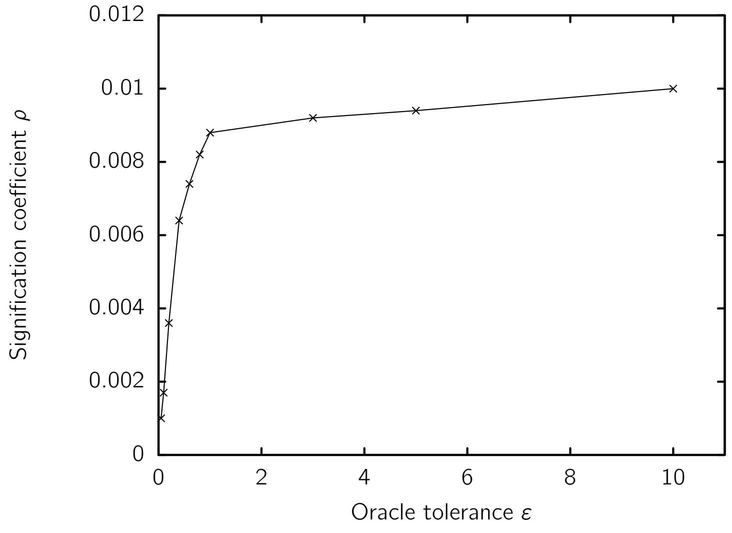


Figure 4.14: Evolution of the value of ρ in function of ε for the Wilcoxon test.

On the charts, we observe that the smaller ε is, the smaller ρ is, i.e., the more efficient the system is. A significant difference ($\rho < 0.01$ i.e., 1%) in terms of reduction of the size of the error is achieved when $\varepsilon = 10$ (which is a high value in this domain). The conclusion is that the higher the tolerance threshold of the oracle is, the bigger the probability for the system to make a mistake is, which confirms the hypothesis of this experiment. Similar tests have been performed with problems of two and three variables, giving similar results.

4.3.3 Impact of a discontinuity on the CBR process

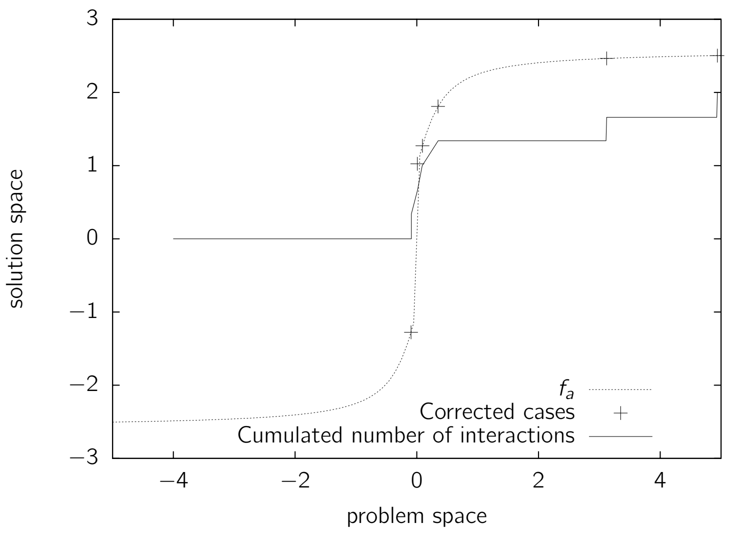
The aim of this experiment is to analyse the behaviour of a CBR system solving problems by approximation when there is a discontinuity in the domain. This experiment is motivated by the observation (b) discussed in section 4.1.3. The hypothesis is that more interactions with the oracle are needed when a problem is in the neighbourhood of a discontinuity.

The ArcTan domain

This experiment has been conducted in the *ArcTan* domain introduced in section 4.2. The f_a function is recalled hereafter.

$$\begin{array}{rl} \mathcal{O}_{f_a} \text{ knowledge: } f_a & \quad f_a : \mathbb{R} \rightarrow \mathbb{R} \\ \text{Tolerence threshold: } \varepsilon & \quad f_a(x) = \begin{cases} 1 + \text{Arctan}(3x) & (\text{if } x \geq 0) \\ -1 + \text{Arctan}(3x) & (\text{if } x < 0) \end{cases} \end{array}$$

As for the previous experiment, an initial knowledge base of 20 cases randomly generated is built by the oracle $\mathcal{O}_{f_a}$, and 70 target problems are also randomly generated. The experiment consists in solving the 70 target problems with IAKA-NF. The results are processed to count the number of problem-solving episodes that have triggered a failure and have required a correction from the oracle. As an example, figure 4.15 shows a graphical interpretation of the result of an experiment conducted with a tolerance threshold $\varepsilon = 0.2$.



The dotted line represents the function to approximate, the crosses are the solved cases that have required a correction from the oracle and the plain line represents the accumulation of the number of interactions with the oracle.

Figure 4.15: Distribution of the corrected cases around a discontinuity (with $\varepsilon = 0.2$).

This experiment has been conducted several times with different values for ε (but still with the same initial knowledge base and the same series of problems). Table 4.2 gives the results of these experiments. For each studied value of ε, the total number of corrected cases is given. It is compared with the number of corrected cases "around" the discontinuity.[6]

[6]The interval "around the discontinuity" is determined manually before the experiment.

Value of ε	0.05	0.1	0.2	0.5	1.0	1.5	2.0	5.0	10
Number of corrected cases	20	13	6	5	3	3	2	0	0
Number of corrected cases around the discontinuity	16	13	5	4	3	3	2	0	0

Table 4.2: Number of corrected cases and number of corrected cases around the discontinuity in function of the tolerance of the oracle.

Empirical results show that the number of cases learned around a discontinuity grows while the oracle tolerance threshold decreases. This tends to confirm the initial hypothesis of this experiment.

The Hat domain

The same experiment was also conducted in the *Hat* domain with another function f_{ht} involving two problem variables.

$\mathcal{O}_{f_{ht}}$ knowledge: f_{ht}

Tolerence threshold: ε

$$f_{ht} : \mathbb{R}^2 \to \mathbb{R}$$
$$f_{ht}(x,y) = \begin{cases} -3 - g(x,y) & (\text{if } x^2 + y^2 \leq 4) \\ -g(x,y) & (\text{if } x^2 + y^2 > 4) \end{cases}$$
$$g(x,y) = \sin\sqrt{x^2+y^2} + \frac{x}{7}$$

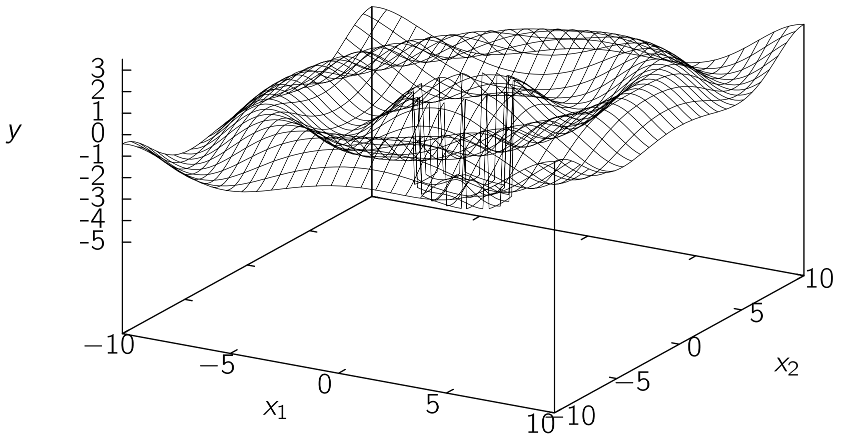
For two-dimensional problems, the results and the conclusions are similar. Figure 4.16 illustrates the conclusion. In this example, the oracle is $\mathcal{O}_{f_{ht}}$, $\varepsilon = 1.0$ and 20.000 problems are solved. A remarkable fact is that only 149 cases had to be corrected by the oracle, 113 of which during the first 1000 solved problems.

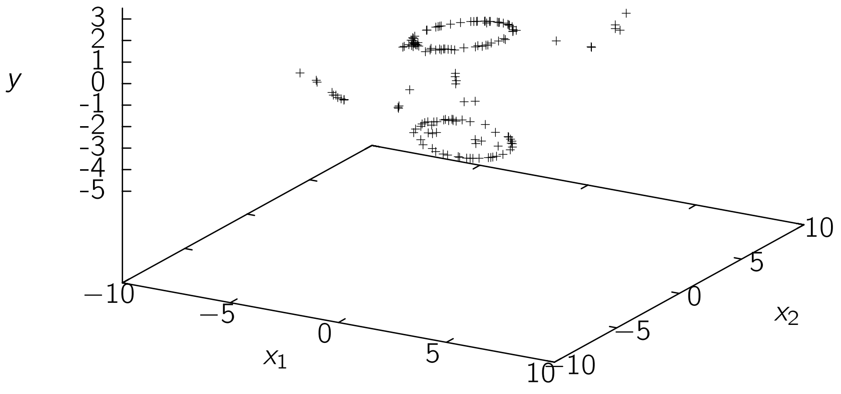
4.3.4 Future experiments

IAKA-NF has been designed as a test-bed application for experiments on knowledge acquisition in CBR. This section aims at giving an overview of the experiments we plan to conduct in future work.

Three main directions are to be considered:

- Study of the advantage of the adaptation decomposition on the efficiency of the knowledge acquisition process (in particular in domains with a high number of problem descriptors), i.e., does the identification and isolation of faulty adaptation operators facilitate the acquisition process?
- Study of the convergence of the system towards the objective function depending on the experimental conditions, i.e., how fast and how "well" does the system improve itself under different conditions (diminution of the error)?





The first figure represents the oracle knowledge (it is a "Mexican hat" where the upper part has been "pushed" to the bottom). The second figure on the right shows the cases learned by the system (after correction by the oracle): a high proportion of cases (80%) are acquired around the discontinuity of the function.

Figure 4.16: Acquisition of cases around a discontinuity.

- Study of the behaviour around discontinuities, i.e., how does the system behave around discontinuities, does the number of acquired cases increase faster than anywhere else?

Among the parameters to make vary are:

- Expert tolerance threshold ε;
- Size of the problem space (number of case descriptors);
- Size of the initial case base.

These experiments can be performed with several numerical functions having different properties. Table 4.3 sums up the different types of functions we would like to experiment with. For each type of function, an example function that could be used is given and the priorities for the experiments are indicated.[7]

Type of function	Example	A	C	D
C^∞ on $\mathbb{R}^n$	exponential function	++	+	
First order discontinuities	$f : \mathbb{R} \to \mathbb{R}$ $f(x) = x - \lfloor x \rfloor$		++	+
Second order discontinuities	$f : \mathbb{R} \to \mathbb{R}$ $f(0) = 0$ and $f(x) = \sin(1/x)$ $x > 0$		+	++

This table indicates our experiments priorities ("++" indicates a high priority). "A" stands for "Adaptation decomposition", "C" stands for "study of the Convergence" and "D" stands for "Discontinuities".

Table 4.3: Future experiments to be carried out with IAKA.

4.4 Future work and discussion

4.4.1 Related work

IAKA is different from off-line approaches in that the knowledge, coming from "the external world" is acquired incrementally and not inferred from existing containers. Off-line approaches generate a large amount of knowledge at once, leading to a significant work for the domain expert to interpret the results. In IAKA, the gradual acquisition alleviates this effort. Several other perform interactive knowledge acquisition, some of them are described hereafter.

Knowledge acquisition in CHEF

CHEF is a case-based planning application. According to the definition given in [Riesbeck and Schank, 1989], planning from cases means remembering failures so that they can be avoided and remembering successes so that they can be reused and remembering repairs so that they can be reapplied. In CHEF, past planning experiences are organised in an

[7] C^∞ on $\mathbb{R}^n$ represents all the functions indefinitely differentiable on $\mathbb{R}^n$.

episodic memory by two sort of indexes: goals to be satisfied and failures to be avoided. A planning failure occurs when a plan does not satisfy some goals that is was designed to deal with. Hence, to avoid failures, CHEF has to anticipate potential goal interaction problems. But this anticipation is only possible if the failure has been seen before. This entails that the case-based planner must have the ability to identify and repair faulty plans that have failed due to unwanted interactions between goals.

CHEF uses past failures to anticipate problems and past solutions as a basis for future solutions. When a failure occurs, it reacts by repairing both the faulty plan and its own faulty knowledge base that has been used to build the faulty plan. Hence, in this system, learning from planning implies:

- Learning new plans that avoid problems;
- Learning the features that help predicting the problems;
- Learning the repairs that have to be made if those problems arise again.

As they both learn from failures, CHEF and IAKA share several principles. However, CHEF differs from IAKA in that it exploits its own knowledge to explain failures and to avoid them in further reasoning.

Correctness and completeness of the retrieval process

Our work has been influenced by some ideas described in [Lieber and Napoli, 1998]. In this paper, the authors discuss of the concepts of correctness and completeness of the retrieval process with respect to the adaptation stage of the CBR. This work is conducted in the framework of the RESYN/CBR application, a case-based planner dedicated to organic chemistry synthesis.

The authors define the properties of correctness and completeness as follows. Retrieval is correct if the retrieved case is adaptable in order to solve the target problem. Retrieval is complete if every case in the case base which is adaptable to solve the target problem can be retrieved.

These properties are illustrated by two algorithm. The first one, a rule-based retrieval algorithms, shows complexity limitations. The second one, however, is more efficient. It is based on the concept of hierarchical classification.

This work can be related to the one proposed by Bergmann on transformational adaptation, which is presented hereafter.

Formal model of transformational adaptation

The IAKA approach can be bridged to the formal model of transformational adaptation presented in [Bergmann and Wilke, 1998]. In this paper, the authors advocate that, for pure case retrieval systems, there is a good theory and mathematical formalisation focusing on similarity measures, preference relations and related properties of the similarity measures and retrieval algorithm like correctness and completeness; but such a formalisation is still missing in systems including adaptation. Hence, the paper proposes a step into the direction of the building of a general formal adaptation framework.

The general model is mainly based on the notion of quality of a solution to a given problem (quality is meant in a general sense and can also denote some kind of appropriateness, utility or degree of correctness). A quality function, denoted by Q, is defined and includes the knowledge about the differences on the problems. The notion of quality can be linked to the notion of *outcome* defined by Kolodner [Kolodner, 1993]: it is a a feedback from the real world when trying the solution to the problem. In IAKA, this feedback is visible through interactions with the expert.

Adaptation knowledge is defined in terms of functions transforming cases. These adaptation functions come in the form of *adaptation operators* that transform cases into successor cases. This definition of an adaptation operator is similar to the one we use here. Given a problem and a solution, the quality function returns a quality value, i.e., the quality of the solution with respect to the problem. Hence, for a given target problem, the solution is chosen in the case base depending on the quality value (the highest the quality value is, the better the solution is).

As the function Q is supposed to contain the whole specification of the problem-solving task, the adaptation can be considered as a special instance of optimisation problem. But in real situation, this is not that simple because the function Q is really hard to formalise. The notion of quality gives a semantics for adaptation knowledge and defines concepts as soundness, correctness and completeness. Soundness is defined for cases and for adaptation operators (with respect to Q). Correctness of a CBR system ensures that the solution is always optimal and completeness ensures that a CBR system always returns a solution when it is possible (i.e., when it has the necessary knowledge for it). In theory, it is advisable for CBR system to guarantee correctness and completeness, but in real world application, it is hard to do.

An ongoing work on the formalisation of the notions of correctness and completeness in IAKA is conducted but remains at the state of intuition. The idea we want to develop is based on the notion of "adaptation class" that defines a set of problems that can be adapted using the same adaptation method and of scope of an adaptation method. The intuition is that by defining properly adaptation classes we can build a model of coverage of the problem space based, not on the cases properties such as in [Smyth and McKenna, 1998] (see next paragraph) but on the "adaptatbiliy" of the cases. Such a model will provides us with tools for defining adaptation competencies of the system and to evaluate adaptation performances.

Competence of a case

In [Smyth and McKenna, 1998], the authors define the notion of "competence" of a case-based system. The competence of a case is the range of problems it can solve. According to the authors, the competence depends on the cases available in the case-based but the relationship between cases and overall competence is difficult to define, in particular because this relationship is complex and because cases play different roles (some of them are critical whereas others are redundant). Hence, the authors propose a model of case competence taking into account the different status of the cases.

This notion of competence can be related to the notion of adaptation class that we have introduced earlier in this chapter. This work is still under investigation

4.4.2 Future work

In this chapter, several experiments have been conducted with IAKA-NF, a prototype implementing the principles of IAKA. A lot of work remains to be done on the formalisation of the IAKA principles and on the development of IAKA-NF.

As it was shown in section 4.3.4, we plan to conduct several experiments with the current prototype to test several hypotheses on knowledge acquisition in CBR. But the major prospect for this part of the work is to apply the concepts to real world application domains and to evaluate IAKA pros and cons more thoroughly in this context. Indeed, although the current experimentations had led to significant and encouraging results, they have been obtained with a virtual expert and thus, they cannot be considered as an obvious proof of the validity of the approach in real conditions. Two issues are then raised. The first issue is to find a suitable application domain, possibly with an existing CBR system in use, and to make sure that domain experts will be available to participate to the experiments. The second issue is to implement the IAKA principles in the target application, or to develop a full application from scratch if needed. IAKA principles being quite generic, this should not be a major problem.

Finally, we plan to investigate the possibilities of combination of IAKA with other knowledge acquisition approaches to make a step towards a general framework for knowledge acquisition embracing all the facets of the process: initial acquisition, interactive acquisition, correction of existing knowledge, revision of knowledge, case base and knowledge base maintenance, etc. A first step into this direction would be to combine the IAKA approach with the FRAKAS approach (described in the next chapter) and to observe problems arising when mixing several knowledge acquisition approaches.

4.4.3 Synthesis

The table 4.4 presents a synthesis of the specificities of IAKA and link them with the main FIKA principles.

4.5 Conclusion

This chapter has described IAKA, an approach for on-line acquisition and learning of cases and adaptation knowledge based on interactions. The approach has been implemented in a system called IAKA-NF where interactions occurs between the systems and a pair oracle-expert. Both the expert and the oracle are simulated by a program.

IAKA has been designed using the idea of a unified view of the knowledge involved in the CBR process. The failures of the CBR inference trigger a process that allows the knowledge base (cases and adaptation methods) to be repaired. The decomposition of the adaptation process into several steps makes the identification of the knowledge involved in the failure, such as adaptation knowledge, easier.

IAKA provides us with a framework to model knowledge learning in CBR systems when CBR is viewed as a kind of approximate reasoning. Another viewpoint is that of uncertain reasoning. A future work direction aims at generalising the IAKA approach and its justification so that it considers both viewpoints.

Several experiments have been conducted with IAKA-NF and tests show that IAKA opportunistic knowledge acquisition improves the accuracy of the CBR system in the vicinity of

IAKA, a model based on FIKA	
Description	Principles
Knowledge intensive	The knowledge comes from the virtual expert and is gathered during interactions, in the form of cases and adaptation operators.
Interactive	Interactions with the experts occurs systematically after each problem-solving episode. If a failure occurs, all the involved adaptation operators are tested, one by one, by the expert. Each test involves an interaction. The expert can follow the reasoning process step by step if needed.
Opportunistic	Failures trigger the knowledge acquisition process. Failures are identified by the *oracle* but are processed by the expert trough interactions.

Table 4.4: IAKA, a model based on FIKA.

the place where failures have occurred. They also show that this acquisition ceases to be efficient around discontinuity points, where the CBR principle is violated.

Although it has been tested, the IAKA approach remains to be compared with a real-world application, using an expert evolving in real conditions instead of a virtual expert and a virtual oracle. As discussed in section 4.4, IAKA should inter-operate with other knowledge acquisition/extraction/learning approaches. Most of the time, these approaches are supposed to be applicable to different phases of the CBR, with different goals and with different knowledge sources. However, IAKA adopts a unified view of the CBR process and its knowledge. Therefore, more work must be done to connect the various approaches in a more general framework. For instance, a future work is to elaborate a strategy that focuses on the type of faulty knowledge (adaptation knowledge, strategic knowledge, domain knowledge, etc.) to trigger an appropriate acquisition method. Although this is a long-term future work, this is an important issue in the field.

Next chapter is dedicated to the presentation of FRAKAS, another approach for interactive knowledge acquisition in systems producing solutions that are consistent with the domain knowledge. The discussion at the end of this next chapter goes back on the possible interactions between FRAKAS and IAKA.

En essayant continuellement, on finit par réussir. Donc : plus ca rate, plus on a de chances que ca marche.

Devise Shadok

5 FRAKAS, FAILURE ANALYSIS FOR DOMAIN KNOWLEDGE ACQUISITION

This chapter presents the FRAKAS *approach for interactive domain knowledge acquisition in* CBR *systems. The approach is based on the exploitation of adaptation failures. The* CBR *system is supposed to produce propositions of solutions that are consistent with the domain knowledge but that may be inconsistent with the oracle knowledge. Such an inconsistency constitutes a failure. Because of an interactive analysis of the failure, pieces of knowledge are acquired and contribute to improve the system domain knowledge. In the context of this work, we have developed* FRAKAS-PL*, a prototype implementing the* FRAKAS *principles with a knowledge representation in propositional logic. In* FRAKAS-PL*, the adaptation is performed using the conservative adaptation principle, which is inspired by the revision theory, and the revision operator plays the role of adaptation operator (as defined in* IAKA*). To establish the usability of the* FRAKAS *principles, the application* FRAKAS-PL(ONCO) *has been built using the* FRAKAS-PL *prototype and exploiting a knowledge base from a specific domain, namely breast cancer treatment. The chapter ends with a discussion on the research issues around* FRAKAS*, a comparison between* FRAKAS *and other approaches of knowledge acquisition and a conclusion discussing the role that research on* FRAKAS *played in the elaboration of the* FIKA *principles.*

Related publications:
[Cordier et al., 2007c], [Cordier et al., 2007e], [Cordier et al., 2007a]

Main related work:
[Lieber, 2007b], [Lieber, 2006], [Alchourrón et al., 1985]

Keywords:
Domain knowledge acquisition, conservative adaptation, adaptation with a revision operator.

Contents

Contributions of this chapter

- A formalisation of a knowledge acquisition process which extends FIKA, focusing on domain knowledge.
- An implementation of the conservative adaptation strategy in a prototype.
- Development of a prototype application (FRAKAS-PL) and experiments with it.

By storing solved problems, CBR allows solution hypotheses (i.e. propositions of solutions) to be obtained, even in weak or incomplete theory domains. These solution hypotheses are called here *propositions*. Meanwhile, these propositions may be not appropriate because of a lack of sufficient knowledge, thus leading to "reasoning failures". Because most of the CBR systems operate in weak theory domains, it is often impossible to provide them with all the knowledge needed to solve all the problems they are designed to solve. Nevertheless, it is often possible to cope with this issue by providing them with mechanisms allowing them progressively to learn new pieces of knowledge. These mechanisms allow the systems to improve over time, as the additional knowledge gained facilitates solving more problems. Indeed, the more knowledge the systems acquire, the more competent they are.

In several CBR systems, domain knowledge is used to control the system inferences and, in the case of decision support applications, to avoid wrong decisions. Thus, the more correct and accurate the domain knowledge is, the better the CBR inferences will be. A part of the domain knowledge is formalised *a priori* by knowledge engineers, but another part has to be acquired as the system is used.

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.

The processing of a proposition involves both the expert and the oracle (as they were defined in chapter 3). A proposition produced by the system is first presented to the oracle who decides if it is satisfactory or not. If a proposition is not satisfactory, the expert has to help the system to correct it. Hence, 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.

This chapter is organised as follows. Section 5.1 introduces FRAKAS. Notions, notations and hypotheses are detailed, and the way FRAKAS handles failures is explained. The roles of the expert and the oracle are also discussed. The implementation of FRAKAS in the prototype FRAKAS-PL is explained in section 5.2. In particular, the knowledge representation formalism and the principles of conservative adaptation are described. Section 5.3 illustrates the use of FRAKAS-PL in a particular application: FRAKAS-PL(ONCO) which focuses on domain knowledge acquisition in the field of breast cancer. Section 5.4 discusses this work and draws some prospects for FRAKAS. This section also compares FRAKAS with close knowledge acquisition techniques. Finally, section 5.5 concludes this chapter.

5.1 FailuRe Analysis for domain Knowledge AcquiSition

FRAKAS stands for FailuRe Analysis for domain Knowledge AcquiSition.[8] In [Lieber, 2007b], Lieber proposed a new approach of the adaptation process: the *conservative adaptation*. The main idea of the conservative adaptation is to perform minimal changes to a solution while staying consistent with the domain knowledge (see section 5.2.2 for more details about conservative adaptation). This principle, relying strongly on the manipulation of knowledge bases, was a tempting field to try out a knowledge acquisition approach. The combination of the conservative adaptation idea with the FIKA approach led to the definition of FRAKAS principles and to the design of the prototype FRAKAS-PL. FRAKAS-PL was first rapidly tried out in the cookery field and was then further developed with a restricted data set coming from the oncology domain.

5.1.1 Knowledge acquisition in FRAKAS

Conservative adaptation relies on the exploitation of domain knowledge to perform adaptation. In consequence, FRAKAS focuses on domain knowledge acquisition. To ensure that a learning process is possible, several hypotheses about the CBR system and the role played by the human being interacting with the system have to be made.

Remark 5.1

In the remaining of this chapter, a candidate solution, i.e. a solution proposed by the system, will be called "a proposition".

Oracle and expert. As it was explained in chapter 3, the notion of *oracle* is a metaphor to explain that there is "a way" to know if a proposition is valid or not. In practice, this could be done by testing the proposition in real world conditions or by checking its consistency by any other conceivable means. The expert is then able to give the system the knowledge it needs to correct an inconsistent proposition. When speaking of the expert, we implicitly make the assumption that he does not contradict himself (otherwise, he would introduce inconsistent knowledge in the system and would put in doubt the ability of the system to produce consistent propositions). This assumption may seem strong, but it only means that if the expert does not "know" how to correct the proposition, he has to use other means to learn how to do it. In other words, the expert has to learn what he needs before being able to help the system. In a sense, this amounts to saying that when a reasoning failure occurs, the pair "expert-system" is a co-evolving learning system.

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 solution 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

[8]This project has been conducted in collaboration with Jean Lieber. He has been closely associated with this research during his visits to the LIRIS and most of the results presented here come from this close collaboration.

knowledge if the proposition is not satisfactory, which amounts to highlighting the parts of the proposition that are not correct.

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.

The following sections are dedicated to the detailed descriptions of the underlying principles of FRAKAS.

5.1.2 Cases and knowledge representation for FRAKAS

In the following, it is assumed that a common knowledge representation formalism is used for all the pieces of knowledge. The choice of the knowledge representation formalism is a matter of implementation and does not impact the definition of the principles of FRAKAS. We assume that the oracle is, at any time, able to say if a solution is satisfactory or not. The expert is able to highlight inconsistencies without making mistakes (this will be discussed later in this chapter). Notions of source cases, problems and solutions are defined as usual. A case is made of a problem part pb and a solution part sol. pb and sol are expressions representing a problem and a solution of the domain in the chosen knowledge representation formalism. It is also assumed that there exists a binary relation linking a solution sol to a problem pb meaning that "sol is a solution of pb". A source case, denoted srce-case, is a pair (srce, Sol(srce)). The case base is made of a finite set of source cases. A problem (resp., a solution) is represented by a set of descriptors interpreted in a conjunctive way: if $\mathtt{pb} = \{\mathtt{d}_1, \dots, \mathtt{d}_n\}$, then pb describes the problem whose instances satisfy each of the descriptors $\mathtt{d}_i$ ($i \in \{1, \dots n\}$).

Hypothesis 5.1 (Problem and solution instances)

Each problem (resp., solution) coded in the CBR system represents a set of problem instances (resp., a set of solution instances).

Remark 5.2

About the examples In the following, notions are illustrated by examples coming from the FRAKAS-COOK system. FRAKAS-COOK is a virtual (because not implemented) system applied to the cookery domain. In FRAKAS-COOK, a problem is a request for a specific meal and a solution is a recipe for preparing a dish fulfilling the requirements expressed in the request.

Example 5.1 - Where solution instances are introduced

In this example, the goal is to cook a main-course for Andrew. In the virtual system FRAKAS-COOK, the following descriptors are defined:

- `man`, `vegetarian`, `main-course` (problem descriptors);
- `pasta`, `salmon-sauce` (solution descriptors).

The problem can be expressed by "prepare a one course meal for a vegetarian man", thus:

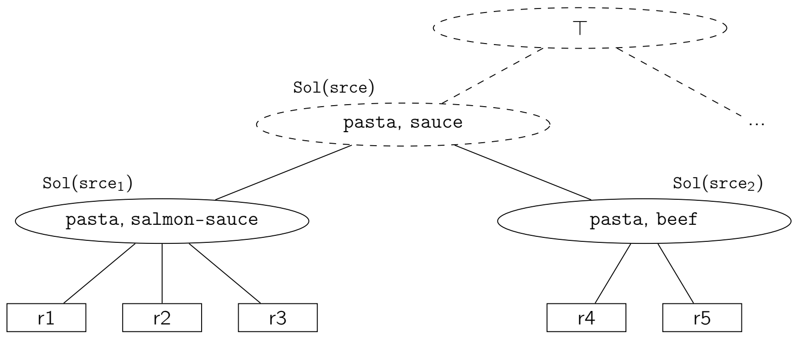
$$\texttt{srce} = \texttt{man},\ \texttt{vegetarian},\ \texttt{main-course}$$

A possible solution to this problem is:

$$\texttt{Sol}(\texttt{srce}) = \texttt{pasta},\ \texttt{salmon-sauce}$$

Figure 5.1 shows an example of solution instances. An interpretation of the solution Sol(srce) is that the recipe r3 (r3 is the recipe of "pasta al salmone affumicato", which is a famous recipe of pasta with a salmon sauce, and also an instance of the solution Sol(srce) = pasta, salmon-sauce) ought to be liked by Andrew, but other solutions are possible too (like r1 and r2 here). This is true, of course, because Andrew is vegetarian and because there is no "not-vegetarian" products in the "pasta al salmone affumicato" dish.

Solution instances in FRAKAS-COOK



The dotted ellipse represents a partially specified solution, solid ellipses are fully specified solutions, rectangle boxes are instances of solutions.

Figure 5.1: Example of solution instances.

Actually, the decomposition of a case in a problem part and a solution part is not mandatory in the FRAKAS approach, but it makes its explanation simpler.

The adaptation produces a result *consistent* with SDK (the system domain knowledge) but not necessarily with the general domain knowledge (i.e., with the global knowledge about the domain). Indeed, the system domain knowledge is always consistent and any solution produced by the system is consistent with respect to SDK. However, because the system does not necessarily have all the knowledge needed to produce a "valid proposition" from the oracle point of view, the solution produced by the system may be inconsistent.

Example 5.2 - Where knowledge about vegetarian people is missing

Once again, Andrew is invited to dinner. FRAKAS-COOK suggests to cook a dish following this type of recipe:

$$\mathtt{Sol(srce)} = \mathtt{pasta,\ beef}$$

A possible solution instance is "spaghetti bolognaise". Although the system knows that Andrew is vegetarian and that beef-sauce contains meat, it proposes this candidate solution because the knowledge "vegetarian people do not eat meat" is not available in SDK. If it were, this proposition would not have been made because it would have been inconsistent.

For practical reasons, FRAKAS handles two levels of presentation of a proposition produced by adaptation: *fully specified propositions* and *partially specified propositions* (motivations for this distinction are discussed in section 5.1.4). A fully specified proposition is a proposition that can be directly applied to solve a problem; a partially specified proposition has to be made precise in order to be applicable. A partially specified proposition can be viewed as a proposition that is more general than a fully specified proposition. To better understand this distinction, let us consider the following example. In figure 5.1, $\mathtt{Sol(srce_1)} = \mathtt{pasta,\ salmon\text{-}sauce}$ and $\mathtt{Sol(srce_2)} = \mathtt{pasta,\ beef}$ are two fully specified propositions (that are linked to recipes) while $\mathtt{Sol(srce)} = \mathtt{pasta,\ sauce}$ is a partially specified proposition. $\mathtt{Sol(srce_1)}$ and $\mathtt{Sol(srce_2)}$ are propositions that are more specific than $\mathtt{Sol(srce)}$.

It must be remarked that the concept of "fully specified proposition" is closely linked to the level of granularity of knowledge representation in the system. A proposition is deemed to be fully specified when its expression only requires a subset of the more specific vocabulary terms available in the system. This is discussed by the hypothesis 5.2.

Hypothesis 5.2 (Fully specified and partially specified propositions)

There exists a computable distinction between a fully specified proposition and a partially specified proposition.

We make this distinction because it helps us to support and to simplify the interaction process with the user, but it must be remarked that this is not a mandatory assumption and that we can avoid it in most situations. A way to distinguish between fully specified propositions and partially specified propositions is to split the vocabulary for representing cases in two subsets: the *abstract* vocabulary and the *concrete* vocabulary. If a proposition needs some of the abstract vocabulary in order to be represented, then this proposition is said to be partially specified.

Example 5.3 - Difference between abstract and concrete vocabulary

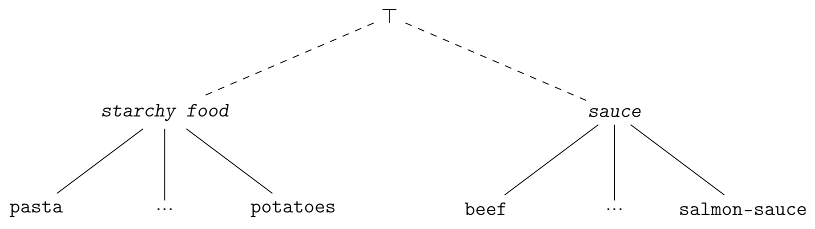
Figure 5.2 gives a graphical view of a subset of the vocabulary used in FRAKAS-COOK. It can be remarked that *concrete* vocabulary is located on the leaves of the tree (cf. figure 5.2). The descriptor `sauce` is a descriptor belonging to the abstract vocabulary. A proposition such as:

$$\mathtt{Sol(srce)} = \mathtt{pasta,\ sauce}$$

is a partially specified proposition. Indeed, `sauce` is an abstract word. To prepare a dish based on this recipe, the type of sauce, such has `salmon-sauce` or `beef`, has to be specified (because, in the system, `salmon-sauce` and `beef` belong to the concrete vocabulary). Hence, the two following propositions are fully specified propositions more specific than this partially specified proposition:

Sol(srce) = pasta, beef and Sol(srce) = pasta, salmon-sauce

Vocabulary in FRAKAS-COOK



The abstract vocabulary is written in italics.

Figure 5.2: A subset of the vocabulary used in FRAKAS-COOK.

5.1.3 Exploitation of reasoning failures by FRAKAS

When a proposition made by FRAKAS does not work, an interactive process aiming at correcting the solution is triggered. The expert is highly involved in this process. Assuming that a proposition is not valid is equivalent to saying that it is inconsistent and amounts to saying that there is at least one inconsistency in the proposition (according to the general domain knowledge). The role of the expert is to highlight this inconsistency. This process allows the system to learn a new piece of knowledge and to give an improved proposition for the current problem. In the following, we make the assumption that the expert does not make mistakes. This assumption is required because FRAKAS does not have mechanisms allowing it to sort out between "correct" and "incorrect" knowledge (how could it?); it needs the interactions with its outside environment to check if a piece of knowledge is valid or not. In other words, FRAKAS considers as correct all the knowledge it learns.

Because of the distinction between fully specified propositions and partially specified propositions, FRAKAS is able to divide the failures into two sub-types of failures: failures on general propositions (called type 1 failures), and failures on specific propositions (called type 2 failures). This division allows an efficient two-steps analysis of the failure to be performed. Each step may lead to specific knowledge acquisition, though quite similar (the second step consists in specifying a proposition before applying the same process as for type 1 failures).

Type 1 failures: inconsistency of a general proposition

Characterisation. Type 1 failures are observed when the oracle points out that the assessment "Sol(tgt) solves tgt" is inconsistent. The inconsistency can signify that the proposition by itself is inconsistent (like serving a raw boiled egg), unrealizable (like making an omelette with a hard boiled egg), or inconsistent with the context of the target problem (like serving eggs when there are none in the refrigerator or serving meat to a vegetarian).

Analysis of the failure. When such a failure occurs, the expert is supposed to highlight (using an appropriate interface), the part of the proposition Sol(tgt) that is inconsistent with his knowledge of the target problem. The part he highlights defines a subset denoted Inc of the set of descriptors constituting Sol(tgt). The first acquired piece of knowledge added to SDK is the fact that "Inc is false" as it is indeed inconsistent with the general knowledge.

Type 2 failures: instances of a partially specified proposition might contain failures

Characterisation. Type 2 failures may occur when the proposition is not specific enough. A partially specified proposition is a proposition that has to be made precise before being applied. FRAKAS is capable at any time of determining if a proposition is only partially specified. This qualification is performed using a specific analysis of the vocabulary used to describe the solution. The vocabulary is split in two subsets: concrete variables and abstract variables. When a solution needs at least one abstract variables to be represented, it is said to be partial.

A set of *instances* is associated to each partially specified proposition. An instance is a fully specified proposition consistent with the partially specified proposition and more specific than the partially specified solution is. Let SI be the set of instances of Sol(tgt), a partial proposition. An instance $s \in SI$ is a non-partial candidate proposition for tgt. If s is deemed satisfactory by the oracle, it constitutes a consistent (and non-partial) proposition for the problem tgt.

Therefore, specifying a partially specified proposition consists in finding an instance of the proposition in SI.

Analysis of the failure. When a solution is partially specified, all the candidate instances in SI are presented to the expert. The expert is asked to check the consistency of each instance $s \in SI$. If an instance s is inconsistent with his knowledge, the expert has to highlight a minimal part Inc of the descriptors of s. Then, a process similar to the one used to handle type 1 failures is performed: a piece of knowledge Inc is built and "Inc is false" is added to SDK.

5.1.4 Main algorithm of FRAKAS

As it was introduced before, the distinction between fully specified solutions and partially specified solutions has been made to facilitate the analysis of the candidate solutions by the expert. To better understand how this distinction allows an efficient approach for correcting solutions, consider the following example.

Example 5.4 - The way of handling failures

We assume that a partial proposition expressed by $\texttt{Sol(srce)} = \texttt{pasta, sauce, meat}$ has sixteen instances (i.e., is a general proposition subsuming sixteen concrete propositions). This partial proposition is inconsistent thus all its instances are inconsistent. A simple way to handle the inconsistency is to correct the general proposition rather than all its instances.

Remark 5.3 - About a design choice

A choice has been made: going from general proposition to specific propositions although the other way would have been possible too. Processing first the partially specified propositions is just an arbitrary choice. We could have decided to process the failures the other way, i.e. to analyse directly all the possible fully specified propositions. Technically speaking, there is no difficulty: if a correction is made on a specific proposition, it can be extended to the more general proposition. This choice has been made for experimental purposes. The point was not to be efficient or to alleviate the effort of the oracle but just to illustrate a possible way of handling failures.

Algorithm 2 describes a way of handling failures in FRAKAS. This algorithm is detailed because it is the one that is used in the prototype FRAKAS-PL, but, as it has been stated in the remark 5.3, other ways of handling failures are possible too.

Basically, type 1 failures are processed first. When there are no more type 1 failures, the system checks if the proposition is fully specified. If this is the case, then the problem is solved, otherwise, possible type 2 failures have to be analysed. After each knowledge acquisition (i.e. after each interaction), the source case is adapted using the newly available knowledge and the new proposition is examined by the oracle. This algorithm does not perform a new retrieval after each interaction. However, it could be relevant to do so. Indeed, the domain knowledge is used during the retrieval of the source case, thus acquiring new knowledge might change the retrieved case. Deciding whether or not a retrieval should be done after each interaction with the expert is an important question. It involves several issues with regard to performance of the system, accuracy of the results, availability of the expert, etc. Thus this design choice has to be made by considering the requirements of the target application.

5.2 FRAKAS-PL

FRAKAS principles have been implemented in a prototype called FRAKAS-PL. In this prototype, the chosen knowledge representation formalism is that of propositional logic. The adaptation is performed using the conservative adaptation principle. Interactions with the user are performed through a set of graphical interfaces. The user behaves alternatively as an expert or as an oracle, depending on the situation (this will be discussed later in remark 5.5).

This section describes the main aspects of the implementation of FRAKAS-PL: knowledge representation formalism and principles of conservative adaptation. As they are not within the scope of this study, retrieval issues are not going to be discussed. To illustrate FRAKAS-PL, section 5.3 presents two examples of use of FRAKAS-PL in the field of oncology. The examples are inspired by the KASIMIR project [d'Aquin et al., 2006].

Algorithm 2 Main algorithm of FRAKAS.

```
Input: tgt, SDK, CB
1: $(\mathtt{srce}, \mathtt{Sol}(\mathtt{srce})) \leftarrow \mathit{Retrieval}(\mathit{SDK}, \mathtt{tgt}, \mathtt{CB})$
2: $\mathtt{Sol}(\mathtt{tgt}) \leftarrow \mathit{Adaptation}(\mathit{SDK}, (\mathtt{srce}, \mathtt{Sol}(\mathtt{srce})), \mathtt{tgt})$
3: {Taking into account type 1 failures}
4: while Sol(tgt) is inconsistent (according to the oracle) do
5:    The expert points out Inc {Inc: the inconsistency}
6:    The expert gives a textual explanation of the failure (stored for later use)
7:    'Inc is false' is integrated to SDK
8:    $\mathtt{Sol}(\mathtt{tgt}) \leftarrow \mathit{Adaptation}(\mathit{SDK}, (\mathtt{srce}, \mathtt{Sol}(\mathtt{srce})), \mathtt{tgt})$
9: end while
10: {Taking into account type 2 failures}
11: if Sol(tgt) is fully specified then
12:    Exit
13: end if
14: while There is an inconsistent interpretation of Sol(tgt) do
15:    {Justification of this loop:}
16:    {The modification of the knowledge base can generate new inconsistent adaptations}

17:    for all inconsistent interpretation do
18:       The expert points out Inc
19:       The expert gives a textual explanation of the failure (stored for later use)
20:       'Inc is false' is integrated to SDK
21:    end for
22:    $\mathtt{Sol}(\mathtt{tgt}) \leftarrow \mathit{Adaptation}(\mathit{SDK}, (\mathtt{srce}, \mathtt{Sol}(\mathtt{srce})), \mathtt{tgt})$
23: end while
```

5.2.1 Propositional logic

This section introduces reminders of the notions of propositional logic used in this chapter. Propositional formulas are built on $\mathcal{V}$, a finite set of propositional variables. Each piece of knowledge manipulated by FRAKAS-PL can be represented by a propositional formula.

$\mathcal{V}$ is partitioned in $\{\mathcal{V}^c_{\mathtt{pb}}, \mathcal{V}^a_{\mathtt{pb}}, \mathcal{V}^c_{\mathtt{sol}}, \mathcal{V}^a_{\mathtt{sol}}\}$ where $\mathcal{V}^x_{\mathtt{pb}}$ (resp., $\mathcal{V}^x_{\mathtt{sol}}$) represents the variables used to represent some problems (resp., some solutions), and if $x = c$ (resp., $x = a$) then this set only contains variables said to be *concrete* (resp., *abstract*). A problem (resp., a solution) is a formula whose variables belong to $\mathcal{V}^c_{\mathtt{pb}} \cup \mathcal{V}^a_{\mathtt{pb}}$ (resp., to $\mathcal{V}^c_{\mathtt{sol}} \cup \mathcal{V}^a_{\mathtt{sol}}$). This distinction between problem variables and solution variables allows one to express a source case as a conjunction of its problem part and its solution part: $\mathtt{srce\text{-}case} = \mathtt{srce} \wedge \mathtt{Sol}(\mathtt{srce})$.

An interpretation on $\mathcal{V}$ is a function $\mathcal{I}$ that, to $x \in \mathcal{V}$, associates $x^{\mathcal{I}} \in \{\mathtt{T}, \mathtt{F}\}$. $\mathcal{I}$ is prolongated on the set of the formulas build on $\mathcal{V}$ in the usual way (for example, $(f \wedge g)^{\mathcal{I}} = \mathtt{T}$ iff $f^{\mathcal{I}} = \mathtt{T}$ and $g^{\mathcal{I}} = \mathtt{T}$). $\mathcal{I}$ is a *model* of f if $f^{\mathcal{I}} = \mathtt{T}$. f entails g (resp., f is equivalent to g) denoted by $f \vDash g$ (resp. $f \equiv g$) if $\mathtt{Mod}(f) \subseteq \mathtt{Mod}(g)$ (resp., $\mathtt{Mod}(f) = \mathtt{Mod}(g)$). f implies g modulo SDK, noted $f \vDash_{SDK} g$ if $SDK \wedge f \vDash g$. $\mathtt{Mod}(f)$ denotes the set of the models of f. The instances of a problem (resp., a solution) are defined here as its interpretations on $\mathcal{V}^c_{\mathtt{pb}} \cup \mathcal{V}^a_{\mathtt{pb}}$ (resp., on $\mathcal{V}^c_{\mathtt{sol}} \cup \mathcal{V}^a_{\mathtt{sol}}$).

A solution $\mathtt{sol}$ is partial if it is not possible to express it without any abstract variable,[9] in other words, if there exists no f such that $\mathtt{sol} \equiv_{SDK} f$ and such that no variable of f belongs to $\mathcal{V}^a_{\mathtt{sol}}$.

5.2.2 Conservative adaptation

FRAKAS is designed to enable knowledge acquisition in CBR systems performing conservative adaptation. This section briefly describes the principles of conservative adaptation because a correct understanding of this adaptation approach is needed to understand how FRAKAS exploits reasoning failures (and thus adaptation failures) to learn knowledge. More details on conservative adaptation are given in [Lieber, 2006], [Lieber, 2007a], [Lieber, 2007b] and [Cojan and Lieber, 2008]. The idea of conservative adaptation is related to the theory of revision of knowledge bases [Alchourrón et al., 1985], [Konieczny, 1999]: the revision of an old knowledge base by a new one consists in making a minimal change on the former while being consistent with the latter. The revision of a knowledge base is performed by a revision operator denoted by $\circ$. Hence, as will be discussed at the end of this section, the revision operator plays the role of an adaptation operator in the CBR field.

Remark 5.4 - Two meanings of the word "revision"

> *In this chapter, the word "revision" refers to the principle of belief revision theory. It has nothing to do with the revise step of the CBR cycle nor with the revision of cases.*

Justification of the conservative adaptation. The conservative adaptation finds its justification in the classic CBR principle:

Similar problems have similar solutions.

This principle has been formalised in [Dubois et al., 1997] by:[10]

$$\mathcal{T}(\mathtt{Sol}(\mathtt{srce}), \mathtt{Sol}(\mathtt{tgt})) \geq \mathcal{S}(\mathtt{srce}, \mathtt{tgt})$$

In this formalisation, $\mathcal{S}$ ant $\mathcal{T}$ are similarity measures respectively between problems and solutions. According to this principle, the solution $\mathtt{Sol}(\mathtt{tgt})$ is constrained to be similar to $\mathtt{Sol}(\mathtt{srce})$.

A proposition $\mathtt{Sol}(\mathtt{tgt})$ is always produced by an adaptation. There are multiple ways to perform adaptation in accordance with the CBR principle, starting from the so-called *null-adaptation*. In null-adaptation, $\mathtt{Sol}(\mathtt{tgt}) = \mathtt{Sol}(\mathtt{srce})$. Null-adaptation is justified in [Riesbeck and Schank, 1989] by the fact that "people often do very little adaptation". But one limit of null-adaptation is the fact that "$\mathtt{Sol}(\mathtt{srce})$ solves $\mathtt{tgt}$" might contradict some domain knowledge. In such a situation, a suitable strategy for adaptation is to keep as many features of the solution as possible while keeping the available knowledge consistent. The example 5.5 illustrates this limitation.

[9]This is checked in FRAKAS as follows. For each $\mathcal{I} \in \mathtt{Mod}(\mathtt{sol})$, let $\mathcal{I}^-$ be the interpretation obtained by projection of $\mathcal{I}$ on the set of variables $\mathcal{V} \backslash \mathcal{V}^a_{\mathtt{sol}}$. Then, let $\mathtt{sol}^-$ be a formula whose models are the $\mathcal{I}^-$'s, for $\mathcal{I} \in \mathtt{Mod}(\mathtt{sol})$. Then the test $\mathtt{sol} \equiv_{SDK} \mathtt{sol}^-$ is done; $\mathtt{sol}$ can be written without any abstract solution variable (i.e., $\mathtt{sol}$ is not partial) iff this test holds.

[10]Dubois et al. notation have been adapted to our notations.

Example 5.5 - Where null-adaptation fails

Annie wants to serve a salad to her nephew Andrew. She remembers that she has made a salad for Ross (Andrew's brother) and that he loved it, so she decides to prepare the same salad (null-adaptation). The salad is made of green salad, tomatoes, crispy bacon, corn and olive oil (and a lot of delightful other ingredients which don't contain meat). Just before starting to prepare the salad, she discovers that Andrew is vegetarian. Yet, Annie's domain knowledge says that vegetarian people do not eat meat and that bacon is meat. There is a contradiction between the solution obtained by null-adaptation and the domain knowledge: Annie needs to adapt her recipe.

A simple way to adapt Annie's recipe to fit the problem while staying consistent with the domain knowledge is to remove the part of the solution that is inconsistent. In this example, it consists in removing the crispy bacon and keeping all the other ingredients.

Conservative adaptation and CBR knowledge bases. In CBR, the conservative adaptation is based on three types of knowledge:

- $\mathtt{KB}_{\mathtt{srce}}$, the context related to the source problem and its solution. It is the old knowledge, it can be altered but must be altered minimally.
- $\mathtt{KB}_{\mathtt{tgt}}$, the context related to the target problem. It is the new knowledge that must not be altered during the adaptation process.
- SDK, the domain knowledge. It is the knowledge that must be true in any context.

Each knowledge base is assumed to be consistent by itself. The principle of minimal change in the revision theory is the following:

> Given two knowledge bases μ and ψ, the revision of ψ by μ is a knowledge base $\psi \circ \mu$ that entails μ and makes minimal changes on ψ to make this revision consistent (with $\circ$ the chosen revision operator).

Thus, conservative adaptation consists in performing a shift from the source context to the target context in order to find a solution for the target problem, staying consistent with both the target problem and the domain knowledge. This process is *conservative* because the shift operates with a minimal change while staying consistent with the definition of the target problem. Indeed, the context of the target problem cannot be modified. For instance, in the example 5.5, the fact that Andrew is vegetarian cannot be altered.

In other words, the aim of the conservative adaptation is to answer the question: "what minimal change on the knowledge base $\mathtt{KB}_{\mathtt{srce}}$ that must be done to be consistent with $\mathtt{KB}_{\mathtt{tgt}}$?" When $\mathtt{KB}_{\mathtt{srce}}$ and $\mathtt{KB}_{\mathtt{tgt}}$ do not contradict each other, there is no reason to perform any change on $\mathtt{KB}_{\mathtt{srce}}$ and thus, conservative adaptation amounts to null-adaptation. When there is a contradiction, an adaptation has to be performed. The conservative adaptation consists, given a revision operator $\circ$, in computing $\mathtt{KB}_{\mathtt{srce}} \circ \mathtt{KB}_{\mathtt{tgt}}$ and to infer, from this new knowledge base, the pieces of knowledge that are relevant to $\mathtt{Sol(tgt)}$ (where $\mathtt{KB}_{\mathtt{srce}}$ and $\mathtt{KB}_{\mathtt{tgt}}$ are interpreted in consistency with the domain knowledge SDK). In the above example, $\mathtt{KB}_{\mathtt{srce}}$ consists of all the knowledge about Ross having a salad, $\mathtt{KB}_{\mathtt{tgt}}$ consists of all the knowledge about Andrew and his dietary habits, and the issue is, given the domain knowledge, how to adapt Ross' meal to stay consistent with Andrew's demand, in particular, with the fact that he is vegetarian.

In summary, in conservative adaptation, the adaptation is performed thanks to the revision operator, which plays the role of an adaptation operator, and the knowledge used to guide the adaptation is the domain knowledge (in addition to the cases). The study of the revision operators is not in the scope of this work but section 5.2.3 describes the particular revision operator implemented in the FRAKAS-PL prototype. More information on revision operators in propositional logic can be found in [Katsuno and Mendelzon, 1991].

The role of the revision operator in the adaptation process. As it was explained before, the revision of a knowledge base ψ (the old knowledge) by a knowledge base μ (the new knowledge) produces a knowledge base $\psi \circ \mu$ (the revised knowledge base). The revision theory guarantees both the consistency of the revised knowledge base and a set of other properties related to the theory itself. Figure 5.3 illustrates this principle.

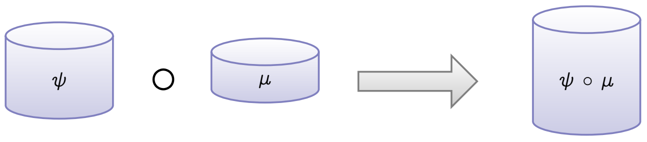
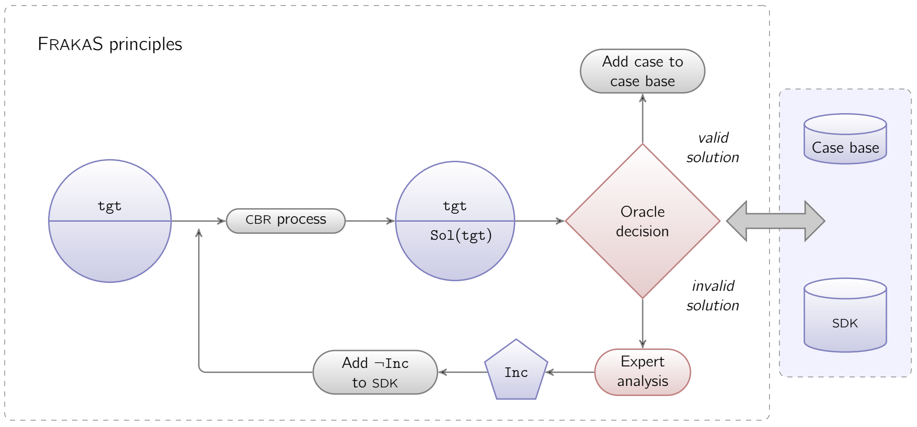


Figure 5.3: Illustration of the revision theory.

FRAKAS principles and conservative adaptation

The FRAKAS principles are illustrated in figure 5.4 and are detailed hereafter.



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.

Figure 5.4: FRAKAS principles.

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.

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. Figure 5.5 details how adaptation is performed using the revision operator in FRAKAS. In order to solve a new problem, FRAKAS retrieves a case which is, of course, consistent with SDK. Then, the retrieved case is adapted, using the revision operator, to propose a solution to the current problem. The revision operator is applied on two knowledge bases: the first one consists of the retrieved source case together with SDK and the other one consists of the target problem, also together with SDK. The result of the adaptation produces a solution consistent with SDK.

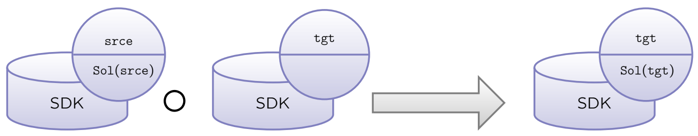


Illustration of the adaptation process in FRAKAS: the cases (target and source cases) are interpreted in consistency with the system domain knowledge base. The result of the adaptation process is a target problem with its candidate solution.

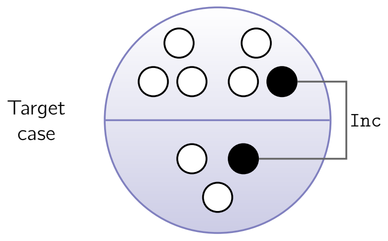
Figure 5.5: Adaptation by revision in FRAKAS.

When the expert points out that a proposition is not satisfactory, it means that there is, given the "general domain knowledge", an inconsistency `Inc` between some of the descriptors of the target case. For instance, if the target problem states that someone is vegetarian, and if the target solution suggests to cook him a piece of meat, then there is an inconsistency in the proposition (given the fact that the general domain knowledge states that vegetarian people do not eat meat). Then, the expert has to identify the inconsistency by marking the parts of the problem and the solution that are concerned. This process is illustrated in figure 5.6.

As soon as the inconsistency is properly identified, the system is able to learn a piece of knowledge that will enable it to avoid reproducing the failure in further reasonings. To resume the previous example, from the inconsistency between "vegetarian" and "meat", the system will learn that vegetarian people do not eat meat. Once it is solved, the target case is added to the case base in prevision of a possible future reuse.

Hence, because of the revision operator, the knowledge available in SDK is *modified* to:

- solve the current problem;
- revise the system knowledge and thus improve it.



Circles represent problem or solution descriptors. Black circles are inconsistent descriptors (according to the expert knowledge): they constitute a new piece of knowledge denoted Inc. The negation of Inc will be added to the knowledge base.

Figure 5.6: An inconsistency in the target case.

5.2.3 Revision operator

In order to perform conservative adaptation, several revision operators can be used. A review of some revision operators is presented in [Katsuno and Mendelzon, 1991]. In the following, the selected revision operator is the Dalal revision operator, noted $\circ_D$ [Dalal, 1988]. $\circ_D$ is defined, in propositional logic, as follows:

Let $\mathtt{dist}$ be the Hamming distance between interpretations on $\mathcal{V}$ ($\mathtt{dist}(\mathcal{I},\mathcal{J})$ is the number of $x \in \mathcal{V}$ such as $x^{\mathcal{I}} \neq x^{\mathcal{J}}$) and let $G^{\lambda}(\psi)$ be a formula (for $\lambda \geq 0$ and ψ a formula) such that:

$$\begin{aligned}\mathtt{Mod}(G^{\lambda}(\psi)) = \{\mathcal{J} \mid \mathcal{J}\text{: interpretation on } \mathcal{V} \text{ such as exists}\\ \mathcal{I} \in \mathtt{Mod}(\psi) \text{ with } \mathtt{dist}(\mathcal{I},\mathcal{J}) \leq \lambda\}\end{aligned}$$

This defines $G^{\lambda}(\psi)$ up to the logical equivalence, which is enough since we adhere to the principle of irrelevance of syntax, saying that whenever $f \equiv g$, an artificial reasoning system using knowledge f makes the same inferences –up to logical equivalence– as the same system using g instead of f. For ψ and μ two formulas such that at least the latter is satisfiable, $\psi \circ_D \mu$ is defined as being $G^{\Delta}(\psi) \wedge \mu$ where Δ is the smallest value such as $G^{\Delta}(\psi) \wedge \mu$ is satisfiable. Intuitively, $\psi \circ_D \mu$ is obtained by generalising ψ minimally (according to the scale $(\{G^{\lambda}\}_{\lambda}, \vDash)$) to be consistent with μ.

5.2.4 Formalisation of the conservative adaptation in propositional logic

The revision operator $\circ_D$ that has been defined previously allows the formalisation of the conservative adaptation (denoted $\mathtt{CA}_{\circ_D}$) in propositional logic. This section presents the conservative adaptation with the CBR notations and gives a full example of adaptation.

Let $\mathtt{tgt}$ be a problem and $\mathtt{srce} \wedge \mathtt{Sol}(\mathtt{srce})$ be a case. The adaptation of this case to solve $\mathtt{tgt}$ aims at giving a proposition of solution $\mathtt{Sol}(\mathtt{tgt})$ of $\mathtt{tgt}$. Given a revision operator $\circ$, the $\circ$-conservative adaptation consists in computing:

$$\mathtt{CA}_{\circ}(SDK, \mathtt{srce} \wedge \mathtt{Sol}(\mathtt{srce}), \mathtt{tgt}) = (SDK \wedge \mathtt{srce} \wedge \mathtt{Sol}(\mathtt{srce})) \circ (SDK \wedge \mathtt{tgt})$$

The result of this computation is a formula f such that $f \equiv_{SDK} \mathtt{tgt} \wedge \mathtt{Sol}(\mathtt{tgt})$ which gives a solution $\mathtt{Sol}(\mathtt{tgt})$ to $\mathtt{tgt}$.

This definition formalises the idea that conservative adaptation consists in keeping as much as possible from $SDK \wedge \texttt{srce} \wedge \texttt{Sol}(\texttt{srce})$, i.e. the source case interpreted keeping consistency with the domain knowledge, while being consistent with $SDK \wedge \texttt{tgt}$, i.e. the target problem interpreted in the context of SDK. The following example illustrates the principle of conservative adaptation applied to CBR.

Example 5.6 - Where Léon performs conservative adaptation at home

This example is strongly inspired by an example created by Jean Lieber [Lieber, 2007b].

Léon is about to invite Thècle and wants to prepare for her a suitable meal. His target problem can be specified by the characteristics of Thècle about food. Let us assume that Thècle is vegetarian (denoted by the propositional variable v) and that she has other characteristics (denoted by c) not detailed in this example: $\texttt{tgt} = v \wedge c$. From his experience as a host, Léon remembers that he had invited Suzanne some time ago and he thinks that Suzanne is very similar to Thècle according to food preferences, except that she is not a vegetarian: $\texttt{srce} = \neg v \wedge c$. He had offered Suzanne a meal with salad (s), beef (b) and a dessert ($\texttt{dessert}$), and she was satisfied by the first two but had not eaten the dessert. Thus Léon has retained the case $(\texttt{srce}, \texttt{Sol}(\texttt{srce}))$ with $\texttt{Sol}(\texttt{srce}) = s \wedge b \wedge \neg\texttt{dessert}$. Besides that, Léon has some general knowledge about food: he knows that beef is meat, that meat and tofu are protein-based food, that tofu is not meat, and that vegetarians do not eat meat. Thus, his domain knowledge is

$$SDK \quad = \quad b \rightarrow m \quad \wedge \quad m \rightarrow p \quad \wedge \quad t \rightarrow p \quad \wedge \quad \neg t \vee \neg m \quad \wedge \quad v \rightarrow \neg m$$

On this example, $\circ_D$-conservative adaptation produces the following result:

$$\texttt{CA}_{\circ_D}(SDK, \texttt{srce-case}, \texttt{tgt}) \equiv_{SDK} \underbrace{v \wedge c}_{\texttt{tgt}} \;\wedge\; \underbrace{s \wedge \neg m \wedge p \wedge \neg\texttt{dessert}}_{\texttt{Sol}(\texttt{tgt})}$$

If Léon follows $\texttt{Sol}(\texttt{tgt})$, he will give Thècle a dinner with a salad, a main course with proteins but no meat (for example, a tofu-based dish) and no dessert.

5.3 FRAKAS-PL(ONCO): an application

FRAKAS-PL(ONCO) is a prototype that focuses on knowledge acquisition after problem-solving sessions. The examples described in this section come from the research project KASIMIR whose framework is knowledge management and decision support in oncology [d'Aquin et al., 2006]. This project is inspired by real situations in the medical domain which have been simplified and anonymised for explanation purposes. A problem is defined as the description of a patient suffering from breast cancer. A solution is a suitable therapy for this patient.[11]

Remark 5.5 - The expert and the oracle in FRAKAS-PL(ONCO)

In the two following examples, we make the assumption that the user behaves as an oracle when evaluating a proposed solution and as an expert when highlighting inconsistencies. As a consequence, and for simplicity's sake, we will use the term "expert" to define the user interacting with the system, whatever his role is.

[11]The medical knowledge given here has been simplified and should not be considered as correct from a medical point of view.

5.3.1 The case of Jules

Target problem

Jules is a man suffering from breast cancer.[12] He has other characteristics that are useful from a medical point of view but that are not detailed in this example (in particular, the fact that he has already had surgery that has removed the tumor, and the fact that the hormone receptors are positive). These characteristics are denoted `other-charac`. The target problem is modelled by:

$$\mathtt{tgt} = \mathtt{man} \wedge \mathtt{other\text{-}charac} \tag{Target problem}$$

Vocabulary

The set of variables $\mathcal{V}$ is partitioned in several subsets as explained in section 5.1. Distinctions are made between *concrete* and *abstract* variables and between *problem* and *solutions* variables. In this example, the available variables are the following:

$$\begin{aligned}
\mathcal{V}^c_{\mathtt{pb}} &= \{\mathtt{man}, \mathtt{woman}, \mathtt{other\text{-}charac}\}\\
\mathcal{V}^c_{\mathtt{sol}} &= \{\mathtt{FEC\text{-}50}, \mathtt{Rad\text{-}50Gy}, \mathtt{ovariectomy}, \mathtt{tamoxifen}, \mathtt{anti\text{-}aromatases}\}\\
\mathcal{V}^a_{\mathtt{sol}} &= \{\mathtt{chemotherapy}, \mathtt{radiotherapy}, \mathtt{anti\text{-}oestrogens}, \mathtt{hormone\text{-}therapy}\}
\end{aligned}$$

Initial domain knowledge

The initial domain knowledge is formalised by:

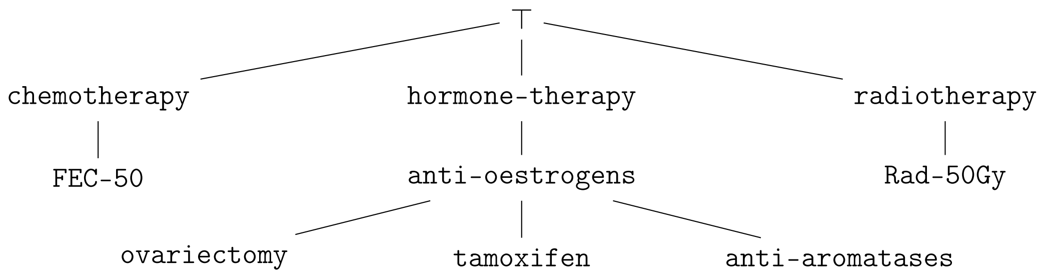
$$\begin{aligned}
SDK_0 = {} & (\neg\mathtt{woman} \vee \neg\mathtt{man}) \wedge\\
& (\mathtt{FEC\text{-}50} \rightarrow \mathtt{chemotherapy}) \wedge\\
& (\mathtt{Rad\text{-}50Gy} \rightarrow \mathtt{radiotherapy}) \wedge\\
& (\mathtt{ovariectomy} \rightarrow \mathtt{anti\text{-}oestrogens}) \wedge\\
& (\mathtt{tamoxifen} \rightarrow \mathtt{anti\text{-}oestrogens}) \wedge\\
& (\mathtt{anti\text{-}aromatases} \rightarrow \mathtt{anti\text{-}oestrogens}) \wedge\\
& (\mathtt{anti\text{-}oestrogens} \rightarrow \mathtt{hormone\text{-}therapy})
\end{aligned}$$

There is a relation of type “kind of” between concrete variables and abstract variables. This relation allows the variables to be organised in a hierarchy. Figure 5.7 illustrates this hierarchy.

SDK_0 translates the following facts. One cannot be a `man` and a `woman` at the same time. `FEC-50` is a `chemotherapy` drug with a dose of 50.[13] `Rad-50Gy` is a breast `radiotherapy` with a dose of $50Gy$. An `ovariectomy` is an `anti-oestrogens` treatment (an ovariectomy is a surgical act that consists of a total ovaries ablation, and thus, it has an anti-oestrogens effect). `tamoxifen` and `anti-aromatases` are two drugs with `anti-oestrogens` effects. All the `anti-oestrogens` treatments are `hormone-therapy` treatments.

[12] Approximatively one per cent of the people suffering from breast cancer are men.

[13] FEC is actually a combination of three chemotherapy drugs: fluorouracil, epirubicin and cyclophosphamide.



Initial domain knowledge without `man` and `woman` concepts.

Figure 5.7: Initial domain knowledge of FRAKAS-PL(ONCO).

Description of the problem-solving process

The aim of this problem-solving episode is to find a solution (i.e. a treatment) for the target problem `tgt`. Interactions with the expert occur during the whole process: the expert is in charge of validating solutions proposed by the system and of correcting faulty knowledge when solutions are not validated.

5.3.2 A running example

This section describes a running example based on the case of Jules defined above. For more information on FRAKAS, a second example is described in appendix A.

First retrieval

In this example, it is assumed that a `srce-case` such that $\texttt{srce} = \texttt{woman} \wedge \texttt{other-charac}$ is retrieved. This source case corresponds to a woman having the same characteristics as Jules, except for her gender. This source case has been retrieved because it is very similar to `tgt` according to the given conservative adaptation-guided retrieval criterion.

The solution of this problem is $\texttt{Sol(srce)} = \texttt{FEC-50} \wedge \texttt{Rad-50Gy} \wedge \texttt{ovariectomy}$: this treatment corresponds to a cure of `FEC-50` (a chemotherapy drug), a breast `radiotherapy` with a dose of 50 Gy, and an `ovariectomy` (ovary ablation), that has an anti-oestrogen effect and, so, constitutes a hormone therapy.

$$\begin{aligned} \texttt{srce} &= \texttt{woman} \wedge \texttt{other-charac} \\ \texttt{Sol(srce)} &= \texttt{FEC-50} \wedge \texttt{Rad-50Gy} \wedge \texttt{ovariectomy} \end{aligned} \qquad \text{(Source case)}$$

First adaptation

Given the initial system knowledge base SDK_0, the target problem `tgt` and the retrieved source case `srce-case`, the system finds a suitable proposition of solution for `tgt`. A conservative adaptation is performed on the initial system domain knowledge SDK_0 together with the source case srce-case and the target problem tgt (according to the Dalal's revision operator $\circ_D$). This consists in doing a conservative adaptation between two knowledge bases:

- $SDK_0 \land \mathtt{srce} \land \mathtt{Sol}(\mathtt{srce})$ (i.e. the source case in the context of the initial domain knowledge),
- $SDK_0 \land \mathtt{tgt}$ (i.e the target problem in the context of the initial domain knowledge)

$$\mathtt{CA}_{\circ_D}(SDK_0, \mathtt{srce\text{-}case}, \mathtt{tgt}) = (SDK_0 \land \mathtt{srce} \land \mathtt{Sol}(\mathtt{srce})) \circ_D (SDK_0 \land \mathtt{tgt})$$

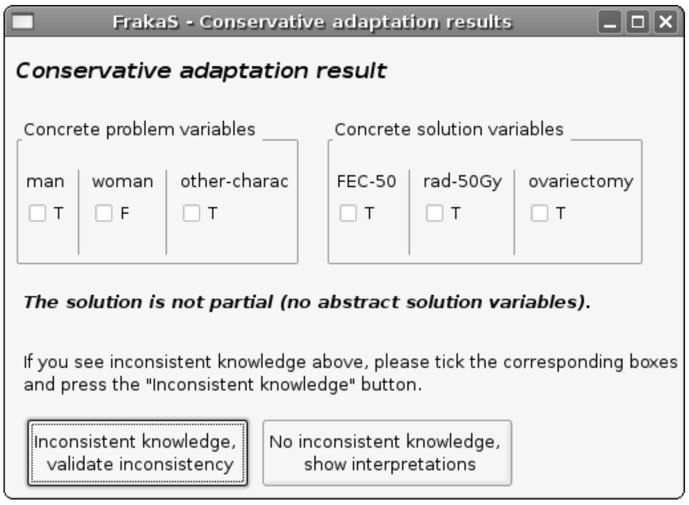
The result of the conservative adaptation is the following:

$$\mathtt{CA}_{\circ_D}(SDK_0, \mathtt{srce\text{-}case}, \mathtt{tgt}) \equiv_{SDK_0} \mathtt{man} \land \mathtt{other\text{-}charac} \land \\ \mathtt{FEC\text{-}50} \land \mathtt{Rad\text{-}50Gy} \land \mathtt{ovariectomy}$$

This result takes into account the properties of the target problem and the source case but does not retain the fact that the source case was a woman, because it is inconsistent with the target problem context.

First interaction —type 1 failure

The result of the first conservative adaptation performed by the system is presented to the expert through a graphical interface (cf. figure 5.8).



FrakaS - Conservative adaptation results

Conservative adaptation result

Concrete problem variables

man	woman	other-charac
☐ T	☐ F	☐ T

Concrete solution variables

FEC-50	rad-50Gy	ovariectomy
☐ T	☐ T	☐ T

The solution is not partial (no abstract solution variables).

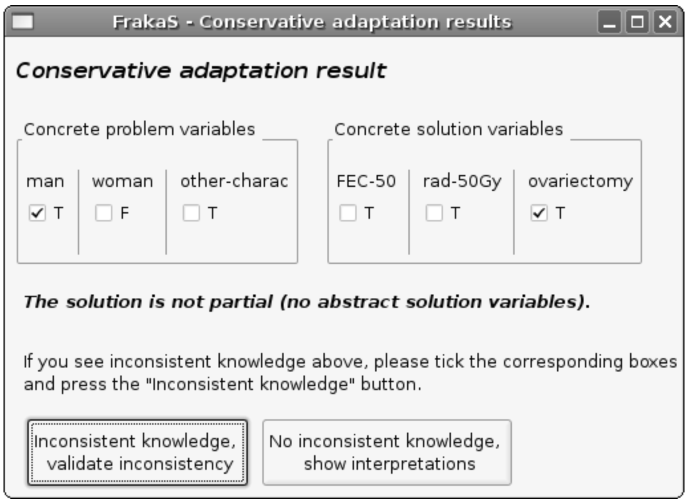
If you see inconsistent knowledge above, please tick the corresponding boxes and press the "Inconsistent knowledge" button.

Inconsistent knowledge, validate inconsistency
No inconsistent knowledge, show interpretations

Display of $\mathtt{CA}_{\circ_D}(SDK_0, \mathtt{srce\text{-}case}, \mathtt{tgt})$

Figure 5.8: First solution presented to the expert.

The expert has to check whether the proposed solution is valid, and for that, he checks it is consistent with his knowledge. In this example, the expert detects an inconsistency: he knows that it is not possible to do an ovariectomy on men. This is a failure of type 1: there is an inconsistency between the system domain knowledge SDK and the expert knowledge. In order to highlight the faulty knowledge, the expert has to check the boxes in front of the involved variables. The conjunction of the corresponding literals constitutes an inconsistent piece of knowledge denoted by `Inc`. In this example, as it is shown in figure 5.9, he checks `man` and `ovariectomy`.


FrakaS - Conservative adaptation results

Conservative adaptation result

Concrete problem variables

man	woman	other-charac
✓ T	F	T

Concrete solution variables

FEC-50	rad-50Gy	ovariectomy
T	T	✓ T

The solution is not partial (no abstract solution variables).

If you see inconsistent knowledge above, please tick the corresponding boxes and press the "Inconsistent knowledge" button.

Inconsistent knowledge, validate inconsistency
No inconsistent knowledge, show interpretations

Display of $\mathtt{CA}_{\circ_D}(SDK_0, \mathtt{srce\text{-}case}, \mathtt{tgt})$

Figure 5.9: Feedback of the expert on the first solution in the form of checked boxes.

The result of this first interaction is the identification of the inconsistency $\mathtt{Inc}_1$ saying that men cannot have ovariectomies:

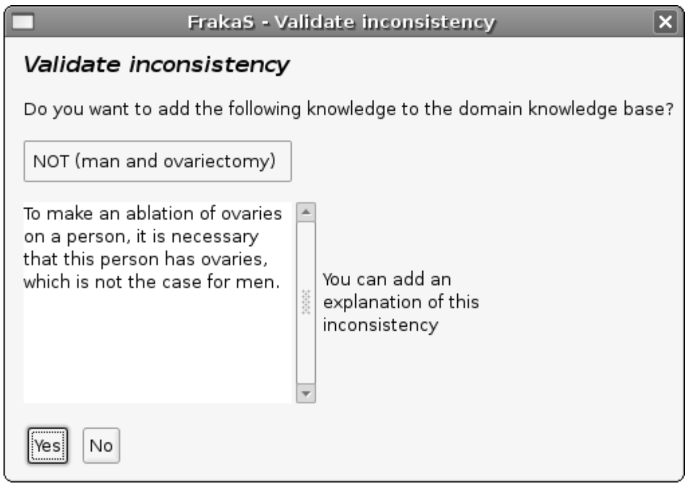
$$\mathtt{Inc}_1 = \mathtt{man} \land \mathtt{ovariectomy} \qquad \text{(Inconsistent knowledge)}$$

First validation and first modification of *SDK*

Before updating the domain knowledge, the system has to verify, with the expert, if the new inferred piece of knowledge is correct or not. This is done through the interface presented on figure 5.10 (the role of the explanation will be explained in section 5.4.1).

When validating (by clicking on "Yes"), the expert confirms that $\mathtt{Inc}_1 = \mathtt{man} \land \mathtt{ovariectomy}$ is false and that $\neg\mathtt{Inc}_1$ can be added to the system domain knowledge. The new system domain knowledge is SDK_1:

$$\begin{aligned} SDK_1 &= SDK_0 \land \neg\mathtt{Inc}_1 \\ &\equiv SDK_0 \land (\neg\mathtt{man} \lor \neg\mathtt{ovariectomy}) \\ &\equiv SDK_0 \land (\mathtt{man} \rightarrow \neg\mathtt{ovariectomy}) \end{aligned}$$


FrakaS - Validate inconsistency

Validate inconsistency

Do you want to add the following knowledge to the domain knowledge base?

NOT (man and ovariectomy)

To make an ablation of ovaries on a person, it is necessary that this person has ovaries, which is not the case for men.

You can add an explanation of this inconsistency

Figure 5.10: Validation of $\neg\texttt{Inc}_1$ and plain text explanation.

Second adaptation

As the first proposition was not satisfactory, the system performs a new adaptation with SDK_1.

The result of this new adaptation is:

$$\texttt{CA}_{\circ_D}(SDK_1, \texttt{srce-case}, \texttt{tgt}) \equiv_{SDK_1} \texttt{man} \wedge \texttt{other-charac} \wedge \texttt{FEC-50} \wedge \texttt{Rad-50Gy} \wedge \\ \neg\texttt{ovariectomy} \wedge \texttt{anti-oestrogens}$$

The conservative adaptation does not keep the ovariectomy, since it is in contradiction with $SDK_1 \wedge \texttt{man}$ but keeps the idea of an anti-oestrogen treatment (because the ovariectomy is an anti-oestrogen treatment).

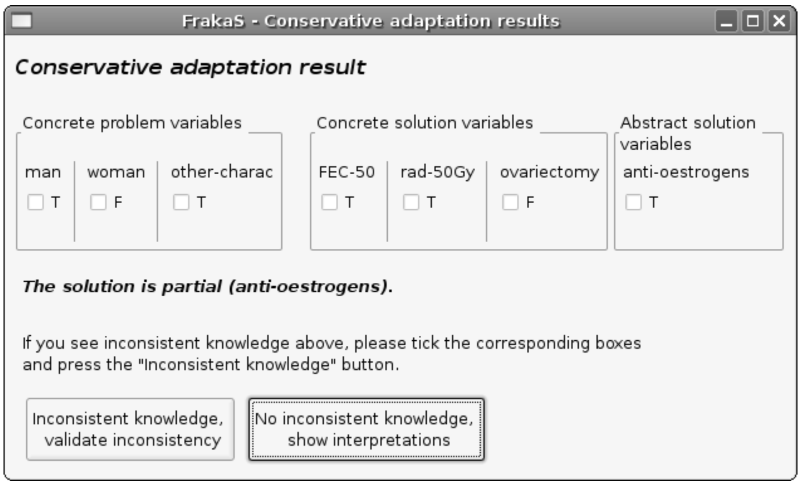
The result of the second adaptation is presented in figure 5.11.

Second interaction —type 2 failure

This second proposition is consistent with the expert domain knowledge, hence there is no type 1 failure. But the system also indicates that some of the variables are *abstract* (in this case, $\texttt{anti-oestrogens}$) and that, consequently, the proposition is not fully specified. This constitutes a type 2 failure. Indeed, $\texttt{anti-oestrogens} \in \mathcal{V}^a_{\texttt{sol}}$ and there exists no formula f that does not contain any variable of $\mathcal{V}^a_{\texttt{sol}}$ that is equivalent to $\texttt{CA}_{\circ_D}(SDK_1, \texttt{srce-case}, \texttt{tgt})$ modulo SDK_1. Then, the type of anti-oestrogen treatment that has to be used must be specified.

By clicking on "No inconsistent knowledge, show interpretations", the expert accesses the available set of interpretations of $\texttt{CA}_{\circ_D}(SDK_1, \texttt{srce-case}, \texttt{tgt})$ as it is shown in figure 5.12.

The expert has to analyse each interpretation of the set (i.e. each line on the figure). For each line, he points out what is inconsistent with his own knowledge, and, for each corresponding interpretation, the conjunction of literals $\texttt{Inc}$ is built. In this example, two of the four interpretations (the first and the fourth on the figure 5.13) are inconsistent. From


FrakaS - Conservative adaptation results

Conservative adaptation result

Concrete problem variables			Concrete solution variables			Abstract solution variables
man	woman	other-charac	FEC-50	rad-50Gy	ovariectomy	anti-oestrogens
☐ T	☐ F	☐ T	☐ T	☐ T	☐ F	☐ T

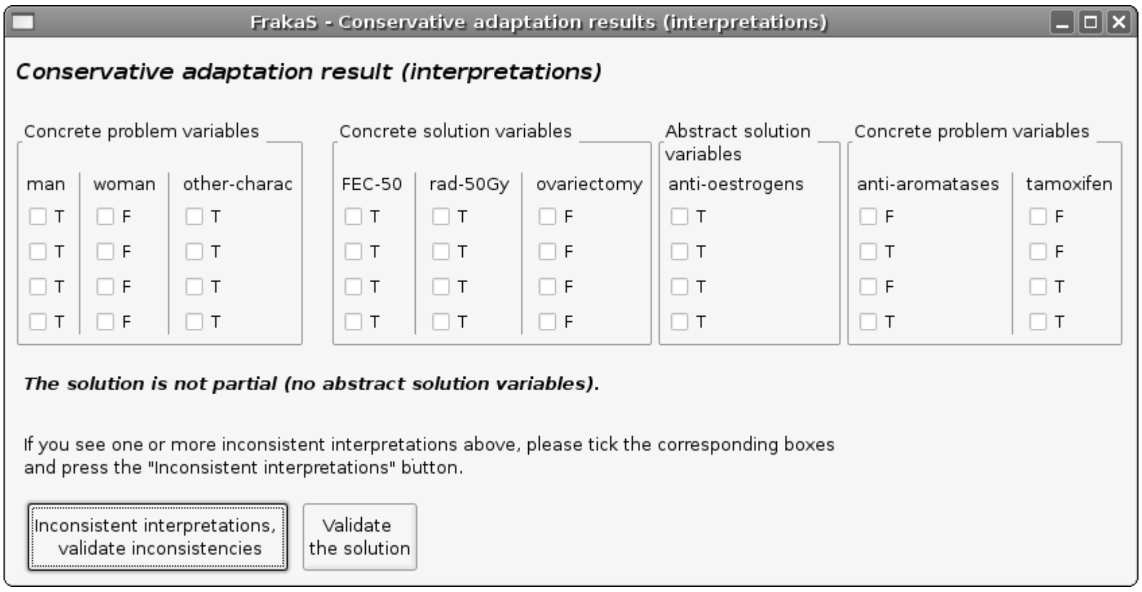
The solution is partial (anti-oestrogens).

If you see inconsistent knowledge above, please tick the corresponding boxes
and press the "Inconsistent knowledge" button.

Inconsistent knowledge, validate inconsistency
No inconsistent knowledge, show interpretations

Display of $\mathtt{CA}_{\circ_D}(SDK_1, \mathtt{srce\text{-}case}, \mathtt{tgt})$ that is judged by the expert to be consistent.

Figure 5.11: Display of the second adaptation.


FrakaS - Conservative adaptation results (interpretations)

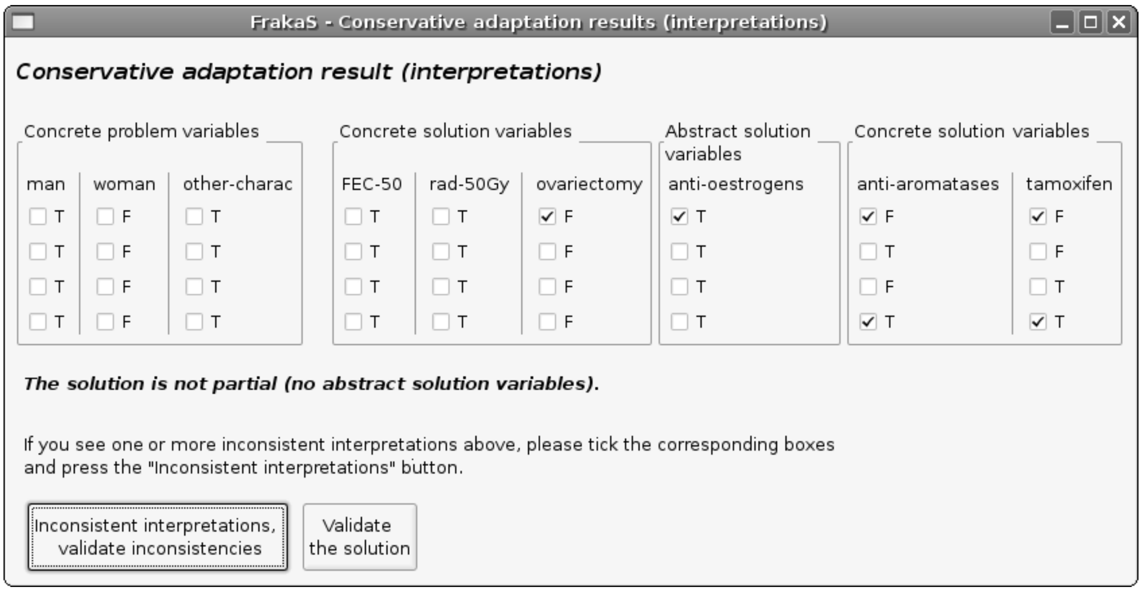
Conservative adaptation result (interpretations)

Concrete problem variables			Concrete solution variables			Abstract solution variables	Concrete problem variables	
man	woman	other-charac	FEC-50	rad-50Gy	ovariectomy	anti-oestrogens	anti-aromatases	tamoxifen
☐ T	☐ F	☐ T	☐ T	☐ T	☐ F	☐ T	☐ F	☐ F
☐ T	☐ F	☐ T	☐ T	☐ T	☐ F	☐ T	☐ T	☐ F
☐ T	☐ F	☐ T	☐ T	☐ T	☐ F	☐ T	☐ F	☐ T
☐ T	☐ F	☐ T	☐ T	☐ T	☐ F	☐ T	☐ T	☐ T

The solution is not partial (no abstract solution variables).

If you see one or more inconsistent interpretations above, please tick the corresponding boxes
and press the "Inconsistent interpretations" button.

Figure 5.12: Display of the interpretations of $\mathtt{CA}_{\circ_D}(SDK_1, \mathtt{srce\text{-}case}, \mathtt{tgt})$.



FrakaS - Conservative adaptation results (interpretations)

Conservative adaptation result (interpretations)

Concrete problem variables			Concrete solution variables			Abstract solution variables	Concrete solution variables	
man	woman	other-charac	FEC-50	rad-50Gy	ovariectomy	anti-oestrogens	anti-aromatases	tamoxifen
☐ T	☐ F	☐ T	☐ T	☐ T	☑ F	☑ T	☑ F	☑ F
☐ T	☐ F	☐ T	☐ T	☐ T	☐ F	☐ T	☐ T	☐ F
☐ T	☐ F	☐ T	☐ T	☐ T	☐ F	☐ T	☐ F	☐ T
☐ T	☐ F	☐ T	☐ T	☐ T	☐ F	☐ T	☑ T	☑ T

The solution is not partial (no abstract solution variables).

If you see one or more inconsistent interpretations above, please tick the corresponding boxes
and press the "Inconsistent interpretations" button.

Figure 5.13: Feedback of the expert on the interpretations he rejects.

the first one, the expert makes a selection that corresponds to:

$$\texttt{Inc}_{2.1} = \neg\texttt{ovariectomy} \wedge \neg\texttt{tamoxifen} \wedge \neg\texttt{anti-aromatases} \wedge \texttt{anti-oestrogens}$$

The domain knowledge is updated in consequence.

$$\begin{aligned} SDK_2 &= SDK_1 \wedge \neg\texttt{Inc}_{2.1} \\ &\equiv SDK_1 \wedge \texttt{anti-oestrogens} \rightarrow \left(\begin{array}{c} \texttt{ovariectomy} \vee \texttt{tamoxifen} \\ \vee\texttt{anti-aromatases} \end{array} \right) \end{aligned}$$

For the other interpretation that is inconsistent with the expert knowledge, $\texttt{Inc}$ is $\texttt{Inc}_{2.2} = \texttt{tamoxifen} \wedge \texttt{anti-aromatases}$ is validated. The meaning of this knowledge is that tamoxifen and anti-aromatases cannot be prescribed to the same patient at the same time.

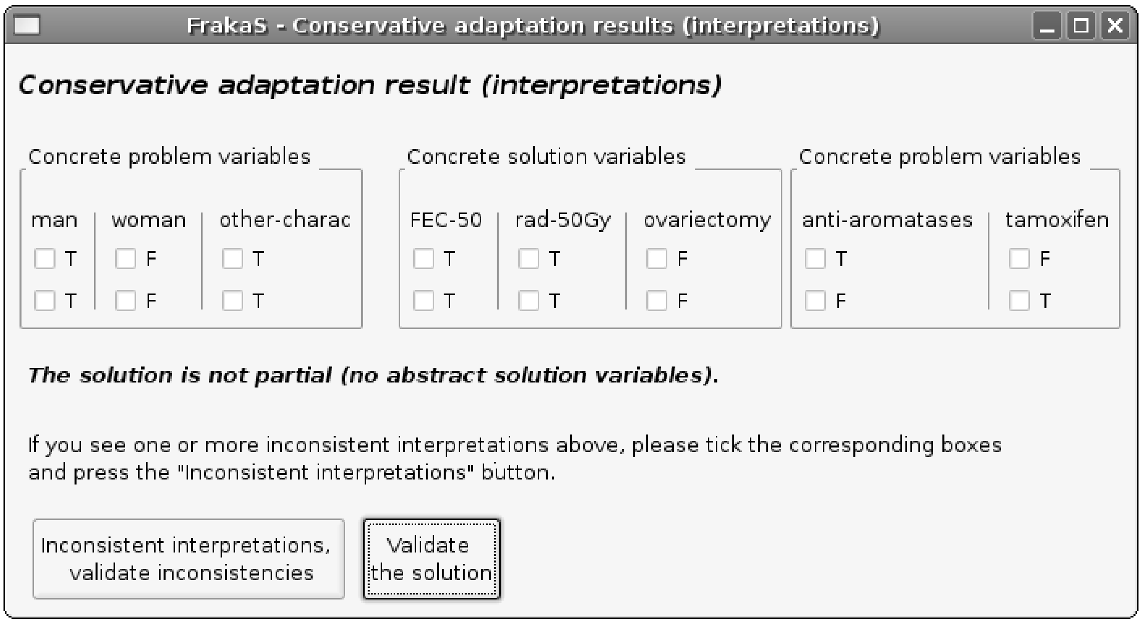
Then, $\neg\texttt{Inc}_{2.2}$ is added to SDK_2:

$$SDK_3 = SDK_2 \wedge \neg\texttt{Inc}_{2.2} \equiv SDK_2 \wedge (\neg\texttt{tamoxifen} \vee \neg\texttt{anti-aromatases})$$

Third adaptation

As there is still no satisfactory proposition for $\texttt{tgt}$, a third conservative adaptation is performed. The result of this new adaptation is the following:[14]

$$\begin{aligned} \texttt{CA}_{\circ_D}(SDK_3, \texttt{srce-case}, \texttt{tgt}) \equiv_{SDK_3} & \texttt{man} \wedge \texttt{other-charac} \wedge \texttt{FEC-50} \wedge \\ & \texttt{Rad-50Gy} \wedge \neg\texttt{ovariectomy} \wedge \\ & (\texttt{tamoxifen} \oplus \texttt{anti-aromatases}) \end{aligned}$$



FrakaS - Conservative adaptation results (interpretations)

Conservative adaptation result (interpretations)

Concrete problem variables

man	woman	other-charac
☐ T	☐ F	☐ T
☐ T	☐ F	☐ T

Concrete solution variables

FEC-50	rad-50Gy	ovariectomy
☐ T	☐ T	☐ F
☐ T	☐ T	☐ F

Concrete problem variables

anti-aromatases	tamoxifen
☐ T	☐ F
☐ F	☐ T

The solution is not partial (no abstract solution variables).

If you see one or more inconsistent interpretations above, please tick the corresponding boxes and press the "Inconsistent interpretations" button.

Figure 5.14: Third solution presented to the expert who validates it.

The result of this adaptation is presented in figure 5.14. The expert does not detect any inconsistency in the proposition (no type 1 failure) and the proposition can be written without abstract variables of solution (no type 2 failure), therefore the proposition is satisfactory. This formula has two interpretations: the first one recommends tamoxifen and the second one, anti-aromatases. These two interpretations correspond to two possible treatments for the patient described by `tgt`.

Remark 5.6 - Textual explanations

The textual explanations have not been discussed in this section because they have not been exploited yet. However, the role of textual explanations is discussed in details in section 5.4.1.

5.4 Future work and discussion

This section presents future research issues for the FRAKAS project and briefly compares FRAKAS with other work. The first part discusses the ability to exploit explanations in FRAKAS to learn more knowledge. The second one is an inventory of problems to be solved in order to improve FRAKAS to make it usable in real world situations. Last section presents a short selection of research that can be related to FRAKAS.

[14] $\oplus$ is the symbol for exclusive or.

5.4.1 Explanations in FRAKAS

FRAKAS-PL already implements the ability to provide explanations after each interaction. An explanation is a full text (written in natural language) that the system user can (after each interaction) explain his decision and/or to "give more knowledge" to the system. Explanations may be complex (either because they are of a high scientific level or because they are expressed in natural language and thus difficult to process automatically) and it is very complicated, if not impossible, to completely automate this part. Formalising this knowledge is the purpose of knowledge engineers and requires competencies that the system user may not have. That is why, in FRAKAS-PL, the system user (who might not be a domain expert) is invited to write explanations in plain text. The resulting document may be used later by a knowledge engineer, in presence of a domain expert and/or the author of the explanation text to manually acquire new knowledge. The interface implemented in FRAKAS-PL offers the possibility of adding explanations, but they have not yet been used. However, in order to see how the system can benefit from processing textual explanations, the reminder of this section is dedicated to the detailed study of an example.

In FRAKAS-PL(ONCO), after each correction step, the user of the system is asked to provide an explanation (cf. figure 5.10). For the example of Jules described in section 5.3.1, the three following texts are given:

> **Text 1:** *To perform an ablation of ovaries on a person, it is necessary that this person has ovaries, which is not the case for men.*
>
> **Text 2:** *The only therapies that are possible and permitted in my hospital for an anti-oestrogen treatment are the ovariectomy, the tamoxifen, and the anti-aromatases.*
>
> **Text 3:** *A given hormone therapy should not use at the same time tamoxifen and anti-aromatases.*

Taking into account the explanations

The three texts entered by the oracle can be used as sources for acquiring some new domain knowledge to be added to $SDK_3 = SDK_0 \wedge \neg\mathtt{Inc}_1 \wedge \neg\mathtt{Inc}_{2.1} \wedge \neg\mathtt{Inc}_{2.2}$. It can be noticed that this new knowledge acquisition (by contrast to the one presented above) is off-line; it is performed during knowledge maintenance operations of the CBR system.

Taking into account the first text. In this text, a knowledge engineer can establish the following knowledge, after discussions with the expert:

- a man does not have ovaries ($f_1 = \mathtt{man} \rightarrow \neg\mathtt{has\text{-}ovaries}$);
- if a person has to be treated by ovariectomy, then this person must have ovaries ($f_2 = \mathtt{ovariectomy} \rightarrow \mathtt{has\text{-}ovaries}$);
- a woman who has already had an ovariectomy does not have her ovaries any more ($f_3 = \mathtt{antecedent\text{-}ovariectomy} \rightarrow \neg\mathtt{has\text{-}ovaries}$).

f_1 and f_2 formalise the text 1. f_3 comes from an answer of the expert to the following question asked by the knowledge engineer: “Are there women who do not have ovaries?”. Then, the state of the domain knowledge is:

$$SDK_4 = SDK_3 \wedge f_1 \wedge f_2 \wedge f_3$$

It can be noticed that during this phase, the vocabulary of the CBR system is enriched. It can also be noticed that $SDK_0 \wedge f_1 \wedge f_2 \vDash SDK_0 \wedge \texttt{Inc} = SDK_1$: f_1 and f_2 explain $\texttt{Inc}_1$ that has to be a consequence of their conjunction. But the additional knowledge f_3 helps to solve correctly the problem $\texttt{tgt}' = \texttt{woman} \wedge \texttt{antecedent-ovariectomy} \wedge \texttt{other-charac}$ by adaptation of the same source case:

$$\begin{aligned}\texttt{CA}_{\texttt{o}_D}(SDK_4, \texttt{srce-case}, \texttt{tgt}') \equiv_{SDK_4} \quad & \texttt{woman} \wedge \texttt{antecedent-ovariectomy} \wedge \\ & \texttt{other-charac} \wedge \texttt{FEC-50} \wedge \texttt{Rad-50Gy} \wedge \\ & \neg\texttt{ovariectomy} \wedge (\texttt{tamoxifen} \oplus \texttt{anti-aromatases})\end{aligned}$$

Taking into account the second text. From the second text, the following fact can be acquired: when an anti-oestrogen treatment is required it is necessarily either an ovariectomy or a treatment with tamoxifen or anti-aromatases:

$$f_4 = \texttt{anti-oestrogens} \rightarrow (\texttt{ovariectomy} \vee \texttt{tamoxifen} \vee \texttt{anti-aromatases})$$

Nevertheless, this does not add new knowledge to what has already been acquired: $f_4 \equiv \neg\texttt{Inc}_{2.1}$. Even if the analysis of this text does not lead to specific knowledge acquisition, the text itself is not useless: it highlights the fact that the knowledge f_4 (or $\neg\texttt{Inc}_{2.1}$) is *contextual*. This knowledge stands in the framework of the expert's hospital but a discussion with the expert points out that there exist other types of anti-oestrogen treatments. Therefore, it is important to avoid using this knowledge $\neg\texttt{Inc}_{2.1}$ in another medical context.

Taking into account the third text. A formalisation of the third text gives $f_5 = \neg\texttt{tamoxifen} \vee \neg\texttt{anti-aromatases}$ but this does not enrich the domain knowledge: $f_5 \equiv \neg\texttt{Inc}_{2.2}$.

An illustration of the usefulness of such explanations can be found in [Lieber et al., 2005].

5.4.2 Future work

The core examples that were used to experiment with FRAKAS-PL(ONCO) are only use cases that have been designed for the experiment purpose by simplifying real medical situations. Several improvements need to be made to FRAKAS to make it usable in real world situations.

In a practical way, for the KASIMIR project, the system should be confronted to cancer specialists under the assistance of computer scientists. This entails to work on the interface ergonomics of FRAKAS and on the optimisation of several parts of the code of FRAKAS. It would be interesting to study the opportunity of selecting relevant interpretations and relevant variables in order to reduce the complexity and to make the work of the oracle easier. This also entails further optimisation of several parts of the code of FRAKAS (the Dalal's revision operator, which has been naively implemented and then optimised during this work, is the most time consuming and takes about 3 seconds CPU on a currently used PC to process the most complex example of the case of Jules).

Integrating FRAKAS to the KASIMIR project is a major issue for this work. Thus, since the KASIMIR system is based on a description logic formalism (see [Baader et al., 2003], [Napoli, 1997]), it will be necessary either to implement translation procedures between propositional logic and description logic or to implement a new version of FRAKAS based on description logic.

Type 2 failures also need additional investigations. Indeed, in our example, an interaction with the oracle was sufficient to handle it but it may not be always the case. Suppose for example that there exists a great number of anti-oestrogen treatments, $\mathtt{ao}_1$, $\mathtt{ao}_2$, ..., $\mathtt{ao}_n$, it will be tedious to enumerate them all and thus to obtain the knowledge $\mathtt{anti\text{-}oestrogens} \rightarrow \mathtt{ao}_1 \vee \mathtt{ao}_2 \vee \ldots \vee \mathtt{ao}_n$. It seems more reasonable that the adaptation process provides such kind of result: "anti-oestrogen treatment, such as one based on tamoxifen or anti-aromatases". This is not a major technical difficulty but it raises a lot of issues in ergonomics and interfaces. Dealing with this issue also requires working with practitioners.

An underlying assumption of this work is that domain knowledge is at any time self-consistent. This does not necessarily hold: SDK may be "approximately true" (true in most situations but not all). In this case, when adding a new knowledge f to SDK, a conflict of $SDK \wedge f$ may occur. Consequently, such a conflict must be detected (which is not difficult). One could go further and propose to merge these two knowledge bases, by using a merging operator (see for example [Konieczny et al., 2004]). In particular, if one considers that SDK can be revised, but f must be kept, one can use a revision operator instead of the conjunction: instead of $SDK_{i+1} = SDK_i \wedge f$, we would have $SDK_{i+1} = SDK_i \circ_{\mathrm{f}}$. This may occur if the use of FRAKAS leads first to the knowledge $\neg\mathtt{Inc}$ and then to the formula f which models the textual explanation given by the oracle. This latter point needs to be studied thoroughly.

Another future work would be to improve the modelling of these reasons to adapt (inapplicability, contraindication, uselessness). This prospect is related to the work on adaptation patterns as defined in the KASIMIR project [d'Aquin et al., 2006]. For example, the contraindication of a treatment ttt for a class of patients pat has been modelled by $pat \rightarrow \neg ttt$. Now, consider the contraindication of epirubicin for patients with a heart problem. One way to adapt a FEC treatment is by removing epirubicin (as it is done in the example described in appendix A). Another way is to keep epirubicin and to add a drug that prevents from the undesirable effects of epirubicin on the heart. This solution is sometimes recommended by physicians but is not consistent with $f = \mathit{heart\text{-}problem} \rightarrow \neg\mathtt{epirubicin}$. Therefore, a more sophisticated modelling of contraindications is required, and this may also be true for the modelling of treatment inapplicability and treatment uselessness. This also raises the problem of the FRAKAS interface: with the above example, the expert will probably check the boxes *heart-problem* and `epirubicin`, that leads to the knowledge $\neg(\mathit{heart\text{-}problem} \wedge \mathtt{epirubicin})$ which is equivalent to f. In a further version of FRAKAS, a new interface based on the reasons to adapt may be developed. One can imagine such an interface with a tab for each reason to adapt. With the above example, the expert may choose the "contraindication tab" and indicate that `epirubicin` is contraindicated because of *heart-problem*. This involves two improvements with the current version: first, it associates explanations to pieces of knowledge and second, it enables to use a sophisticated model of, e.g., contraindication.

5.4.3 Related work

Being an approach following the FIKA principles, FRAKAS can be related to interactive knowledge acquisition approaches discussed in the chapter 3.

The closest application is probably the CHEF system [Hammond, 1990]. CHEF, a case-based planner in the cooking domain, uses a causal model to test an adapted plan. In case of failure, CHEF generates an explanation to guide the repair of the solution. Then, the learning process sets appropriate indexes in order to avoid a later retrieval of the faulty plan in similar circumstances. Besides case-based planning, CHEF inspired many subsequent lines of research based on explanations in order to search for failure causes, propose the associated repairs of the case solution, and modify the knowledge involved in the failure.

The objective that consists in exploiting explanations in FRAKAS can be related to several researches on the use of explanations in CBR. Among work conducted on explanations, the METAAQUA system [Cox and Ram, 1999] provides a taxonomy of failure causes associated with explanations in order to determine appropriate learning strategies. CREEK's reasoning and learning models [Aamodt, 1991] are built upon explanations in a knowledge intensive context and [Sormo et al., 2005] emphasis the importance of explanations in the machine-learning process (and also for human learning and understanding).

In FRAKAS, textual explanations are used off-line by knowledge engineers and domain experts in order to maintain domain knowledge. But FRAKAS also interacts with an expert during the reasoning process in a simple manner to point out faulty knowledge and gives the opportunity to add a textual explanation. A parallel may be established between FRAKAS and the relevance feedback principle [Rocchio, 1966] of information retrieval where items are emphasised or weakened depending on user feedback. In relevance feedback, users are marking documents as relevant to their needs and this gives information to the information retrieval system on how to modify the query for better further retrievals. In FRAKAS, the user marks inconsistent knowledge which allows a new piece of knowledge to be integrated to domain knowledge and further adaptation is retried thanks to this modification. This kind of interaction is quite simple and intuitive for the user while it gives minimal but useful information to the system to enhance the process.

5.4.4 Synthesis

The table 5.1 presents a synthesis of the specificities of FRAKAS and link them with the main FIKA principles.

5.5 Conclusion

This chapter has presented FRAKAS, an approach for incremental domain knowledge acquisition based on the exploitation of reasoning failures (i.e. adaptation failures) of a CBR system. This approach is suitable to frameworks where the adaptation produces solutions that are consistent with the available domain knowledge, as it is the case, for instance, with conservative adaptation. Reasoning failures are decomposed in two types of failures.

FRAKAS, a model based on FIKA	
Description	Principles
Knowledge intensive	The knowledge comes from the domain expert and is gathered during interactions, in the form of domain knowledge.
Interactive	The expert is able to interact with the system after each problem-solving session (in the FRAKAS-PL, this is done trough a graphical interface). When a failure occurs, the expert corrects the involved knowledge and the system retries to solve the problem with its newly available knowledge.
Opportunistic	Failures trigger the knowledge acquisition process. When a failure occurs, the expert has to correct it before being able to continue the problem-solving process.

Table 5.1: FRAKAS, a model based on FIKA.

A first type of failure is detected when the proposed solution is not valid (i.e. it does not work in real-world situations). In practical, this suppose that there are inconsistencies in the proposition and that the expert should be able to detect these inconsistencies. An analysis of this failure allows one to highlight the faulty descriptors which led to the conflict and then, to integrate new domain knowledge. The second type of failure is characterised by the fact that the solution is only partial: some pieces of information needed to make it usable is missing. If an analysis of solution instances shows that some of them are conflicting, the result of this analysis is used to learn new pieces of knowledge. Furthermore, textual explanations provided also constitute a starting point for learning new knowledge or even for clarifying the context of some knowledge pieces. This approach has been implemented in FRAKAS-PL, a prototype based on propositional logic. This formalism has been chosen because it is a simple one for expressing inconsistencies, but the ideas presented here should be transposable to other formal frameworks (e.g. description logics and fuzzy logics). The adaptation mechanism implemented in FRAKAS-PL is the conservative adaptation and the chosen revision operator is the Dalal's revision operator $\circ_D$.

FRAKAS is an adaptation of the FIKA principles to a certain category of CBR systems. The implementation of FRAKAS-PL, though at a prototypical stage, allowed the FRAKAS and FIKA theoretical principles to be validated in an experimental situation. FRAKAS-PL is also a concrete implementation of the conservative adaptation and thus provides an experimental validation of this method of adaptation for CBR.

FRAKAS proposed a new way to perform knowledge acquisition in CBR systems producing solutions that are consistent with the domain knowledge. Current results with FRAKAS-PL(ONCO) are promising and this work has to be continued in particular to overcome the current limitations of the approach and then, to make it usable in real world applications more easily.

A major issue for this work is about the combination of the FRAKAS approach with other knowledge acquisition approaches in order to provide an integration mechanism that allows the good knowledge learning method to be chosen depending on the knowledge to be acquired and the source of acquisition. This raises the issue of the knowledge containers of CBR. In FRAKAS, for example, domain knowledge is used to guide the adaptation process. In other researches, domain knowledge is often used to control the adaptation process, for example by choosing the appropriate adaptation operator or by checking the adaptation knowledge applicability.

Another issue is the evolution of the knowledge bases. As was discussed earlier in this chapter, big issues such as "How to take into account time evolution of a knowledge base when updating?" "What are the links between knowledge bases in CBR systems?" "How is a knowledge base affected when another knowledge base is revised?" have to be investigated.

Future work is currently studied in the context of the TAAABLE project. In particular, the integration of the FIKA sub-approaches IAKA and FRAKAS, is being experimented. This project is still at an early stage, but the experimental field is promising and the results might lead to relevant prospects, in particular, for FRAKAS.

En cuisine, l'erreur est humaine, mais un véritable désastre nécessite un ordinateur.

6 TAAABLE

This chapter presents the TAAABLE project and, more precisely, the work on knowledge acquisition that has been conducted within this project. The first part of the chapter is dedicated to the description of how the TAAABLE project has addressed the textual case-based reasoning challenge of the CCC, using a combination of principles, methods, and technologies of various fields of knowledge-based system technologies, namely CBR, ontology engineering (manual and semi-automatic), data and text-mining using textual resources of the Web, text annotation (used as an indexing technique), knowledge representation, and hierarchical classification. Indeed, to be able to reason on textual cases, indexing them by a formal representation language using a formal vocabulary has proven to be useful. Then, we focus on knowledge acquisition issues that have been raised during this project. We show how the FIKA principles can be applied in the context of this new project and we discuss the benefits they can bring to improve the knowledge base used by the TAAABLE system.

Related publications:
[Champin et al., 2008], [Badra et al., 2008]

Keywords:
Case-based cooking, web application, CBR application, interactive knowledge acquisition, ontology engineering, hierarchical classification, Computer Cooking Contest

Contents

Contributions of this chapter

- Description of the TAAABLE project
- Test of the FIKA principles in a stand-alone application
- Illustration of the FIKA benefits in a real world application

TAAABLE is an application that has been developed initially to enter the first *Computer Cooking Contest* (CCC[15]) that has been organised the 1st of September 2008, in Trier, Germany, during the 9th European Case-Based Reasoning Conference. The Computer Cooking Contest is an open research competition where every participant is invited to submit a software able to create, given a particular query, a recipe for a dish or a menu. The TAAABLE project entered the contest and won the second price, thus becoming "European vice-champion of the CCC".[16]

To address the CCC challenge, the TAAABLE team has decided to develop an application based on a combination of principles, methods, and technologies of various recent fields of research, namely ontology engineering (manual and semi-automatic), information retrieval, data and text-mining using textual resources of the Web, text annotation (used as an indexing technique), knowledge representation, knowledge extraction, knowledge acquisition, hierarchical classification and interface ergonomics. In addition to be a real-world application, TAAABLE has a certain number of characteristics that make of it a pleasant platform to experiment with and to demonstrate or illustrate principles. Hence, TAAABLE is a *CBR application*, *knowledge intensive*, *easy to use*, *accessible*, and *web oriented*.

With time, TAAABLE became more than a simple CCC competitor. Several researchers involved in the project saw in this application a good opportunity to experiment with their own researches (ontology building, automatic knowledge acquisition, CBR experimentations, etc.). For our concern, we have decided to experiment the FIKA principles with TAAABLE. Indeed, during the experimentation phase, the initial knowledge base used by TAAABLE appeared to be incorrect and incomplete. We had to improve it and it was therefore a good opportunity to experiment with interactive knowledge acquisition principles. This chapter is dedicated to the presentation of the TAAABLE application and to the description of our work in the framework of this project.

The chapter is organised as follows. Section 6.1 describes the Computer Cooking Contest, its rules and its objectives. Then, section 6.2 is dedicated to the description of the TAAABLE application and its three main components: the CBR engine, the knowledge base (cases and domain knowledge) and the interface of the application). Section 6.3 presents the knowledge acquisition strategy experimented in TAAABLE and shows how the FIKA principles have been applied in this context. Finally a synthesis and a conclusion close the chapter.

6.1 The Computer Cooking Contest

The Computer Cooking Contest is an open competition that was created to provide researchers with the opportunity to work on the same problem, on the basis of a shared set of data. This kind of contest has a lot of advantages: it is a good opportunity to create collaborations, it attracts people, students, researchers from other communities, and it is a good way to contribute to the diffusion of science.

[15] http://www.wi2.uni-trier.de/eccbr08/index.php?task=ccc

[16] The TAAABLE system is on-line at the following address: http://taaable.fr

6.1.1 CCC rules

The rules of this first edition of the contest were very simple. The goal was to build a system (using any kind of technology) able to answer a given type of queries. The only restriction was the need to use the Computer Cooking Contest recipe book: a set of approximately 900 recipes in a shallow XML structure.

The systems were to be able to take into account the following parameters in the query: ingredients (to include and to avoid), type of cuisine, type of dish and dietary practices.

The competition was structured into three tasks:

- The compulsory task which involved answering queries by selecting and modifying recipes of the recipe book;
- The negation challenge which involved answering queries by avoiding ingredients (i.e. by replacing or removing ingredients);
- The menu challenge which corresponded to the composition of a three-courses menu based on the available recipes of the recipe base.

6.1.2 Query examples

Contest requests were given in free-text and competitors were free to use any way to exploit them. The following requests are examples of what the CCC competitors had to deal with.

Compulsory task.

- Cook a main dish with meat and cauliflower (focus on ingredients and type of meal);
- I would like to have a nut-free cake (focus on dietary practice);
- Prepare a Chinese dessert with fruit (focus on type of cuisine);
- Cook a main dish with turkey, pistachio, and pasta (focus on recipe modification);
- I would like to cook and eggplant soup (focus on recipe modification).

Negation challenge.

- I want to have a salad with tomato but I hate garlic and cucumber.

Menu challenge.

- I do have a filet of beef, carrots, celery, field garlic and cucumber. Potatoes are available, too. For the dessert, we have oranges and mint. A soup would be preferable for the starter.

6.1.3 Evaluation criteria

In such a contest, there is obviously no “best” answer for a given query, thus evaluation criteria were manifolds. Systems were evaluated by scientist who were concerned by the scientific and the technical quality of the proposed softwares. A cook was also member of the jury to evaluate the culinary quality of the recipes proposed by the systems.

6.2 The TAAABLE application

TAAABLE is a web application involving three main components: the CBR engine, the knowledge base and the interface. This section is dedicated to the description of the main parts of this application. It must be remarked that the process of building the knowledge base has required a large amount of work. This work has been done by several members of the Orpailleur Team of the LORIA (a research laboratory in Nancy, France). The CBR engine, developed by Lieber, relies on concepts he introduced during his PhD [Lieber, 1997]. Within our team, we have developed the interface and we have worked with the Orpailleur team on the improvements of the knowledge base (that includes the annotated case base and an ad-hoc ontology) Being responsible for the project, I was also in charge of the coordination of the team and of the integration of the various parts of the project.

Definitions and notations

This chapter makes use of specific notations relating to ontologies and hierarchies because they are used in TAAABLE. This section aims at giving a short explanation of these notations.

Let $\mathcal{O}$ be an ontology. If f is a formula, we note $\vDash_{\mathcal{O}} f$ if, for each model $\mathcal{I}$ of $\mathcal{O}$, $\mathcal{I}$ is a model of f. In other words, $\vDash_{\mathcal{O}} f$ means that, given the knowledge available in the ontologie $\mathcal{O}$, f is "true". In the same way, $\nvDash_{\mathcal{O}} f$ signifies that, given the ontology $\mathcal{O}$, f is false.

The ontology is a set of concepts or classes organised by subsumption relations. If C and D are two classes, $C \sqsubseteq D$ signifies that C subsumes D. The instantiation test is an inference test that consists in, given an ontology $\mathcal{O}$ and a class C, finding all the instances i of $\mathcal{O}$ such as $\vDash_{\mathcal{O}} C(i)$.

In TAAABLE, queries and indexes are represented by a conjunction of literals based on a vocabulary $\mathcal{V}$, which is a finite set of propositional variables. In consequence, f and g denote formulas (i.e., conjunction of literals) representing a query or an index.

For simplicity sake, as we denote $\mathcal{O}$ the system's ontology, we denote $\mathcal{U}$ the user's knowledge. It should be remarked that there is no available representation of this knowledge and that this concept is totally "fictive". However, we use this notation to express the user's opinion. For example, $\vDash_{\mathcal{O}} f$ can be read as "according to the user, f is not true".

6.2.1 The CBR engine

As it was explained in section 6.1, a CCC query can be something like "I would like to cook a nut-free cake with fruits but without chocolate". This query involves several issues: selecting ingredients, avoiding others, selecting a type of dish (salad, cake, etc.), making precise the dietary practise, etc. Moreover, queries can also involve constraints on the "type of cuisine" (Mediterranean, Spanish, etc.). The goal for the CBR engine is, given the query, to retrieve and adapt suitable recipes.

Query representation

In TAAABLE, a query is represented by a conjunction of literals based on a vocabulary $\mathcal{V}$, which is a finite set of propositional variables. For example, searching for a Chinese dessert recipe with fruits but without ginger corresponds to the query:

$$Q_{ex} = \mathtt{Chinese} \wedge \mathtt{dessert} \wedge \mathtt{fruit} \wedge \neg\mathtt{ginger}$$

All these literals belong to the vocabulary $\mathcal{V}$ (i.e., $\mathtt{Chinese}, \mathtt{dessert}, \mathtt{fruit}, \mathtt{ginger} \in \mathcal{V}$) and they respectively stand for "Chinese recipe", "recipe for a dessert", "recipe with fruit", and "recipe with ginger". The propositional variables of the domain represent recipe classes (e.g., `ginger` is used as an abbreviation of `RecipeWithGinger`). A recipe R is also indexed by a conjunction of literals $Idx(R)$. More precisely, the index $Idx(R)$ of a recipe R is the most specific expression in the query language that describes R. For example, if R_{ex} is a recipe for a Chinese dessert with pear, cherry, pineapple, sugar, and no other ingredients, then it is indexed by:

$$Idx(R_{ex}) = \mathtt{Chinese} \wedge \mathtt{dessert} \wedge \mathtt{pear} \wedge \mathtt{cherry} \wedge \mathtt{pineapple} \wedge \mathtt{sugar} \wedge \mathtt{nothing\text{-}else}$$

where `nothing-else` is a conjunction of the negative literals $\neg a$, for all the $a \in \mathcal{V}$ such that $\mathtt{Chinese} \wedge \mathtt{dessert} \wedge \mathtt{pear} \wedge \mathtt{cherry} \wedge \mathtt{pineapple} \wedge \mathtt{sugar} \nvDash_{\mathcal{O}} a$. This kind of "closed world assumption" means that every propositional variable that is not stated in the index recipe, either explicitly or through deduction modulo the ontology, is forced to false.

A recipe R exactly matches a query Q if $Idx(R) \vDash_{\mathcal{O}} Q$ i.e., given the knowledge in the ontology $\mathcal{O}$, every literal stated in Q is satisfied by the index $Idx(R)$. With each recipe R is associated an index $Idx(R)$ and, conversely, given an index $Idx(R)$, the corresponding recipe R is accessed through a pointer.

By analogy with the CBR classical paradigm (cf. figure 2.1, page 9), figure 6.1 illustrates the reasoning process conducted in TAAABLE.

$$\begin{tikzcd}
Idx(R) \arrow[r, phantom, "\sqsubseteq"] \arrow[d, dash, dashed] & \mathtt{SP}(Q) \arrow[r, dash] & Q \arrow[d, dash] \\
R \arrow[rr, dash, "\mathtt{AP} = \mathtt{SP}^{-1}"] & & \mathtt{SP}^{-1}(R)
\end{tikzcd}$$

Figure 6.1: Reasoning process in TAAABLE.

In order to adapt a retrieved recipe, one needs adaptation knowledge. The adaptation knowledge in TAAABLE is used to perform adaptations and substitutions.

Adaptation knowledge

The adaptation knowledge is constituted by a set of adaptation operators. An adaptation operator, as defined in IAKA, is a pair $(\mathtt{r}, \mathcal{A}_{\mathtt{r}})$ where $\mathtt{r}$ is a binary relation between queries and $\mathcal{A}_{\mathtt{r}}$ is an adaptation function associated with $\mathtt{r}$. It has the following semantics: if $Q_1 \;\mathtt{r}\; Q$ then, for every recipe R_1 matching the query Q_1 such as $\mathcal{A}_{\mathtt{r}}(Q_1, R_1, Q) = R$, we can get a recipe matching the query Q.

In the current version of TAAABLE, an adaptation operator $(\mathtt{r}, \mathcal{A}_\mathtt{r})$ is given by a substitution σ such that $Q_1 \ \mathtt{r} \ Q$ if $Q_1 = \sigma(Q)$ and $\mathcal{A}_\mathtt{r}(Q_1, R_1, Q) = \sigma^{-1}(R_1)$, where σ^{-1} is the inverse of the substitution σ. Thus, the adaptation knowledge is given by a set of substitutions.

For example, let $Q = \mathtt{dessert} \wedge \mathtt{lemon}$ be a query and $\sigma = \mathtt{lemon} \rightsquigarrow \mathtt{orange}$ be a substitution. Then, $Q_1 = \sigma(Q) = \mathtt{dessert} \wedge \mathtt{orange}$. If a recipe R_1 exactly matches Q_1, i.e. $\mathit{Idx}(R_1) \vDash_{\mathcal{O}} Q_1$, then $R = \sigma^{-1}(R_1)$, obtained by substitution of orange by lemon in R_1, matches Q, i.e. $\mathit{Idx}(R) \vDash_{\mathcal{O}} Q$.

A substitution $\sigma = \alpha \rightsquigarrow \beta$ is given by two conjunctions of literals α and β. σ is applicable to a query Q if $Q \vDash_{\mathcal{O}} \alpha$ and, when it is the case, $\sigma(Q)$ is obtained from Q by first removing the literals of α and then adding the literals of β: if Q, α, and β are considered as sets of literals closed modulo the ontology $\mathcal{O}$, then $\sigma(Q) = (Q \backslash \alpha) \cup \beta$.

For example, to express that, for Thai cuisine, basil can be substituted by mint, the substitution $\sigma = \mathtt{Thai} \wedge \mathtt{basil} \rightsquigarrow \mathtt{Thai} \wedge \mathtt{mint}$ can be used.

A large part of the substitutions used in the system is simply based on the ontology: if a is a subclass of b in $\mathcal{O}$ (i.e., $(a \implies b) \in \mathcal{O}$), then the *substitution by generalisation* $a \rightsquigarrow b$ may be considered. If $a \rightsquigarrow b$ has been accepted as adaptation knowledge, this means that a food of type a may be substituted by any food of type b. For example, $(x \implies \mathtt{citrus}) \in \mathcal{O}$, for any $x \in \{\mathtt{grapefruit}, \mathtt{lemon}, \mathtt{orange}\}$. Thus, if $\mathtt{orange} \rightsquigarrow \mathtt{citrus}$ is in the adaptation knowledge, then substituting orange by lemon or by grapefruit is likely to produce a "correct recipe".

A numerical value, $\mathtt{cost}(\sigma) > 0$, is associated with each substitution. This value is a measure of the "adaptation effort". This cost represents a preference between adapted recipes: if $\mathtt{cost}(\sigma) < \mathtt{cost}(\tau)$ then, a recipe $\sigma^{-1}(R_1)$ is preferred to a recipe $\tau^{-1}(R'_1)$, where R_1 and R'_1 are two recipes from the recipe book that respectively match $\sigma(Q)$ and $\tau(Q)$.

CBR **process**

The CBR process implemented in TAAABLE basically consist in retrieving a set of suitable recipes and to adapt them if needed. It relies on the notions of strong classification and smooth classification in an index hierarchy [Lieber, 1997]. The adaptation consist in following, in the reverse sense, the similarity path built during the retrieval.

Strong classification. Strong classification points out the indexes $\mathit{Idx}(R)$ that exactly match the query Q, i.e., $\mathit{Idx}(R) \vDash_{\mathcal{O}} Q$. The algorithm is based on a depth-first search of the index hierarchy. This algorithm has been chosen because its response time performances are good [Baader et al., 1994]. If no such index exists, then smooth classification is executed.

Smooth classification. Smooth classification aims at finding a modification Q' of the query Q such that there exists at least one index matching exactly Q'. The modification of Q into Q' is based on a *similarity path*, i.e., a composition $\mathtt{SP} = \sigma_p \circ \sigma_{p-1} \circ \ldots \circ \sigma_1$ of substitutions:

$Q_p = \sigma_p(\sigma_{p-1}(\ldots(\sigma_1(Q))\ldots))$. The cost is defined as follows.

$$\mathtt{cost}(\mathtt{SP}) = \sum_{i=1}^{p} \mathtt{cost}(\sigma_i)$$

The smooth classification algorithm uses an A* search for finding the similarity path SP minimising $\mathtt{cost}(\mathtt{SP})$ such that $Q_p = \mathtt{SP}(Q)$ exactly matches at least one of the indexes of the recipe. A* is a method for finding a similarity path in the query space with a minimal cost (for a given heuristic).

The retrieval process returns a variable number of recipes depending on the request. An algorithm, that is not detailed here, is used to rank the result.

Complexity remark. The use of propositional logic for case retrieval, with a large case base and hundreds of propositional variables, may be computationally costly. Actually, it is not the case in our application: only a fragment of propositional logic leading to polynomial problems is used. In practice, strong classification is very efficient and smooth classification depends on the cost of the first similarity path giving a non-empty set of indexes: the higher is this cost, the more smooth classification requires time and memory, and the more adaptation effort is needed.

Recipe adaptation. If strong classification is successful, then there is no need to adapt the retrieved recipe. If smooth classification is successful, it returns a recipe R_p and a similarity path $\mathtt{SP} = \sigma_p \circ \sigma_{p-1} \circ \ldots \circ \sigma_1$ such that $Idx(R) \vDash_{\mathcal{O}} \mathtt{SP}(Q)$. Then, adaptation consists simply in applying every σ_i^{-1} successively: $R = \mathtt{SP}^{-1}(R_p) = \sigma_1^{-1}(\sigma_2^{-1}(\ldots\sigma_p^{-1}(R_p)\ldots))$ is the adapted recipe.

6.2.2 The knowledge base

The CBR engine of TAAABLE relies on a strong knowledge base. The previous section has highlighted three ingredients of this knowledge base: the ontology, the indexed case base, and the adaptation knowledge. All of them have required a large amount of work that is presented in [Champin et al., 2008] and [Badra et al., 2008]. This section only gives a short overview of this work.

Ontology building

The ontology $\mathcal{O}$ has been built in order to help the conception of the retrieval and adaptation processes of the TAAABLE system. Therefore, the conceptual choice for the ontology development has been strongly driven by the goal of this particular CBR system. During the building of the ontology, three tasks have been carried out: development of the cooking conceptual model, formalisation of the domain, and implementation in OWL language. The reuse of existing ontologies has been carefully examined but no more considered as the examined ontologies did not cover what was intended to be reached in this project. During the elaboration of the cooking conceptual model, several main classes were identified: `Recipe`, `Ingredient`, `FoodComponent`, `Food`, `Action`, `Amount`, and `Utensil`. These classes were sufficiently independent to decide to build a modular ontology. This conceptual choice has been made to facilitate the enrichment of the different modules (because the concepts included in these modules are disjoint) without changing the content of the others. The main relations linking these modules have been manually defined. In order to develop the first version of TAAABLE, we have decided to focus on four modules of this ontology: *ingredient*, *dish type*, *dish origin* and *dish moment* (we plan to use other modules in a future version of TAAABLE). For each module, we have built a hierarchy using manual and semi-automatic techniques. The approaches we have used are described hereafter.

DishType and DishOrigin hierarchies. These two hierarchies have been built manually. Starting from the organisation of dish types and dish origins in the Recipe Source database, a list of general dish types and dish origins has been collected, and hierarchically organised following the specialisation relation.[17],

Concretely, the dish origin hierarchy has only two levels. The first level classifies dishes following their regions, such as *Africa*, *MiddleEast*, *Asia*, *Europe*, etc. Each first level concept is specialised, on the second level, by the country, origin of the dishes. For example, *Austrian*, *British*, *French*, *German*, etc. are "sub-concepts" of *Europe*, meaning that, for example, a *German* recipe is also a recipe of *Europe*.

In the same way, the DishType hierarchy is mainly organised into two levels. At the first level, there are concepts like *BakedGood*, *Burger*, *Dessert*, *MainDish*, etc. The second level details, if necessary, the first level concepts. For example, *BakedGood* is divided into *Bagel*, *Biscuit*, *Bread*, *Muffin*, *Brownie*, *Cookie*, etc. However, these concepts are not detailed more deeply even if more specific categories exist in Recipe Source, as it is for example the case for *Cookie* which is subdivided into *Apple Cookie*, *Chocolate Chip Cookie*, *Diabetic Cookie*, etc. These concepts could indeed be defined by the conjunction of being a *Cookie* and by whether or not they contain some specific food.

Food hierarchy. The `Food` hierarchy was also built manually starting from the Cook's Thesaurus and from the CCC recipe book.[18] The Cook's Thesaurus is a cooking encyclopedia that covers thousands of ingredients, including synonyms and suggested substitutions. More than 250 HTML files were examined in order to extract an initial `Food` hierarchy. At the same time, a terminological database was built in order to associate to each food subconcept a linguistically preferred form (e.g., `bok choy`) as well as a set of morphological variants or synonyms (e.g., `pak choy`, `pak choi`, `Chinese cabbage`, `Chinese mustard cabbage`, etc.). The food hierarchy and the terminological database are then manually enriched by adding new concepts and new lexical forms that occur in the recipe book but do not occur in the Cook's Thesaurus. This extension process is iterative and stops when each food component of the recipe book can be linked to a food concept by the annotation process (see below).

Annotation and indexing process

The CBR engine needs a formal representation of a recipe. The annotation process aims at formally representing the content of a recipe as well as defining its categories. This

[17] http://www.recipesource.com/
[18] http://www.foodsubs.com

process is in between what is usually called *controlled indexing* [Jo et al., 2000] where terms come from a predefined terminology and *semantic annotation* [Uren et al., 2006] where terms (named entities, sequences of words) are explicitly associated with the respective and most specific classes in the ontology. The result of the annotation of a recipe is a set of concepts indexing the recipe. The current state of our prototype does not deal with the preparation part of recipes. Only the list of ingredients is parsed. First, each ingredient entry in the recipe is parsed and split into the following 4-tuple (`<quantity>,<unit>,<food-component>,<modifiers>`). For example, the entry `<IN>1/3 cup milk</IN>` is parsed as `(1/3,cup,milk,_)`. The terminological database guides the parsing process. For instance, as ``light brown sugar'' is a lexical form associated to the food component concept `light_brown_sugar` in the food hierarchy, the entry `<IN>1/2 c Packed light brown sugar</IN>` is parsed as `(1/2,cup,light_brown_sugar, packed)`.

For example, the recipe entitled `Cinnamon Rolls` is indexed by the conjunction of the ingredients `sweet dough mixture`, `brown sugar`, `pecans`, `dark seedless raisins`, `ground cinnamon`, `butter or margarine`, and `sugar glaze` (`butter or margarine` is a concept in the ontology which is more general than both concepts `butter` and `margarine`).

Finally, the annotation process has to index recipes following the region and the country (e.g., Asia, Chinese) and following the dish type (e.g., main dish, dessert). All these types are defined in the ontology as concepts. As there is no indication concerning these types in the recipe book, Recipe Source is used again in order to build a corpus where recipes are assigned to types.

Hence, several techniques have been used in order to build the initial knowledge base used in TAAABLE. Despite the very good results obtained with these techniques, the resulting knowledge base is still incorrect and incomplete (an incorrectness corresponds to a faulty knowledge in the knowledge base). For that reason, we had to find solutions to improve this knowledge base, and that is why we have decided to experiment FIKA with TAAABLE.

6.2.3 The interface

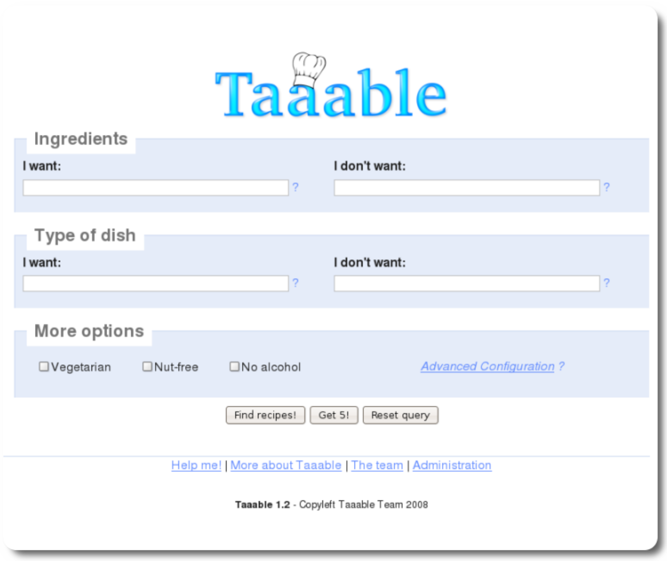
The interface has been developed to be easy to use but also to demonstrate the various functionalities of the application. It includes the following features:

- Interactive help to enter the request;
- Request validation;
- Display results features:
  - Common similarity path and its cost;
  - Specific similarity path and its cost;
  - Adaptation overview;
- Navigation features:
  - Browse recipes;
  - Browse results;
  - Display adapted recipes;
- Display adapted recipe features:
  - Display of the implicit subsumptions;
  - Display of the explicit subsumptions;

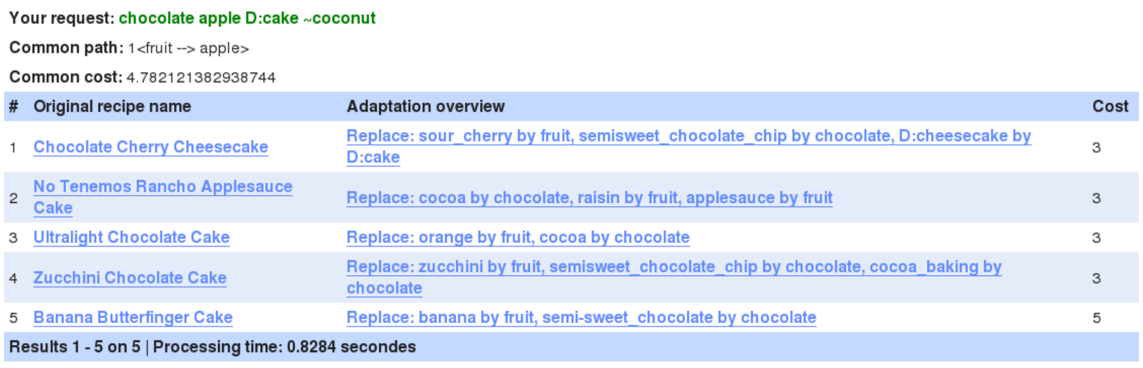
- Display of the original recipe;

- Customisation parameters:
  - Dietary practices (vegetarian, nut-free, no-alcohol);
  - Display parameters;
  - Session parameters.

Figure 6.2 includes two screen-shots of the TAAABLE application. Additional screen-shots are available in appendix B.



(a) Main interface of TAAABLE.



Your request: chocolate apple D:cake ~coconut
Common path: 1<fruit --> apple>
Common cost: 4.782121382938744

#	Original recipe name	Adaptation overview	Cost
1	Chocolate Cherry Cheesecake	Replace: sour_cherry by fruit, semisweet_chocolate_chip by chocolate, D:cheesecake by D:cake	3
2	No Tenemos Rancho Applesauce Cake	Replace: cocoa by chocolate, raisin by fruit, applesauce by fruit	3
3	Ultralight Chocolate Cake	Replace: orange by fruit, cocoa by chocolate	3
4	Zucchini Chocolate Cake	Replace: zucchini by fruit, semisweet_chocolate_chip by chocolate, cocoa_baking by chocolate	3
5	Banana Butterfinger Cake	Replace: banana by fruit, semi-sweet_chocolate by chocolate	5

Results 1 - 5 on 5 | Processing time: 0.8284 secondes

(b) A set of results to the request "I want a chocolate cake with apples, but I hate coconut".

These screen-shots illustrate the main functionalities of the TAAABLE application. The interface is quite simple but offers a certain number of options: navigation, information on costs, overview of adaptation, etc.

Figure 6.2: Overview of the TAAABLE application.

6.2.4 Technical involvements

The TAAABLE project makes use of a certain number of technologies. As we wanted to make of TAAABLE an "open" project, we have decided to use, as often as possible, open technologies:

- Java for the CBR engine implementation;
- Tomcat for the web application management;
- Java, servlets and Java Server Pages for the Web interface;
- SQL database, XML and text files for the storage of the knowledge bases;
- HTML and Javascript for the client interface.

For the sake of usability, we have tried to use the standards of the web applications development (Valid HTML, Valid CSS). Figure 6.3 sums up the technologies that are used in TAAABLE and the features that are implemented.

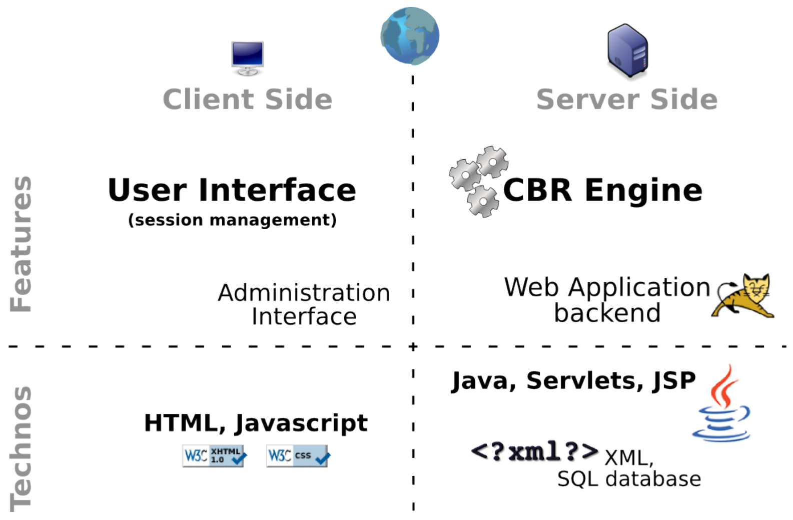


Figure 6.3: Technologies involved in TAAABLE.

6.3 Interactive knowledge acquisition in TAAABLE

During the development of TAAABLE, we have made a lot of experiments. A large number of these experiments have highlighted problems in the knowledge used by TAAABLE. In this section, we present a simple formalisation of the failures we have observed and we discuss of the knowledge acquisition process that can be triggered when such failures occur.

6.3.1 Type of failures in TAAABLE

The role of TAAABLE is to provide a set of recipes, possibly adapted, in answer to a given query. Hence, we have identified three main possible situations of failure:

- A recipe was retrieved by the system but, according to the user, it should not have been (denoted by type A failure);
- The user observes that a particular recipe was not retrieved by the system though it should have been (denoted by type B failure);
- A recipe was retrieved and adapted by the system, but the adaptation is not correct (denoted by type C failure).

In other words, in type A failures, we have $\mathit{Idx}(R) \sqsubseteq \mathtt{SP}(Q)$ and we should not, and in type B failures, we do not have $\mathit{Idx}(R) \sqsubseteq \mathtt{SP}(Q)$ but we should. These failures are illustrated on figure 6.4.

$$\begin{tikzcd}
\mathit{Idx}(R) \arrow[d, dashed, dash, "\text{A or B}"] & \sqsubseteq & \mathtt{SP}(Q) & & Q \arrow[ll] \arrow[d] \\
R & & & & \mathtt{SP}^{-1}(R) \arrow[llll, "\mathtt{AP} = \mathtt{SP}^{-1}"']
\end{tikzcd}$$

C (and neither A nor B)

Figure 6.4: Different failures in TAAABLE.

Naturally, the first situation of failure is much more easy to identify than the second one. Indeed, in order to identify a type B failure, one must have a good knowledge of the available recipes (or one need an *external retrieval process* able to provide him with suitable recipes). However, this is sometimes the case, particularly during the setup phase of the system. For example, while we made our experiments, we often had to deal with problems such as "I asked for a chicken with nuts recipe, why TAAABLE had not retrieved the *chicken with pistachio nuts* recipe? Is it because TAAABLE does not know that pistachio nuts are nuts?". Hence, these situations of failure are of importance and have to be taken into account.

For each failure of type A and type B, two causes can be incriminated: either the system misses a piece of knowledge (type 1 cause) or it has a faulty piece of knowledge (type 2 cause). In the first situation, the knowledge need to be completed, in the second one, the knowledge has to be corrected, if possible. Table 6.1 summarises the type of failures we have identified in the context of the TAAABLE application (except the type C failures).

All the examples introduced in this table are detailed hereafter. One could observe that when a failure occurs, several pieces of knowledge can be incriminated. If it is the case, pieces of knowledge are acquired incrementally, one after the other.

Type A 1 failures

Description of the failure. The system has retrieved a recipe that should not have been retrieved. The reason is that the system has used a faulty knowledge during the reasoning process.

	Cause of the failure	
	Wrong knowledge (type 1 failure)	Missing knowledge (type 2 failure)
A	**Objective:** correct faulty knowledge.	**Objective:** complete knowledge.
	Example: I asked for a sweet cake and the system offers me an olive cake recipe.	**Example:** I asked for a nut-free cake and the system offers me a peanut-butter cake.
$\models_{\mathcal{O}} f \implies g$	**Cause:** for the system, every cake recipe is a sweet recipe.	**Cause:** the system misses the knowledge that peanut-butter contains peanuts and thus, is not a nut-free ingredient.
$\not\models_{\mathcal{U}} f \implies g$	**Action:** correct the knowledge. (`cake` $\implies$ `sweet`)	**Action:** add a new piece of knowledge to the system, namely `peanut-butter` $\implies$ `peanut`.
B	**Objective:** complete knowledge.	**Objective:** correct faulty knowledge.
	Example: I asked for a fruit pie and the system did not retrieve the rhubarb pie.	**Example:** I asked for a vegetarian dish and the system did not retrieve my favourite "omelette aux pommes de terres" (and egg and potatoes dish).
$\not\models_{\mathcal{O}} f \implies g$	**Cause:** the system misses the knowledge that rhubarb is a fruit.	**Cause:** the system considers that eggs are not vegetarian ingredients.
$\models_{\mathcal{U}} f \implies g$	**Action:** add the knowledge (`rhubarb` $\implies$ `fruit`) to the knowledge base.	**Action:** correct the faulty knowledge (`eggs` $\implies$ `meat`).

Table 6.1: Typology of failures in TAAABLE.

- **Relevant element of ontology:** `cake` $\implies$ `sweet`
- **Request:** $g =$ `sweet`
- **Retrieved recipe:** $f =$ `cake` $\land$ `olives`

Given the ontology, `cake` $\implies$ `sweet`, which is contested by the user. Hence, we need to transform the ontology $\mathcal{O}$ in an ontology $\mathcal{O}'$ such as $\not\models_{\mathcal{O}'}$ `cake` $\implies$ `sweet`. In other words, we need a new version of the ontology without the "faulty" knowledge `cake` $\implies$ `sweet`.

Performing such a change on the ontology could by performed using a contraction operator, denoted by $\dot{-}$ [Gärdenfors, 1992]. Then we would have: $\mathcal{O}' = \mathcal{O} \dot{-} ($`cake` $\implies$ `sweet`$)$.

Type A 2 failures

Description of the failure. The system has retrieved a recipe that should not have been retrieved. The reason is that the system misses a piece of knowledge that would have been useful in this case.

- **Relevant element of ontology:** `cake` $\implies$ `bakedgood`
- **Request:** $g = \neg$`peanut`

- **Retrieved recipe:** $f = \texttt{cake} \land \texttt{peanut-butter}$

The ontology does not entails $\texttt{peanut-butter} \implies \texttt{peanut}$, which is highlighted by the user. Hence, we need to transform the ontology $\mathcal{O}$ in an ontology $\mathcal{O}'$ such as $\vDash_{\mathcal{O}'} \texttt{peanut-butter} \implies \texttt{peanut}$. In other words, we need a new version of the ontology with an additional knowledge, that is $\texttt{peanut-butter} \implies \texttt{peanut}$.

Performing such a change on the ontology could by performed using a revision operator, denoted by $\dot{+}$ [Gärdenfors, 1992]. Then we would have: $\mathcal{O}' = \mathcal{O} \dot{+} (\texttt{peanut-butter} \implies \texttt{peanut})$. This amounts to add a new edge in the ingredient hierarchy and possibly to remove all the useless transitivity edges.

Type B 1 failures

Description of the failure. The system has not retrieved a recipe and it should have. The reason is that the system misses a piece of knowledge that would have been useful in this case.

- **Relevant element of ontology:** $\texttt{tart} \implies \texttt{pie}$
- **Request:** $g = \texttt{fruit}$
- **Recipe that should have been retrieved:** $f = \texttt{tart} \land \texttt{rhubarb}$

The ontology does not entails $\texttt{rhubarb} \implies \texttt{fruit}$, which is highlighted by the user. We are in the same case as in type A 2 failures. We need to use a revision operator to correct the ontology with this new knowledge.

For this type of failure, we are going to illustrate in details how the interactive knowledge acquisition process is performed.

The type B 1 problem can be formalised as follows. Given the ontology, $Idx(R)$ is not subsumed by the query Q, though, according to the user, it should.

$$\nvDash_{\mathcal{O}} Idx(R) \sqsubseteq Q$$
$$\vDash_{\mathcal{U}} Idx(R) \sqsubseteq Q$$

Let f be defined as $f = Idx(R) \sqsubseteq Q$. Hence, from the system's viewpoint, $\nvDash_{\mathcal{O}} f$ and from the user's viewpoint, $\vDash_{\mathcal{U}} f$.

Going back to the example, we have:

$$\begin{aligned} Q &= \texttt{pie} \land \texttt{apple} \\ \texttt{SP} &= \texttt{apple} \rightsquigarrow \texttt{fruit} \\ \texttt{SP}(Q) &= \texttt{pie} \land \texttt{fruit} \\ Idx(R) &= \texttt{pie} \land \texttt{rhubarb} \land \texttt{sugar} \land \texttt{nothing-else} \end{aligned}$$

Intuitively, it can be observed that the system misses the knowledge that, from a culinary viewpoint, rhubarb should be considered as a fruit. The issue is to find how to acquire this knowledge through interactions in an acquisition process triggered by the failure. For that purpose, we are going to apply a method similar to the one used in FRAKAS.

In this example, $f = \texttt{pie} \wedge \texttt{rhubarb} \wedge \texttt{sugar} \implies \texttt{pie} \wedge \texttt{fruit}$. However, given the system knowledge, $\not\models_{\mathcal{O}} f$. This can be reformulated as follows.

$$\begin{aligned}
\neg f &= \neg(\texttt{pie} \wedge \texttt{rhubarb} \wedge \texttt{sugar} \implies \texttt{pie} \wedge \texttt{fruit}) \\
&\equiv \neg(\neg(\texttt{pie} \wedge \texttt{rhubarb} \wedge \texttt{sugar}) \vee (\texttt{pie} \wedge \texttt{fruit})) \\
&\equiv \texttt{pie} \wedge \texttt{rhubarb} \wedge \texttt{sugar} \wedge \neg(\texttt{pie} \wedge \texttt{fruit}) \\
&\equiv \texttt{pie} \wedge \texttt{rhubarb} \wedge \texttt{sugar} \wedge (\neg\texttt{pie} \vee \neg\texttt{fruit}) \\
&\equiv \texttt{pie} \wedge \texttt{rhubarb} \wedge \texttt{sugar} \wedge \neg\texttt{fruit}
\end{aligned}$$

In such a case, the system is able to present to the user the following result:

Text 1: *The recipe contains: pie, rhubarb and sugar. The recipe does not contain: fruit.*

On this text, the user is able to highlight an inconsistency:

Text 1 (after analysis by the user): *The recipe contains: pie,* **rhubarb** *and sugar. The recipe does not contain:* **fruit**.

Indeed, rhubarb, from a culinary viewpoint, is a fruit, so if the recipe contains rhubarb, it should necessarily contain fruit, which is not the case, according to the system's knowledge.

Using the information given by the user, the system is able to build a new piece of knowledge denoted by `Inc` (the inconsistent knowledge) and to add the negation of this new piece of knowledge to its knowledge base.

$$\neg\texttt{Inc} = \neg(\texttt{rhubarb} \wedge \neg\texttt{fruit})) \equiv \texttt{rhubarb} \implies \texttt{fruit}$$

With this new knowledge in the system domain knowledge, the recipe of rhubarb pie which has not been retrieved before will be retrieved if the same request ("I want an apple pie") is performed again. Indeed, now, $\models_{\mathcal{O}} \mathit{Idx}(R) \sqsubseteq \texttt{SP}(Q)$.

Type B 2 failures

Description of the failure. The system has not retrieved a recipe and it should have. The reason is that the system has applied a faulty knowledge during the reasoning and that this faulty knowledge has led to the removal of the recipe from the candidate recipes list.

- **Relevant element of ontology:** $\texttt{eggs} \implies \texttt{meat}$
- **Request:** $g = \neg\texttt{meat}$
- **Recipe that should have been retrieved:** $f = \texttt{eggs} \wedge \texttt{potato}$

Given the ontology, $\texttt{eggs} \implies \texttt{meat}$, which is contested by the user. We are in the same case as in type A 1 failures. We need to use a contraction operator to complete the ontology with this new knowledge.

6.3.2 Type C failures

Type C failures occur when a recipe is correctly retrieved but badly adapted. We have not studied this issue in details and this constitutes a prospect for TAAABLE 2.

6.4 Synthesis

The table 6.2 presents a synthesis of the specificities of TAAABLE and link them with the main FIKA principles.

TAAABLE, an application implementing the FIKA principles		
FIKA	Description	Principles
	Knowledge intensive	The knowledge comes from the users of the system.
	Interactive	Users ask requests to the system. When solutions seem odd, they correct (manually) the knowledge base to avoid finding again odd results in the future.
	Opportunistic	Problems in the available knowledge base are identified by users when they use the system and they find recipes that are odd with regard to the given request. In this sense, failures trigger the knowledge acquisition process.

Table 6.2: TAAABLE, an application implementing the FIKA principles.

6.5 Conclusion

The first version of the TAAABLE system relies on simple choices: a propositional representation language (that is not expressive enough for representing, e.g., quantities), a vocabulary restricted to ingredients and recipe types (that does not handle, e.g., the actions of the preparation), and a CBR process relying on simple choices. The future work on this project will concentrate on improving these features.

From the specific viewpoint of knowledge acquisition, several studies are going to be carried on with TAAABLE. For example, the CABAMAKA [d'Aquin et al., 2007] system will be used to acquire adaptation knowledge in the next version of the system. Additional experiments with the FIKA principles will also be performed. TAAABLE is a convenient project to experiment with several knowledge acquisition (or knowledge learning strategies) at the same time, and why not, to try to combine them. This issue is the main focus of our work for TAAABLE 2.

CBR systems are often in error but never in doubt.

7 Discussion

As several recent researches have pointed out, the issue of knowledge acquisition in case-based reasoning has become increasingly important ([Richter and Aamodt, 2005], [López de Mántaras et al., 2005]). Our research focuses on this issue. The main question that interests us is “how to provide the users with systems that more efficiently support their interactions during the knowledge acquisition process?”. We consider this issue as a major one since we believe that the user has a central and active role to play in the knowledge acquisition process.

Consequently, we have developed general principles for an interactive and opportunistic knowledge acquisition support in case-based reasoning. Then, we have illustrated this approach in three different contexts. These illustrations are presented respectively in chapters IAKA (chapter 4), FRAKAS (chapter 5) and TAAABLE (chapter 6). At the end of each chapter, we have considered the current limitations of each approach and we have presented possible future work. In this discussion, we adopt a global view of the question we have dealt with during this research to discuss the more general prospects that will undoubtedly constitute a guideline for future research. Indeed, the preliminary results we have obtained so far, together with their links with other very recent research, encourage us to believe we are on a right track.

The first section of this chapter gives a synthesis of our contributions related to FIKA. Naturally, we want to enrich our work and our main objective is to contribute to the elaboration of a unified approach for knowledge acquisition in CBR. This unified approach involves the intelligent integration of various knowledge acquisition, or knowledge learning, strategies (manual, automatic, semi-automatic, on-line, off-line, etc.). We also consider the possibility of applying our work to other problem-solving approaches than CBR. We shall discuss these issues in section 7.2.

7.1 A synthesis of the FIKA related work

At first glance, IAKA and FRAKAS may seem very different, but in fact, they share the same principles. In chapter 3, we have proposed a comparison of IAKA and FRAKAS (cf. table 3.1, page 45) in order to highlight their differences. With table 7.1, we go back to this comparison to show how these differences are in fact only "different ways" of doing the same things, in different contexts.

	IAKA	FRAKAS	Common issues
Knowledge to acquire	Cases and adaptation knowledge	Domain knowledge	The targeted knowledge is used to support the adaptation process (and the retrieval, as adaptation knowledge is taken into account during the similarity assessment)
Type of solution	Best operational solution according to the knowledge available in the system (this entails the notion of quality of a solution with respect to a given problem)	Operational solution consistent with the system domain knowledge	The system always gives the "best solution" according to its "point of view", i.e. given its knowledge on the domain and its reasoning capabilities
Reasoning failure	The solution is not "good enough" according to a predefined set of criteria	The solution is inconsistent with respect to the expert's knowledge	In both cases, the solution is not satisfactory for the expert. This constitutes a failure, and the system has to acquire additional knowledge to avoid reproducing this failure
Interaction mode	The expert analyses each intermediate solution	The expert analyses the final solution	Interactions occurs on-line, during the problem-solving process
Reasoning mode	Decomposition of the reasoning process in several steps (intermediate problems)	Monolithic reasoning (black-box)	CBR principles are implemented: retrieval and adaptation are performed in the two approaches

Table 7.1: IAKA and FRAKAS: different methods and different tools to accomplish common objectives.

This table illustrates the fact that IAKA and FRAKAS implement different tools to reach common objectives (they are in some way "variations" around the main theme of FIKA). The specificities of the IAKA model make it better adapted to a certain category of knowledge acquisition problems, and the same remark is also true for FRAKAS. Hence, an attractive option for us is to find a way to combine IAKA, FRAKAS, and possibly other approaches, thus taking a step towards a more general knowledge acquisition support, where, depending on the problem, the better methodology will be automatically selected. This is a long term prospect for our work and, at the same time, it can be seen as a guideline for our present and future research.

In chapter 3, we have introduced a diagram to illustrate how we could organise our different contributions. This diagram is completed with possible future work in 7.1. IAKA and FRAKAS are two models which implement the FIKA principles. IAKA-NF is an implementation of the IAKA model with the numerical function formalism. This prototype has been tested with several functions, but other options are still possible. Moreover, the IAKA model could also be implemented in a further prototype with another knowledge representation formalism. In the same way, FRAKAS-PL implements the FRAKAS model with a simple propositional logic formalism. This formalism has the advantage of simplicity but can be too limited (in terms of expressiveness) in several situations. That is why we would like to develop a different application, based on another formalism, such as description logics, to further the capability of FRAKAS. Last but not least, as we have previously discussed, the main aim of this work is to investigate how we can combine different models in a common approach to facilitate the exploitation of the advantages of each approach depending on the context of the problem.

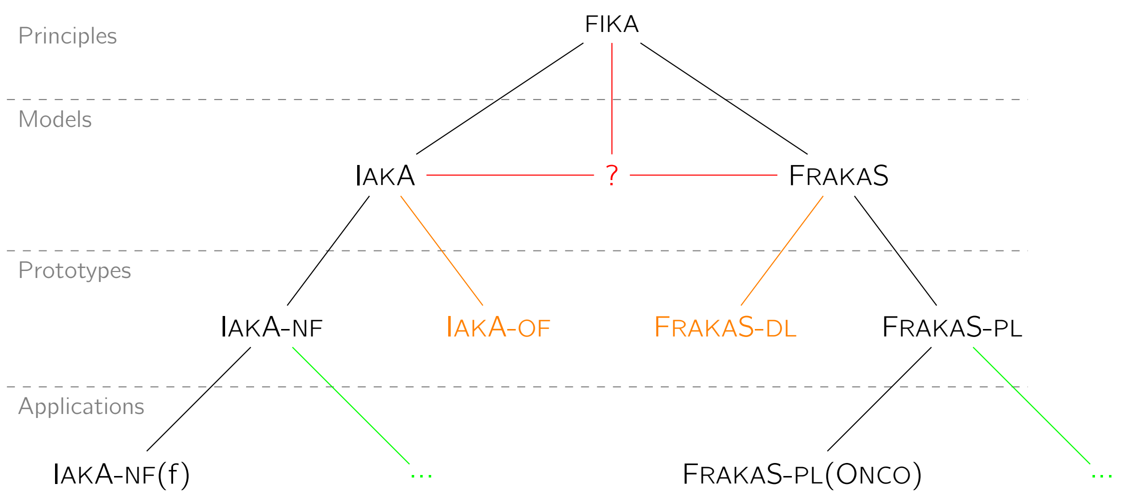


Figure 7.1: FIKA: synthesis and prospects.

7.2 Future work

7.2.1 Using interactions to guide automatic knowledge acquisition

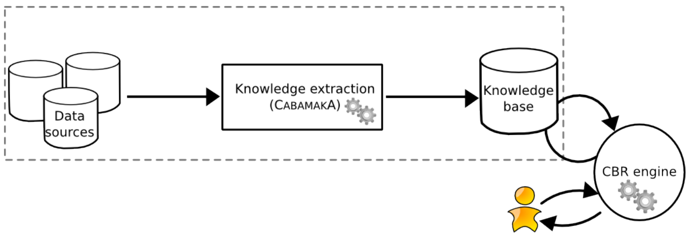
From the beginning of this work, we have made a clear distinction between a knowledge light approach and a knowledge intensive approach. A knowledge light approach makes use of the knowledge already available in the system to acquire new knowledge, whilst a knowledge intensive approach (sometimes called a *lazy-learning* approach) focuses on knowledge available outside the system. When trying to build a global approach for knowledge acquisition, it becomes clear that there is obviously space for both these general knowledge acquisition approaches. Indeed, a significant amount of knowledge can be efficiently acquired with automatic, or semi-automatic approaches, thus alleviating the modelisation effort. Nevertheless, interactive approaches are still useful and efficient for capturing knowledge that was missed by the other strategies. However, we can also find another way of integrating interactive and semi-automatic approaches.

As an example, we can consider the CABAMAKA application [d'Aquin et al., 2007]. CABAMAKA is a system that allows the extraction of adaptation rules by applying data-mining techniques on a case-base. All the available pairs of cases in the case base are considered, making the assumption that one case is the result of the adaptation of the other by a given transaction, corresponding to the application of adaptation rules. Adaptation patterns are extracted using a specific algorithm that looks for similarities between these transactions. The knowledge representation formalism used in CABAMAKA is that of description logics. Thus, the system is able to process symbolic descriptors. A recent work on the representation of the variations between cases will allow the system to process numerical data as well [Badra and Lieber, 2008].

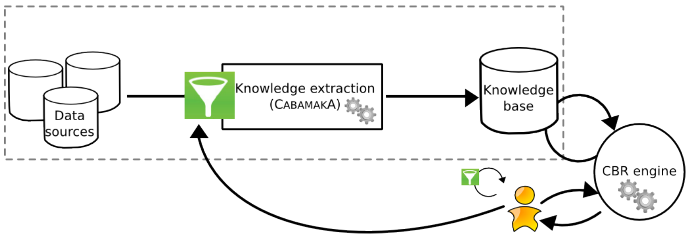
Compared to fully-automatic approaches, the advantage of CABAMAKA is that it produces a smaller number of adaptation rules that need to be validated by the expert. However, this number is still large and the expert's task remains tedious. The number of rules to analyse could decrease if the expert had the ability to "configure" the research of adaptation rules according to his needs.

We believe that it is possible to use an interactive approach to help the user configure the adaptation rules extraction process according to his needs. Figure 7.2 summarises this idea. The upper part of the figure describes the current CABAMAKA process: data sources (databases, files, etc.) are used by a knowledge extraction process which produces knowledge stored in a knowledge base after validation by expert. This knowledge base is used by a CBR engine to solve problems. The lower part of the figure illustrates how this process can be transformed by integrating an interactive dimension into the process. The main idea is to exploit reasoning failures to build "filters" that will be used as parameters to configure the knowledge extraction process. The belief is that it might be easier to build these filters based on a previous example, i.e., in context, rather than building them from scratch.

This idea is an interesting illustration of how knowledge light and knowledge intensive approaches complement each other.



(a) The CABAMAKA framework



(b) A possible extension of the CABAMAKA framework

In the CABAMAKA framework, knowledge is extracted during an off-line process supervised by the user. Then, this knowledge is used by the CBR engine. A possible extension of the CABAMAKA framework could exploit failures of the CBR process to help the user building filters that will be used to improve the efficiency of the knowledge extraction mechanism.

Figure 7.2: Combining on-line and off-line knowledge acquisition.

7.2.2 Trace-based reasoning: how to really include context in CBR?

According to the traditional definition of the CBR paradigm, identical problems should have identical solutions. So what if identical problems, from the system's point of view, have different solutions from the user's point of view?

To illustrate this issue, we return to the example of Flatland, introduced in chapter 4, page 51 (for further reading, this example is developed in [Mascret, 2008]). Let us assume that several rules govern the Flatland world:

- The *wedding rule*, where two shapes give birth to a child whose characteristics depend on the characteristics of its parents;
- The *fighting rule*, where two shapes produce a shape that has the same number of edges as the shape with the highest number of edges;
- The *coalition*, where two shapes produce a shape that has a number of edges equal to the sum of edges of the two shapes.

These rules are summarised in figure 7.3.

In this example, when we have to evaluate what will happen when two individuals meet, we need to know what is the purpose of the meeting. Is it going to be a wedding, a fight or a coalition? In such a situation, we need a *context* information in order to be able to trigger the reasoning process. If this context information is not available in the problem description and in the cases of the case base, the adaptation process might lead to a failure.

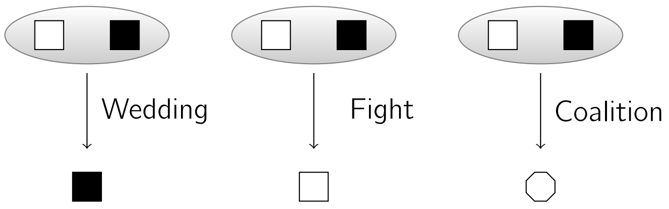
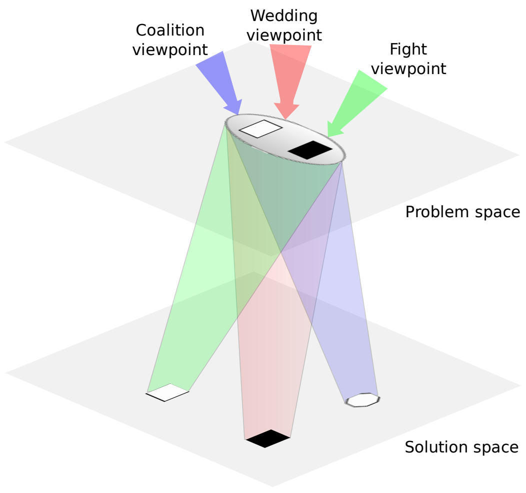


Figure 7.3: Flatland alternative rules.

This context information is a point of view on the problem, and it is because of this point of view that what we think are "similar" problems might in fact have different solutions. This concept of point of view of the user on the problem is illustrated in figure 7.4.



Three viewpoints (wedding, fight and coalition) on the same problem give three different "solutions".

Figure 7.4: Viewpoints in Flatland.

One could argue that the two cases above are in fact not identical, because they correspond to two different problems. If the system finds them identical, it is because they are ill-defined: the system misses elements which would enable it to make a distinction between, for example, a wedding problem and a coalition problem. In order to avoid this issue, the problem has to be properly defined. But we also need to consider the cases of the case base: is this context information available in the cases? If not, the system will not be able to reuse these cases. One solution could be to define the cases in the case base in order to add this piece of information, but this might be very difficult, if not impossible, because generally, the memory of the context in which the problem occurred is lost.

To summarise, the problem here is that case-based reasoning suffers from the "frame problem": in some situations, the context information is missing. In CBR, problem-solving episodes are often considered independently from the context in which they occur. Moreover, as cases are usually represented in a fixed way, with a static vocabulary and at a given granularity, it is very difficult to take the context information into account if it has not been anticipated.

In our opinion, moving from case-based reasoning to trace-based reasoning (TBR) could be a valuable way to overcome this issue. Trace-based reasoning, as defined in [Mille, 2006a] and [Mille, 2006b] can be viewed as an extension, or a generalisation, of the CBR paradigm, allowing the context to be taken into account in the reasoning. Trace-based reasoning is based on the observation that human experience is, by definition, temporally situated. In TBR, traces are defined as "records" of the human activity, and context is naturally embodied in these traces.

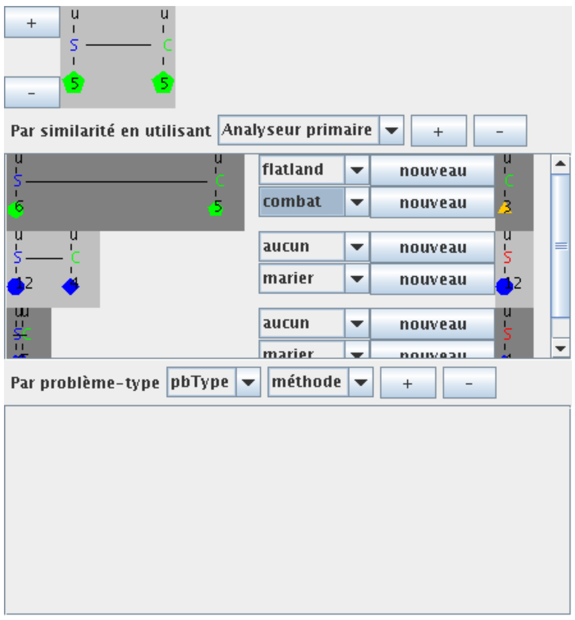
The first trace model has been described in [Champin et al., 2003]. In this framework called MUSETTE, the authors have defined the concept of "trace" and have shown how a trace can be represented, which kind of computation can be performed on traces and how traces can be used to retrieve useful past experiences. Since then, several pieces of research have been conducted both on the formalisation of the trace model as a way to manage knowledge and experience ([Laflaquière et al., 2006]) and on the possible applications of trace-based reasoning ([Georgeon et al., 2007], [Georgeon, 2008]).

In order to develop TBR applications, one must deal with several issues:

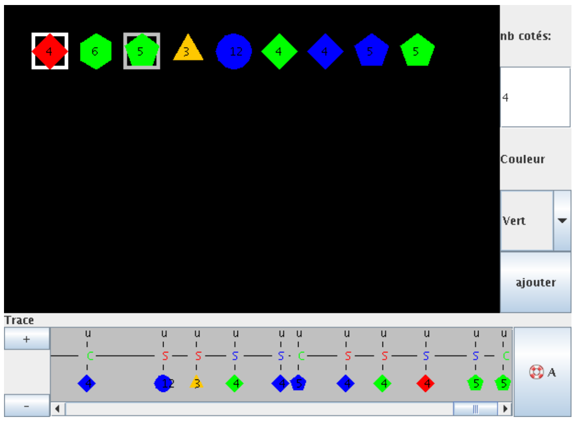
- How to identify relevant episodes in the trace?
- How to compare different episodes, which involves defining similarity measures applicable on traces and not only on cases.
- How to deal with episodic and temporal aspects of the traces?
- How to identify relevant context information, among other things, in traces; and how to make sure all the elements we need are in the trace?
- How to deal with the unstructured nature of the trace?

Some of these issues are open research paths to follow, others are currently studied within our research team. For example, the definition of a trace-based system is an active field of investigation (see, for example, [Settouti et al., 2007]). More specifically, in [Ben Saad, 2008], the author has worked on a definition of a similarity measure applicable to temporal sequences and has shown how the use of this measure can support knowledge discovery in interaction traces. In [Mascret, 2008], Mascret has worked on the use of traced experience to facilitate knowledge acquisition in problem-solving systems. The targeted application domain of this work is that of assistance tools. For this purpose, Mascret has made use of the IAKA model and has applied it to the TBR context. He has also developed a prototype application inspired by the Flatland domain, but based on traces. Figure 7.5 shows an overview of the interface of this prototype.

This work is also linked to that of Stuber [Stuber, 2007] who has proposed an assistance system based on trace-based reasoning. In his work, Stuber has considered the knowledge acquisition issue, but he was mainly focused on experience sharing between users, using a system of *alter-ego*. The work of Cram is in the same vein [Cram et al., 2007b], [Cram et al., 2007a], [Cram et al., 2008]. He works on the reuse of interaction traces on collaborative environments to support and facilitate collaborations between users.



(a) Similarity assessment and adaptation methods



(b) Display of the Flatland trace

This assistant prototype, though based on traces, strongly exploits case-based reasoning principles. Moreover, it implements adaptation methods and adaptation operator. The first illustrations shows how adaptation methods are to be taken into account when evaluating the similarity. Then, the elements of the trace that are considered in this example are highlighted.

Figure 7.5: A trace-based assistant based on Flatland.

Hence, work with regard to the TBR issue is an active research field within our team. From the CBR viewpoint, TBR is in our opinion, an opportunity to go one step further with the robust and efficient CBR paradigm. Traces offer the possibility to build dynamically new case structure and to extend the context of cases if necessary. Hence, traces offer the ability to easily put experiences "in context". Therefore, if we have the ability to extract cases from traces, then we will be able to reuse all the know-how we have in CBR in contextualised situations.

We consider TBR as a promising research direction because of all its possible applications, in particular in the context of collaborative applications, and semantic web. Moreover, TBR could be a way to facilitate the use of the web as a source for knowledge acquisition and, as it has been shown in [Leake and Powell, 2007], this is an open research field.

Finally, we believe that some of the principles we defend, in particular the notion of feedback of the user through interactions, should be related to the recent work on case provenance [Leake and Whitehead, 2007]. Recent papers have proposed applications of the concept of provenance to exploit feedback in CBR ([Leake and Dial, 2008], [Leake and Kendall-Morwick, 2008], [Leake and Powell, 2008] and [Briggs and Smyth, 2008]). We are convinced that links can be made between FIKA and the concept of case provenance and we are going to explore this research subject.

Le système est une mémoire d'usages, qui sédimente les expériences passées pour penser les usages à venir.

Bruno Bachimont

8 CONCLUSION

The contributions we have presented in this manuscript are twofolds. They are intended as an answer to the theoretical question of knowledge acquisition in CBR and the answer to certain practical requirements expressed by CBR system designers. When applying their systems to real conditions, with the assistance of experts in the domain concerned, they came up against concrete problems of knowledge acquisition in numerous fields of application.

Our research team possesses a wide experience in the design and implementation of applications utilising CBR principles which are used in industry. For example we may mention PADIM [Fuchs et al., 1995], [Fuchs, 1997], [Mille et al., 1999]; PROLABO [Mille et al., 1996]; RADIX [Corvaisier et al., 1997], [Corvaisier et al., 1998]; ACCELERE [Herbeaux, 2000], [Herbeaux and Mille, 2007]; and PIXED [Heraud and Mille, 2000].

We have also had exchanges with collaborators who have developed their own systems for particular applications. KASIMIR [Lieber et al., 2002], [Lieber et al., 2005] for example, is used daily in the field of breast cancer treatment. These colleagues have also met similar difficulties regarding the acquisition of knowledge. The feedback of experiences which system designers and developers shared with us was the main inspiration and motivation for our work.

In order to gain a better understanding of our method, let us review the questions and thought processes which preceded it, with an illustration of a typical conversation we might have had at the beginning of this research.

CBR Designer: CBR systems do no "construct" solutions to problems, they adopt existing solutions. This is indeed their main strength.

Candide: Why?

CBR Designer: Because, in order to build a solution, it is necessary to possess all the knowledge of the domain concerned. If part of it is missing, we cannot reach the end of the process. On the contrary, if we have an existing solution, even if we don't know all its aspects, we can at least use what is already there. It is true that in CBR, we are not guaranteed to have the best solution in the end (supposing it exists...), but at least we do have a solution, which most of the time is quite sufficient. The user of the system is always happy to have this solution as a source of inspiration. It is a solution which can, in some cases, help him to solve the problem himself.

Candide: I see. The advantage of CBR systems compared to rule-based systems is that they always give a solution. From the system's point of view, it is "the best possible solution", which may not necessarily be the opinion of the expert.

CBR Designer: That's it. The system does its best, given its knowledge and reasoning capacity, to give what it considers the best solution. It uses its reasoning capacity to adapt solutions it already knows to the user's or the expert's problem.

Candide: So the strong point of CBR is the adaptation stage.

CBR Designer: Exactly. But the problem is that adaptation is a really difficult process in some respects.

Candide: So, if we want to improve CBR systems, all we have to do is improve the adaptation mechanisms?

CBR Designer: No. Generally, these mechanisms are good. Or, at least, we don't want to question them.

Candide: So if reasoning mechanisms are good, where's the problem?

CBR Designer: The problem is that the knowledge used to feed these inferences is not necessarily suitable. And what happens if we start from the wrong hypotheses?

Candide: Well, we reach the wrong conclusions, don't we?

CBR Designer: That's exactly right.

Candide: In that case, we just have to amend this knowledge.

CBR Designer: It's not that simple. The knowledge is not necessarily wrong... Sometimes the problem is that it is not "applicable" to the situation being considered. For example, most of the time it's a good idea to use salt to whisk up a meringue, but is some cases, it's better to use lemon.

Candide: But how do we know?

CBR Designer: The expert knows. He has practical knowledge linked to his own experience. For example, he knows that when making a white chocolate mousse, it is better to use lemon than salt.

Candide: So why didn't he model this into the system knowledge when designing it?

CBR Designer: I don't know, ask him yourself.

Domain expert: I did not do it because I didn't think of it. We can't think of everything. And the knowledge engineer who asked me so many questions never asked me! To be honest, I had never realised that I used lemon in some cases, so when it comes to mentioning it to the system designer... It's only when the system suggested I made a white chocolate mousse with salt in the egg whites that I realised the problem.

Candide: So if I understand it, you realised you could solve the problem only when it arose?

Domain expert: I suppose so.

Candide: And what happened?

Domain expert: I mentioned it to the knowledge engineer. He said that he was going to "model this knowledge" into the system, and I suppose he made a good job of it because the problem never occurred again. Well, it did not occur again in this context, anyway... Having said that, it might be even more efficient if I could tell the system what to do in such a case.

Candide: What prevented you?

Domain expert: I am only a user of the system, you know; it helps me solve problems by saving time, but I can't communicate with it.

Candide: Is there no possible interaction?

CBR Designer: No, not really. We have to go through a knowledge modelling stage every time we want the system to evolve. It is often complicated and we seldom take the time to do it. And the experts are not often available...

Domain expert: Oh !

This is how we might summarise the preliminary considerations which were at the start of our work. We based ourselves on the study of existing work as well as on feedback from system designers who were confronted with concrete problems. In so doing, we were able to identify the problems inherent to the acquisition of practical knowledge, to define priorities and to highlight certain needs, in particular the need to formalise knowledge.

In answer to these various questions, we proposed FIKA, an interactive and opportunistic approach for the acquisition of case-based reasoning knowledge. Interactive, because it gives the user, expert in his field, the main role in the process; opportunistic, because it triggers the knowledge acquisition process at the most opportune moment, that is when a "reasoning failure" occurs. Our intention in this work was to propose a form of support for the delicate process of knowledge acquisition in case-based reasoning. In order to do this, we decided to place the expert at the centre of the process since our hypothesis is that this constitutes a favourable context for the acquisition of a certain type of knowledge, i.e. knowledge linked to experience. Thus, in FIKA, the user plays a crucial part by participating actively in the building of knowledge as it will appear in the system. An immediate consequence of this process is that the knowledge handled by the system make sense to the user and "system-user" interactions become easier. Progressively, the system acquires new knowledge. In the meantime, by observing his own practice, the expert finds out that he is able to build "knowledge" from his "know-how" and to share this knowledge to the system. Expert and system are then at the heart of a virtuous circle where each one learns by interaction with the other, with a common aim: to solve the problem under study.

In order to illustrate the input of FIKA when it is used in CBR applications, we have developed two applications: IAKA-NF and FRAKAS-PL. The studies made into these applications have given rise to more general reflections which led to the definition of two models: IAKA and FRAKAS, which can be viewed as possible implementations of FIKA under certain conditions. In other words, these models are "variations" around the theme of FIKA. At first sight, IAKA and FRAKAS may appear different, but in fact they share the same principles: interactivity, opportunism, user-focus, and the same hypotheses, implied in one and explained in the other. From the implementation viewpoint, IAKA and FRAKAS show a few distinctive aspects. In IAKA, the main interactive learning targets are cases and adaptation knowledge, whereas in FRAKAS, one seeks to acquire "domain" knowledge which is still used in adaptation. IAKA handles adaptation knowledge which is organised into methods and adaptation operators, and each piece of knowledge can be the subject of a specific adaptation process. In FRAKAS, the domain knowledge is used to guide the adaptation, even if there is no specific representation of adaptation knowledge. In IAKA, adaptation is broken into adaptation stages, and when reasoning is to be analysed, the expert observes each stage independently. In FRAKAS, adaptation is considered as a black box and the expert analyses only the "final" result of the adaptation, not the intermediary problems. Finally, the role of adaptation knowledge in determining similarity is obvious in IAKA: it appears during the building of the similarity path. In FRAKAS, adaptation knowledge enables us to find a case which will be used as support for the adaptation process. Thus IAKA and FRAKAS are built on a common architecture and their differences only make them better adapted to particular contexts.

IAKA-NF and FRAKAS-PL were developed to demonstrate the properties of the FIKA approach. Several experiments have been made with these prototypes, sometimes using a virtual expert. The results of these experiments demonstrate the validity of the approach and highlight the benefits of a support for the knowledge acquisition process in CBR systems. Nevertheless, both these applications have been specifically developed to test FIKA's general strategy, and one might question its usability in other contexts. As an initial response to this remark, we have applied the FIKA principles to knowledge acquisition within an existing application which had not been developed for this purpose: TAAABLE.

TAAABLE is a web application based on a case based reasoning engine. Its role is to propose cooking recipes in reply to a request formulated by a user involving an available recipe book and a quantity of knowledge which permits the adaptation of these recipes. This application has won the title of "European Vice Champion" at the first Computer Cooking Contest, a cooking application contest organised in September 2008 as part of the European CBR conference. In TAAABLE, we have used the FIKA principles to support the process of editing initially acquired knowledge (in an off-line process), but we are considering the integration of these principles in an online phase to offer the user some support during an interactive process of recipe adaptation. The use of FIKA in TAAABLE illustrates the portability of this approach.

As we have observed from the very beginning of this study, the formalisation of the delicate process of knowledge acquisition in CBR constitutes an important and topical subject of research. Research conducted in this thesis provides a step towards this formalisation. Although our various studies are still exploratory, the results we have obtained in different conditions with each of our systems has proved encouraging. Nevertheless, they must be viewed with care as the conditions of our experiments did not allow testing in the field. This is why one of the aims in this work is to continue along these lines and to apply FIKA to larger scale applications. This objective opens numerous prospects for study: implementation of our models in existing systems, general reflection on the interoperability of the various approaches we have proposed, study of the portability of our approach in other knowledge engineering domains.

Knowledge acquisition in IAKA, FRAKAS and TAAABLE as well as FIKA, is generally a thoroughly "knowledge intensive" process (i.e. we are seeking to acquire knowledge from the outside world) and differs therefore from automatic knowledge acquisition methods using existing knowledge (so-called knowledge light approaches). However, FIKA does not present itself as an alternative to these techniques. On the contrary, these approaches complement each other. FIKA is a strategy which is particularly efficient where knowledge light approaches fail, which is often the case when one seeks to acquire precise knowledge linked to experience. Therefore we naturally envisage as an outcome of this work the definition of a general strategy of CBR knowledge acquisition which would integrate various approaches and exploit the best of each according to the problem to be dealt with. This perspective should constitute an important challenge for CBR research. The improvement of the knowledge acquisition process is indeed a necessary stage towards a better understanding of what CBR really is and we think that mastery of this process can only facilitate the design of even more efficient CBR systems.

Appendix

A

FRAKAS, THE CASE OF JULIE

A.1 Target problem

Julie is a woman suffering from breast cancer. She has positive hormone receptors `HR+`, she has already had a radical mastectomy with a lymph node dissection `Patey-done` (Patey is the name of a surgical act), with no involved lymph nodes `N-`. Julie is also suffering from a liver disease denoted by `liver-disease`. A set of other characteristics, denoted by `c` compiles pieces of information about her gender, her age, her tumor size, etc.).

This patient corresponds to the following problem:

$$\mathtt{tgt} = \mathtt{HR+} \land \mathtt{Patey\text{-}done} \land \mathtt{N\text{-}} \land \mathtt{liver\text{-}disease} \land \mathtt{c} \qquad \text{(Target problem)}$$

A.2 Vocabulary

$$\begin{aligned}
\mathcal{V}^c_{\mathtt{pb}} &= \{\mathtt{HR+}, \mathtt{Patey\text{-}done}, \mathtt{N\text{-}}, \mathtt{liver\text{-}disease}, \mathtt{c}, \gamma\} \\
\mathcal{V}^c_{\mathtt{sol}} &= \{\mathtt{tamoxifen}, \mathtt{anti\text{-}aromatases}, \mathtt{ovary\text{-}ablation}, \\
&\qquad \mathtt{FEC}, \mathtt{fluorouracil}, \mathtt{epirubicin}, \mathtt{cyclophosphamide}\} \\
\mathcal{V}^a_{\mathtt{sol}} &= \{\mathtt{anti\text{-}oestrogens}\}
\end{aligned}$$

A.3 Initial domain knowledge

The initial domain knowledge is defined as follows:

$$
\begin{aligned}
SDK = {} & \gamma \to \mathtt{c} \wedge && \text{(A.1)}\\
& \mathtt{liver\text{-}disease} \to \neg\mathtt{tamoxifen} \wedge && \text{(A.2)}\\
& (\mathtt{tamoxifen} \vee \mathtt{anti\text{-}aromatases} \vee \mathtt{ovary\text{-}ablation}) &&\\
& \leftrightarrow \mathtt{anti\text{-}oestrogen} \wedge && \text{(A.3)}\\
& \mathtt{FEC} \leftrightarrow (\mathtt{fluorouracil} \wedge \mathtt{epirubicin} \wedge \mathtt{cyclophosphamide}) && \text{(A.4)}
\end{aligned}
$$

In the initial domain knowledge, line A.1 indicates that a patient having the characteristics represented by γ has the characteristics represented by $\mathtt{c}$. Line A.2 indicates that tamoxifen is contraindicated for people having a liver disease. Line A.3 indicates that tamoxifen, anti-aromatases, and ovary ablation are anti-oestrogen treatments and that they are the only available anti-oestrogen treatments (in the context of a given hospital). Finally, line A.4 indicates that a FEC treatment is recommended if and only if fluorouracil, epirubicin, and cyclophosphamide are recommended (FEC is composed of three chemotherapy drugs: fluorouracil, epirubicin, and cyclophosphamide).

A.4 Source case

The retrieved source case $\mathtt{srce} \wedge \mathtt{Sol(srce)}$ is the following:

$$
\begin{aligned}
\mathtt{srce} &= \mathtt{HR+} \wedge \mathtt{Patey\text{-}done} \wedge \mathtt{N-} \wedge \mathtt{c}\\
\mathtt{Sol(srce)} &= \mathtt{FEC} \wedge \mathtt{tamoxifen} && \text{(Source case)}
\end{aligned}
$$

This case describes a protocol rule that can be formalised as: "If the patient has positive hormone receptors, has already had Patey surgery, no involved lymph nodes, and some other characteristics not detailed here (denoted by $\mathtt{c}$), then a treatment composed of a chemotherapy based on FEC and a hormone therapy based on tamoxifen is recommended."

A.5 Adaptation

Since $\mathtt{tgt}$ corresponds to a specific situation of the general case $\mathtt{srce}$ ($\mathtt{tgt} \vDash_{SDK} \mathtt{srce}$), the source case $(\mathtt{srce}, \mathtt{Sol(srce)})$ is selected by the retrieval process. However, a straightforward application of this source case contradicts the target problem, given the domain knowledge: $SDK \wedge \mathtt{srce} \wedge \mathtt{Sol(srce)} \wedge \mathtt{tgt}$ is unsatisfiable (since $SDK \wedge \mathtt{liver\text{-}disease} \wedge \mathtt{tamoxifen}$ is). The $\circ_D$-conservative adaptation gives:

$$
\begin{aligned}
&\mathtt{CA}_{\circ_D}(SDK, \mathtt{srce} \wedge \mathtt{Sol(srce)}, \mathtt{tgt})\\
&\equiv_{SDK} \mathtt{tgt} \wedge \underbrace{\mathtt{FEC} \wedge (\neg\mathtt{tamoxifen} \wedge (\mathtt{anti\text{-}aromatases} \vee \mathtt{ovary\text{-}ablation}))}_{\mathtt{Sol(tgt)}}
\end{aligned}
$$

Tamoxifen, since it is contraindicated for the target patient, is removed, but according to the conservative adaptation principle, as much as possible from $\mathtt{Sol(srce)}$ is kept. In particular,

$\mathtt{tamoxifen} \vDash_{SDK} \mathtt{anti\text{-}oestrogen}$ and $\mathtt{anti\text{-}oestrogen}$ is kept ($\mathtt{tgt} \wedge \mathtt{anti\text{-}oestrogen}$ is consistent). Thus, $\mathtt{anti\text{-}aromatases} \vee \mathtt{ovary\text{-}ablation}$ is proposed, following the element (line A.3) of the domain knowledge:
$\neg\mathtt{tamoxifen} \wedge \mathtt{anti\text{-}oestrogen} \vDash_{(A.3)} \mathtt{anti\text{-}aromatases} \vee \mathtt{ovary\text{-}ablation}$.

Therefore, the hormone therapy recommended for the target patient according to $\mathtt{Sol}(\mathtt{tgt})$, is a cure of anti-aromatases or an ovary ablation.

In this example, the source case $\mathtt{srce} \wedge \mathtt{Sol}(\mathtt{srce})$ introduced above is adapted to solve $\mathtt{tgt}$. The conservative adaptation produces a set of interpretations. Each interpretation is a possible solution of $\mathtt{tgt}$.

Figure A.1(a) shows the result of the conservative adaptation performed according to SDK and the feedback of the expert in form of checked boxes. It can be observed that the expert indicates an inconsistency between the proposed solution and his knowledge: $\mathtt{epirubicin}$ is inconsistent with $\mathtt{liver\text{-}disease}$. The system uses this feedback to acquire a new piece of knowledge. In figure A.1(b), the confirmation screen is presented: the acquired piece of knowledge is expressed in propositional logic and the expert can add a textual explanation before allowing the system to learn this knowledge. Textual explanations may be used off-line to acquire more domain knowledge (see section 5.4.1 for more information on textual explanation).

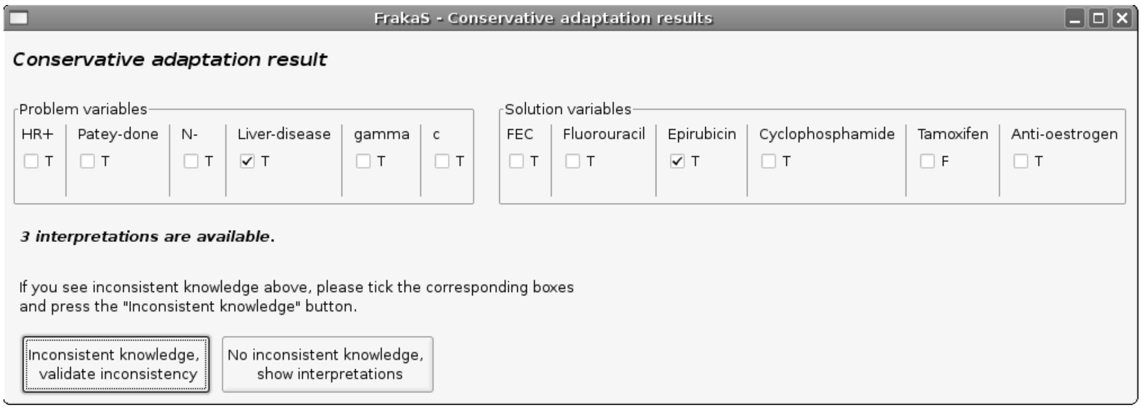
Thus, after this step, the domain knowledge SDK has evolved into SDK':

$$\begin{aligned} SDK' &= SDK \wedge \neg(\mathtt{liver\text{-}disease} \wedge \mathtt{epirubicin}) \\ &\equiv SDK \wedge (\mathtt{liver\text{-}disease} \rightarrow \neg\mathtt{epirubicin}) \end{aligned}$$

Then, a new conservative adaptation is performed with SDK':

$$\begin{aligned} &\mathtt{CA}_{\circ_D}(SDK', \mathtt{srce} \wedge \mathtt{Sol}(\mathtt{srce}), \mathtt{tgt}) \\ &\equiv_{SDK'} \mathtt{tgt} \wedge \\ &\underbrace{\begin{pmatrix} \neg\mathtt{FEC} \wedge \mathtt{fluorouracil} \wedge \neg\mathtt{epirubicin} \wedge \mathtt{cyclophosphamide} \\ \wedge\, \mathtt{anti\text{-}oestrogen} \wedge \neg\mathtt{tamoxifen} \wedge (\mathtt{anti\text{-}aromatases} \vee \mathtt{ovary\text{-}ablation}) \end{pmatrix}}_{\mathtt{Sol}(\mathtt{tgt})} \end{aligned}$$

Since they are both inconsistent with $SDK' \wedge \mathtt{tgt}$, the system does not keep $\mathtt{epirubicin}$ and $\mathtt{FEC}$ in the new proposition. As the expert validates the first part of the proposition (common variables), the system displays the three possible interpretations (cf. figure A.2). Since each interpretation is correct, the expert validates the proposition.


FrakaS - Conservative adaptation results
Inconsistent knowledge, validate inconsistency
No inconsistent knowledge, show interpretations

Conservative adaptation result

Problem variables

HR+	Patey-done	N-	Liver-disease	gamma	c
☐ T	☐ T	☐ T	✓ T	☐ T	☐ T

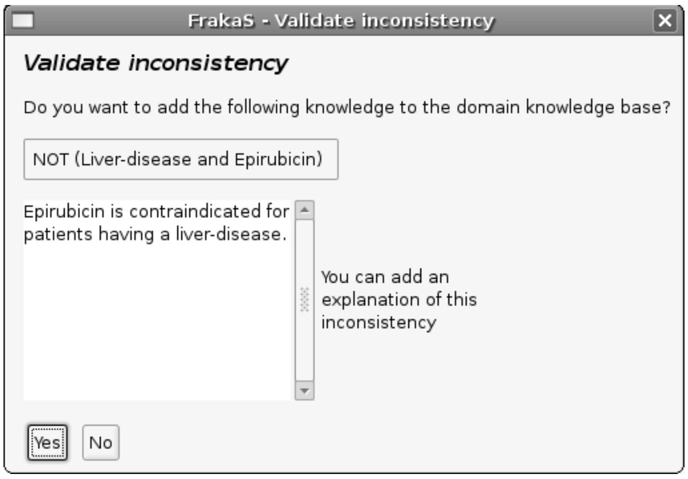
Solution variables

FEC	Fluorouracil	Epirubicin	Cyclophosphamide	Tamoxifen	Anti-oestrogen
☐ T	☐ T	✓ T	☐ T	☐ F	☐ T

3 interpretations are available.

If you see inconsistent knowledge above, please tick the corresponding boxes and press the "Inconsistent knowledge" button.

(a) Result of the first conservative adaptation and feedback of the expert in form of checked boxes: the expert points out an inconsistency between the domain knowledge and his knowledge.



Validate inconsistency

Do you want to add the following knowledge to the domain knowledge base?

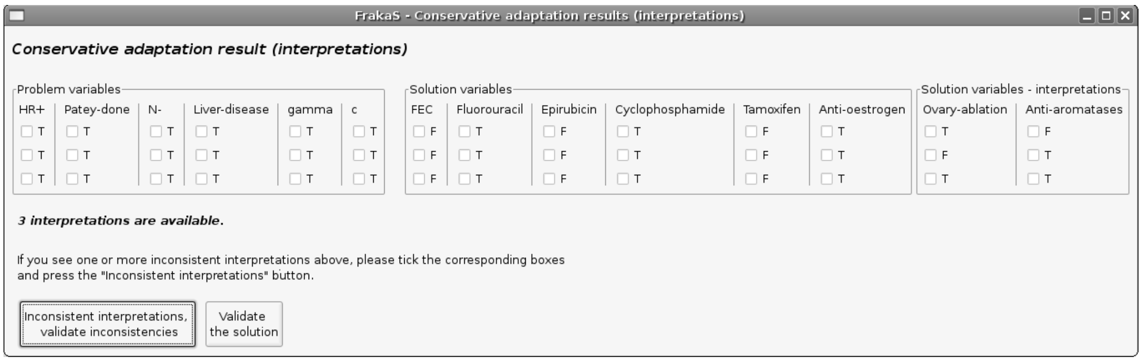
NOT (Liver-disease and Epirubicin)

Epirubicin is contraindicated for patients having a liver-disease.

You can add an explanation of this inconsistency

(b) The expert provides an explanation in plain text.

Figure A.1: First solution presented to the expert and his feedback (a). Plain text explanation provided by the expert (b).


FrakaS - Conservative adaptation results (interpretations)
Inconsistent interpretations, validate inconsistencies
Validate the solution

Conservative adaptation result (interpretations)

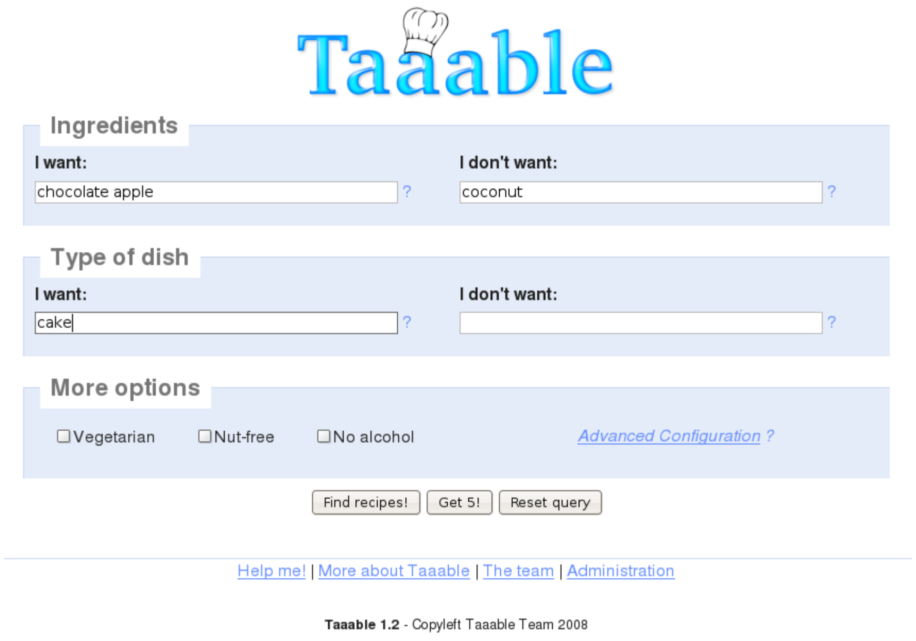
Problem variables						Solution variables						Solution variables - interpretations	
HR+	Patey-done	N-	Liver-disease	gamma	c	FEC	Fluorouracil	Epirubicin	Cyclophosphamide	Tamoxifen	Anti-oestrogen	Ovary-ablation	Anti-aromatases
☐ T	☐ T	☐ T	☐ T	☐ T	☐ T	☐ F	☐ T	☐ F	☐ T	☐ F	☐ T	☐ T	☐ F
☐ T	☐ T	☐ T	☐ T	☐ T	☐ T	☐ F	☐ T	☐ F	☐ T	☐ F	☐ T	☐ F	☐ T
☐ T	☐ T	☐ T	☐ T	☐ T	☐ T	☐ F	☐ T	☐ F	☐ T	☐ F	☐ T	☐ T	☐ T

3 interpretations are available.

If you see one or more inconsistent interpretations above, please tick the corresponding boxes and press the "Inconsistent interpretations" button.

Figure A.2: Proposed solution presented to the expert, who validates it.

B
TAAABLE


Ingredients

I want: chocolate apple

I don't want: coconut

Type of dish

I want: cake

I don't want:

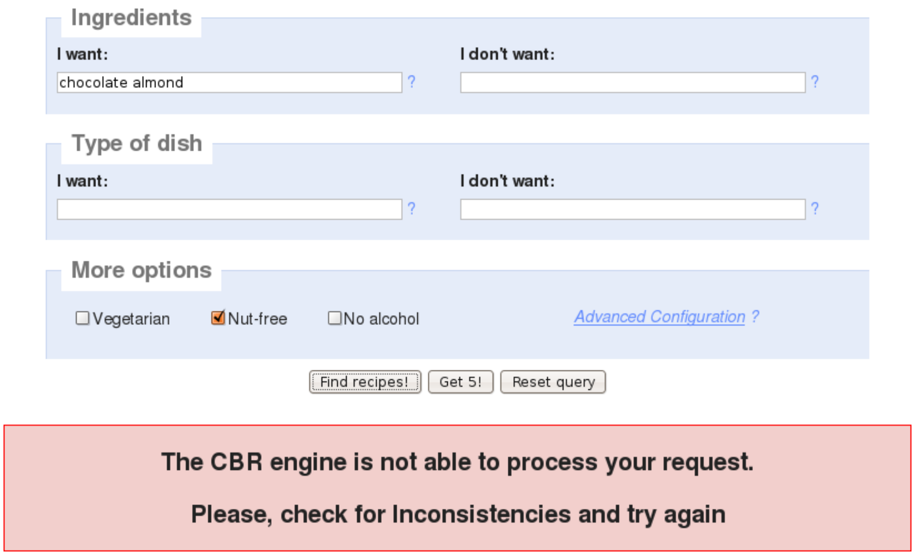
More options

☐ Vegetarian ☐ Nut-free ☐ No alcohol

Advanced Configuration ?

Taaable
?
Find recipes!
Get 5!
Reset query
Help me! | More about Taaable | The team | Administration
Taaable 1.2 - Copyleft Taaable Team 2008

Figure B.1: A request is being entered in TAAABLE.



Ingredients

I want:	I don't want:
chocolate almond	

Type of dish

I want:	I don't want:

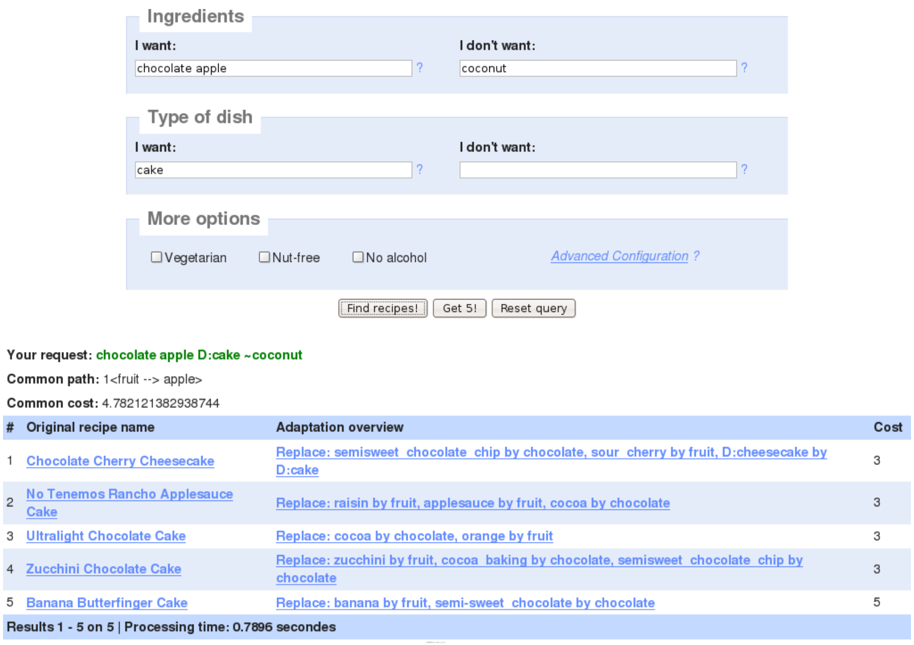
More options

☐ Vegetarian ☑ Nut-free ☐ No alcohol

The CBR engine is not able to process your request.

Please, check for Inconsistencies and try again

Figure B.2: Warning in case of error in the entered request.



Ingredients

I want:	I don't want:
chocolate apple	coconut

Type of dish

I want:	I don't want:
cake	

More options

☐ Vegetarian ☐ Nut-free ☐ No alcohol

Your request: chocolate apple D:cake ~coconut

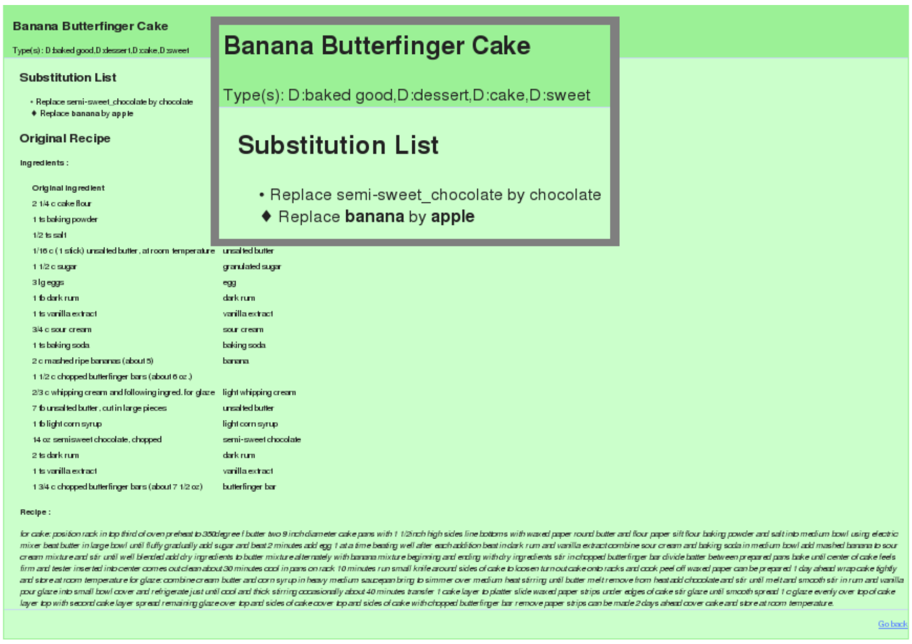
Common path: 1<fruit --> apple>

Common cost: 4.782121382938744

#	Original recipe name	Adaptation overview	Cost
1	Chocolate Cherry Cheesecake	Replace: semisweet_chocolate_chip by chocolate, sour_cherry by fruit, D:cheesecake by D:cake	3
2	No Tenemos Rancho Applesauce Cake	Replace: raisin by fruit, applesauce by fruit, cocoa by chocolate	3
3	Ultralight Chocolate Cake	Replace: cocoa by chocolate, orange by fruit	3
4	Zucchini Chocolate Cake	Replace: zucchini by fruit, cocoa_baking by chocolate, semisweet_chocolate_chip by chocolate	3
5	Banana Butterfinger Cake	Replace: banana by fruit, semi-sweet_chocolate by chocolate	5

Results 1 - 5 on 5 | Processing time: 0.7896 secondes

Figure B.3: Display of results for a request.



Banana Butterfinger Cake

Type(s) : D:baked good,D:dessert,D:cake,D:sweet

Substitution List

- Replace semi-sweet_chocolate by chocolate
- ♦ Replace banana by apple

Original Recipe

Ingredients :

Original Ingredient	
2 1/4 c cake flour	
1 ts baking powder	
1/2 ts salt	
1/16 c (1 stick) unsalted butter, at room temperature	unsalted butter
1 1/2 c sugar	granulated sugar
3 lg eggs	egg
1 tb dark rum	dark rum
1 ts vanilla extract	vanilla extract
3/4 c sour cream	sour cream
1 ts baking soda	baking soda
2 c mashed ripe bananas (about 5)	banana
1 1/2 c chopped butterfinger bars (about 6 oz.)	
2/3 c whipping cream and following ingred. for glaze	light whipping cream
7 tb unsalted butter, cut in large pieces	unsalted butter
1 tb light corn syrup	light corn syrup
14 oz semisweet chocolate, chopped	semi-sweet chocolate
2 ts dark rum	dark rum
1 ts vanilla extract	vanilla extract
1 3/4 c chopped butterfinger bars (about 7 1/2 oz)	butterfinger bar

Recipe :

for cake: position rack in top third of oven preheat to 350degree f butter two 9 inch diameter cake pans with 1 1/2inch high sides line bottoms with waxed paper round butter and flour paper sift flour baking powder and salt into medium bowl using electric mixer beat butter in large bowl until fluffy gradually add sugar and beat 2 minutes add egg 1 at a time beating well after each addition beat in dark rum and vanilla extract combine sour cream and baking soda in medium bowl add mashed banana to sour cream mixture and stir until well blended add dry ingredients to butter mixture alternately with banana mixture beginning and ending with dry ingredients stir in chopped butterfinger bar divide batter between prepared pans bake until center of cake feels firm and tester inserted into center comes out clean about 30 minutes cool in pans on rack 10 minutes run small knife around sides of cake to loosen turn out cake onto racks and cook peel off waxed paper can be prepared 1 day ahead wrap cake tightly and store at room temperature for glaze: combine cream butter and corn syrup in heavy medium saucepan bring to simmer over medium heat stirring until butter melt remove from heat add chocolate and stir until melt and smooth stir in rum and vanilla pour glaze into small bowl cover and refrigerate just until cool and thick stirring occasionally about 40 minutes transfer 1 cake layer to platter slide waxed paper strips under edges of cake stir glaze until smooth spread 1 c glaze evenly over top of cake layer top with second cake layer spread remaining glaze over top and sides of cake cover top and sides of cake with chopped butterfinger bar remove paper strips can be made 2 days ahead cover cake and store at room temperature.

Go back

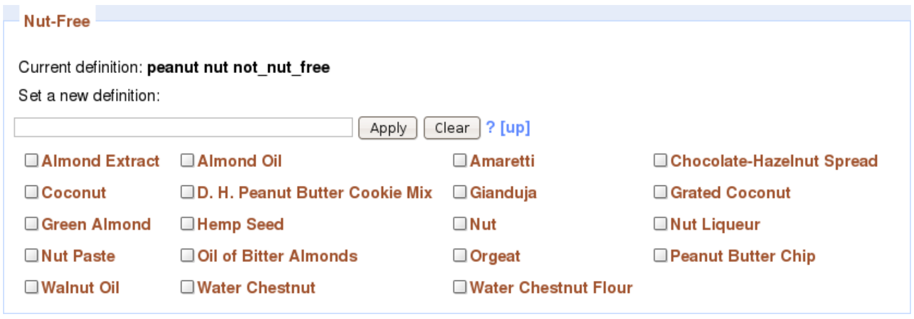
Banana Butterfinger Cake

Type(s): D:baked good,D:dessert,D:cake,D:sweet

Substitution List

- Replace semi-sweet_chocolate by chocolate
- ♦ Replace **banana** by **apple**

Figure B.4: Display of an adapted recipe.



Nut-Free

Current definition: **peanut nut not_nut_free**

Set a new definition:

☐ Almond Extract	☐ Almond Oil	☐ Amaretti	☐ Chocolate-Hazelnut Spread
☐ Coconut	☐ D. H. Peanut Butter Cookie Mix	☐ Gianduja	☐ Grated Coconut
☐ Green Almond	☐ Hemp Seed	☐ Nut	☐ Nut Liqueur
☐ Nut Paste	☐ Oil of Bitter Almonds	☐ Orgeat	☐ Peanut Butter Chip
☐ Walnut Oil	☐ Water Chestnut	☐ Water Chestnut Flour	

Figure B.5: An example of the customization interface.

C
INDEX

INDEX

D NOTATIONS

D.1 Acronyms

FIKA	Failure-driven Interactive Knowledge Acquisition
IAKA	InterActive Knowledge Acquisition
IAKA-NF	IAKA-Numerical Functions
IAKA-OF	IAKA-Other Formalism
FRAKAS	FailuRe Analysis for domain Knowledge AcquiSition
FRAKAS-PL	FRAKAS-Propositional Logic
FRAKAS-DL	FRAKAS-Description Logics
TAAABLE	Taaable

D.2 Abbreviations

CBR	Case-based reasoning
TBR	Trace-based reasoning
KA	Knowledg acquisition
CCC	Computer Cooking Contest
Inc	Inconsistent knowledge
SDK	System domain knowledge

D.3 Case vocabulary

$\mathtt{pb}$	Problem
$\mathtt{sol}$	Solution
$\mathtt{srce\text{-}case}$	Source case
$\mathtt{srce}$	Source problem
$\mathtt{Sol}(\mathtt{srce})$	Source solution
$\mathtt{tgt}$	Target problem
$\mathtt{Sol}(\mathtt{tgt})$	Target solution
$\widetilde{\mathtt{Sol}}(\mathtt{tgt})$	Candidate solution for $\mathtt{tgt}$
$\mathcal{H}$	Dependence set
$\mathcal{L}_{\mathtt{pb}}$	Problem space
$\mathcal{L}_{\mathtt{sol}}$	Solution space
$\mathtt{CB}$	Case base

D.4 IAKA

$\mathtt{AO}$	Adaptation operator
$\mathtt{AM}$	Adaptation method
$\mathtt{r}$	Relation (between problems)
$\mathcal{A}_{\mathtt{r}}$	Adaptation function
$\mathtt{SP}$	Similarity path
$\mathtt{AP}$	Adaptation path
$\ell(\mathtt{SP})$	Length of a similarity path
$\mathtt{dist}(\mathtt{srce}, \mathtt{tgt})$	Distance between problems
$\mathtt{e}_r$	Adaptation error
$\widetilde{\mathtt{e}}_r$	Adaptation error estimation

D.5 FRAKAS

$\circ$	Revision operator
$\circ_D$	Dalal's revision operator
$\mathcal{V}$	Variable set
$\mathcal{I}$	Interpretation
$\mathtt{Mod}$	Modele

D.6 TAAABLE

$\mathcal{O}$	Ontology
Q	Query
R	Recipe
$Idx(R)$	Index
σ	Substitution
$\sqsubseteq$	Subclass
$\dot{-}$	Contraction operator
$\dot{+}$	Revision operator

D.7 Systems

ACCELERE	CABAMAKA	CARMA	CASEY
CHEF	CREEK	CYRUS	DÉJÀ VU
DIAL	KASIMIR	KRITIK	MEDIATOR
MIKAS	MUSETTE	PADIM	PATDEX
PERSUADER	PIXED	PRODIGY	PRODIGY/ANALOGY
PROLABO	PROTOS	RADIX	RESYN/CBR
SWALE	WEB ADAPT		

E
BIBLIOGRAPHY

Bibliography

[Aamodt, 1989] Aamodt, A. (1989). Towards Expert Systems that Learn from Experience. In *Proceedings of the DARPA Case-Based Reasoning Workshop*, pages 181–187, Pensacola Beach, Florida. Morgan Kaufmann. 38

[Aamodt, 1990] Aamodt, A. (1990). Knowledge-Intensive Case-Based Reasoning and Sustained Learning. In Aiello, L., editor, *Proceedings of the 9th European Conference on Artificial Intelligence (ECAI'90)*, pages 1–6. Pitman Publishing, London. 32

[Aamodt, 1991] Aamodt, A. (1991). *A Knowledge-Intensive, Integrated Approach to Problem Solving and Sustained Learning*. PhD thesis, Trondheim University. 19, 20, 109

[Aamodt, 1994] Aamodt, A. (1994). Explanation-driven Case-Based Reasoning. In Wess, S., Althoff, K. D., and Richter, M. M., editors, *Topics in Case-Based Reasoning*, volume LNAI837, pages 274–288, Berlin. Springer. 13

[Aamodt, 2001] Aamodt, A. (2001). Modeling the knowledge contents of CBR systems. In *Proceedings of the Workshop Program (at ICCBR'01)*, pages 32–37, Vancouver. 24, 25

[Aamodt, 2004] Aamodt, A. (2004). Knowledge-Intensive Case-Based Reasoning in Creek. In *Proceedings of the 7th European Conference on Case-Based Reasoning (ECCBR'04)*, pages 1–15, Madrid, Spain. Springer Berlin / Heidelberg. 32

[Aamodt and Plaza, 1994] Aamodt, A. and Plaza, E. (1994). Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches. *AI Communications*, 7(1):39–59. 7, 12, 13, 15, 25

[Abbott, 1884] Abbott, E. A. (1884). *Flatland, a romance of many dimensions*. E.Books, Second revised edition edition. 51

[Alchourrón et al., 1985] Alchourrón, C. E., Gärdenfors, P., and Makinson, D. (1985). On the Logic of Theory Change: partial meet functions for contraction and revision. *Journal of Symbolic Logic*, 50:510–530. 81, 92

[Arcos and De Mantaras, 1997] Arcos, J. L. and De Mantaras, R. L. (1997). Perspectives: A Declarative Bias Mechanism for Case Retrieval. In Leake, D. and Plaza, E., editors, *Proceedings of the 2nd International Conference on Case-Based Reasoning (ICCBR'97)*, pages 279–290. Springer Berlin / Heidelberg. 17

[Armengol and Plaza, 1994a] Armengol, E. and Plaza, E. (1994). A Knowledge Level Model of Case-Based Reasoning. In *Proceedings of the European Workshop on Case-Based Reasoning (EWCBR'94)*, pages 53–64, London, UK. Springer-Verlag. 23

[Armengol and Plaza, 1994b] Armengol, E. and Plaza, E. (1994). Case-Based Reasoning at the Knowledge Level: an Analysis of CHEF. In *Proceedings of the European Workshop on Case-Based Reasoning (EWCBR'94)*, pages 53–64, London, UK. Springer-Verlag. 23

[Avesani and Blanzieri, 1999] Avesani, P. and Blanzieri, E. (1999). Adaptation-Dependent Retrieval Problem: A Formal Definition. In Schmitt, S. and Vollrath, I., editors, *Proceedings of the 3rd International Conference on Case-Based Reasoning (ICCBR'99)*, Munich, Germany. LSA, University of Kaiserslautern. 22

[Baader et al., 2003] Baader, F., Calvanese, D., McGuinness, D., Nardi, D., and Patel-Schneider, P., editors (2003). *The Description Logic Handbook*. Cambridge University Press, cambridge, UK. 108

[Baader et al., 1994] Baader, F., Hollunder, B., Nebel, B., Profitlich, H.-J., and Franconi, E. (1994). An Empirical Analysis of Optimization Techniques for Terminological Representation Systems. *Journal of Applied Intelligence*, 4(2):109–132. 119

[Bachimont, 2004] Bachimont, B. (2004). Pourquoi n'y a-t-il pas d'expérience en ingénierie des connaissances ? In Matta, N., editor, *Actes de Ingénierie des connaissances (IC'04)*. Presses Universitaires de Grenoble. 41

[Badra et al., 2008] Badra, F., Bendaoud, R., Bentebitel, R., Champin, P.-A., Cojan, J., Cordier, A., Després, S., Jean-Daubias, S., Lieber, J., Meilender, T., Mille, A., Nauer, E., Napoli, A., and Toussaint, Y. (2008). Taaable: Text Mining, Ontology Engineering, and Hierarchical Classification for Textual Case-Based Cooking. In Schaaf, M., editor, *Computer Cooking Contest - Workshop at 9th European Conference on Case-Based Reasoning (ECCBR'08)*, pages 219–228, Trier, Germany. 113, 120

[Badra and Lieber, 2008] Badra, F. and Lieber, J. (2008). Representing Case Variations for Learning General and Specific Adaptation Rules. In Cesta, A. and Fakotakis, N., editors, *Proceedings of the 4th Starting AI Researchers' Symposium (STAIRS'08)*, pages 1–11. 134

[Ben Saad, 2008] Ben Saad, M. (2008). Découverte de connaissances dans les traces d'interaction : une approche par similarité des séquences temporelles. Master's thesis, Université Claude Bernard Lyon 1. 137

[Bergmann et al., 1993] Bergmann, R., Pews, G., and Wilke, W. (1993). Explanation-Based Similarity: A Unifying Approach for Integrating Domain Knowledge into Case-Based Reasoning for Diagnosis and Planning Task. In *Proceedings of the European Workshop on Case-Based Reasoning (EWCBR'93)*, pages 182–196. Springer-Verlag - London, UK. 13

[Bergmann and Wilke, 1998] Bergmann, R. and Wilke, W. (1998). Towards a New Formal Model of Transformational Adaptation in Case-Based Reasoning. In Prade, H., editor, *Proceedings of the 13th European Conference on Artificial Intelligence (ECAI'98)*, pages 53–57. John Wiley and Sons. 24, 25, 47, 77

[Bisson, 1994] Bisson, G. (1994). Définition de la similarité dans les modèles à objets. In Rechenmann, F., editor, *Proceedings of Langages et Modèles à Objets (LMO'94)*, pages 70–86, Grenoble. INRIA Rhônes-Alpes. 16

[Branting et al., 2001] Branting, K., Hastings, J., and Lockwood, J. (2001). CARMA: A Case-Based Range Management Advisor. In *Proceedings of the 13th Innovative Applications of Artificial Intelligence Conference (IAAI'01)*, pages 3–10, Seattle, Washington. AAAI Press. 16, 32

[Briggs and Smyth, 2008] Briggs, P. and Smyth, B. (2008). Provenance, Trust, and Sharing in Peer-to-Peer Case-Based Web. In Althoff, K.-D., Bergmann, R., Minor, M., and Hanft, A., editors, *Proceedings of the 9th European Conference on Case-Based Reasoning (ECCBR'08)*, pages 89–103. Springer - LNAI 5239. 139

[Carbonell, 1984] Carbonell, J. G. (1984). Learning by analogy: Formulating and generalizing plans from past experiences. In Michalski, R. S., Carbonell, J. G., and Mitchell, T. M., editors, *Machine Learning: An Artificial Intelligence Approach*, pages 137–162. Springer - Berlin, Heidelberg. 17

[Carbonell, 1986] Carbonell, J. G. (1986). Derivational analogy: A theory of reconstructive problem solving and expertise acquisition. In Michalski, R. S., Carbonell, J. G., and Mitchell, T. M., editors, *Machine Learning: An Artificial Intelligence Approach*, pages 371–392. Morgan Kaufman Publishers: San Mateo, CA. 18

[Carbonell and Veloso, 1988] Carbonell, J. G. and Veloso, M. (1988). Integrating Derivational Analogy into a General Problem Solving Architecture. In *Proceedings of the 1st Workshop on Case-Based Reasoning*, pages 104–124. Tampa, FL: Morgan Kaufmann. 18

[Champin et al., 2008] Champin, P.-A., Cordier, A., Després, S., Fuchs, B., Lieber, J., and Mille, A. (2008). Construction manuelle de la partie haute d'une ontologie modulaire destinée à une annotation de cas textuels - étude de cas pour une application culinaire dans le cadre du projet Taaable. In *Actes du 16ème atelier de Raisonnement à Partir de Cas*, pages 36–50, Nancy. 113, 120

[Champin et al., 2003] Champin, P.-A., Mille, A., and Prié, Y. (2003). MUSETTE: Modelling USEs and Tasks for Tracing Experience. In *Proceedings of the workshop "From structured cases to unstructured problem solving episodes" (at ICCBR'03)*, pages 279–286, NTNU, Norway. 137

[Cheng and Hüllermeier, 2008] Cheng, W. and Hüllermeier, E. (2008). Learning Similarity Functions from Qualitative Feedback. In Althoff, K.-D., Bergmann, R., Minor, M., and Hanft, A., editors, *Proceedings of the 9th European Conference on Case-Based Reasoning (ECCBR'08)*, pages 120–134. Springer - LNAI 5239. 35

[Cho et al., 1999] Cho, K. D., Lee, J. P., Kim, K. T., and Hwang, S. C. (1999). A Learning of Adaptation Knowledge for Case-Based Reasoning System. In *Proceedings of the 17th IASTED International Conference - Applied Informatics*, pages 673–675, Innsbruk, Austria. ACTA Press, Anaheim, Calgary, Zurich. 35

[Cojan and Lieber, 2008] Cojan, J. and Lieber, J. (2008). Conservative Adaptation in Metric Spaces. In Althoff, K.-D., Bergmann, R., Minor, M., and Hanft, A., editors, *Proceedings of the 9th European Conference on Case-Based Reasoning (ECCBR'08)*, pages 135–149. Springer - LNAI 5239. 92

[Compton and Jansen, 1988] Compton, P. and Jansen, R. (1988). Knowledge in Context: a strategy for expert system maintenance. In *Proceedings of the 2nd Australian Joint Artificial Intelligence Conference*, pages 292–306. 35

[Compton and Jansen, 1990] Compton, P. and Jansen, R. (1990). A philosophical basis for knowledge acquisition. *Knowledge Acquisition*, 2:241–257. 35

[Corchado and Lees, 2003] Corchado, J. and Lees, B. (2003). *Soft Computing in Case-Based Reasoning*, chapter 13. Adaptation of cases for case-based forecasting with neural network support, pages 293–319. Springer, Pal, S. K. and Dillon, T. S. and Yeung, D. S. edition. 35

[Cordier, 2004] Cordier, A. (2004). Gestion des Connaissances pour des Systèmes à Base de Connaissances hybrides. Master's thesis, Université Claude Bernard Lyon 1. 7, 27

[Cordier and Fuchs, 2005] Cordier, A. and Fuchs, B. (2005). Un assistant pour la conception et le développement des systèmes de RàPC. In Després, S., editor, *Actes du 13ème atelier de raisonnement à partir de cas (RàPC'05)*, Nice. 27

[Cordier et al., 2008a] Cordier, A., Fuchs, B., Lana de Carvalho, L., Lieber, J., and Mille, A. (2008). Opportunistic Acquisition of Adaptation Knowledge and Cases - The IakA Approach. In Althoff, K.-D., Bergmann, R., Minor, M., and Hanft, A., editors, *Proceedings of the 9th European Conference on Case-Based Reasoning (ECCBR'08)*, pages 150–164. Springer - LNAI 5239. 47

[Cordier et al., 2007a] 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. 81

[Cordier et al., 2007b] Cordier, A., Fuchs, B., Lieber, J., and Mille, A. (2007). Acquisition interactive des connaissances d'adaptation intégrée aux sessions de raisonnement à partir de cas – Principes, architecture IakA et prototype KayaK. In Cordier, A. and Fuchs, B., editors, *Actes du 15ème atelier de raisonnement à partir de cas (RàPC'07)*, pages 71–84, Grenoble. Plateforme AFIA. 47

[Cordier et al., 2007c] Cordier, A., Fuchs, B., Lieber, J., and Mille, A. (2007). Failure Analysis for Domain Knowledge Acquisition in a Knowledge-Intensive CBR System. In Weber, R. and Richter, M., editors, *Proceedings of the 7th International Conference on Case-Based Reasoning (ICCBR'07)*, pages 463–477, Belfast, UK. Springer-Verlag Berlin Heidelberg, LNAI 4626. 81

[Cordier et al., 2007d] Cordier, A., Fuchs, B., Lieber, J., and Mille, A. (2007). Interactive Knowledge Acquisition in Case-Based Reasoning. In *Proceedings of the workshop on Knowledge Discovery and Similarity (at ICCBR'07)*, pages 85–94. Workshop of the seventh International Conference on Case-Based Reasoning Workshop. 27, 47

[Cordier et al., 2007e] Cordier, A., Fuchs, B., Lieber, J., and Mille, A. (2007). On-Line Domain Knowledge Management for Case-Based Medical Recommendation. In Wilson, D. and Khemani, D., editors, *Proceedings of the 5th workshop on CBR in the Health Sciences (at ICCBR'07)*, pages 285–294. . 81

[Cordier et al., 2006a] Cordier, A., Fuchs, B., and Mille, A. (2006). Apprendre à mieux adapter en raisonnement à partir de cas. In Morello, B., editor, *Actes du 14ème atelier de raisonnement à partir de cas (RàPC'06)*, pages 27–37, Besancon. 7

[Cordier et al., 2006b] Cordier, A., Fuchs, B., and Mille, A. (2006). De l'Ingéniérie à l'Apprentissage des connaissances d'adaptation en Raisonnement à Partir de Cas. In Harzallah, M., Charlet, J., and Aussenac-Gilles, N., editors, *Actes des 17èmes journées francophones d'Ingénierie des connaissances (IC'06)*, pages 41–50. 7

[Cordier et al., 2006c] Cordier, A., Fuchs, B., and Mille, A. (2006). Engineering and Learning of Adaptation Knowledge in Case-Based Reasoning. In Staab, S. and Svatek, V., editors, *Proceedings of the 15th International Conference on Knowledge Engineering and Knowledge Management (EKAW'06)*, pages 303–217. Springer-Verlag Berlin, LNAI 4248. 15, 27

[Cordier et al., 2008b] Cordier, A., Fuchs, B., and Mille, A. (2008). *Raisonner à partir de l'expérience*, chapter Le raisonnement à partir de cas : un paradigme de réutilisation de l'expérience (To appear). Traités IC2. Hermes. 7, 27

[Corvaisier et al., 1997] Corvaisier, F., Mille, A., and Pinon, J.-M. (1997). Information Retrieval on the World Wide Web using a decision making system. In *Proceedings of RIAO'97*, pages 284–295, Montréal. 141

[Corvaisier et al., 1998] Corvaisier, F., Mille, A., and Pinon, J.-M. (1998). RADIX 2, Assistance à la recherche d'information documentaire sur le Web. In *Actes des journées d'Ingénierie des Connaissances (IC'98)*, pages 153–163, Pont-à-Mousson. 141

[Cox and Ram, 1999] Cox, M. T. and Ram, A. (1999). Introspective multistrategy learning: On the construction of learning strategies. *Artificial Intelligence*, 112:1–55. 20, 35, 109

[Cram et al., 2007a] Cram, D., Fuchs, B., Mille, A., and Prié, Y. (2007). Raisonner à partir de l'expérience tracée : application à un environnement collaboratif. Research report. 137

[Cram et al., 2008] Cram, D., Fuchs, B., Prié, Y., and Mille, A. (2008). An approach to User-Centric Context-Aware Assistance based on Interaction Traces. In *Proceedings of Modeling and Reasoning in Context (MRC'08)*, Delft. 137

[Cram et al., 2007b] Cram, D., Jouvin, D., and Mille, A. (2007). Visualizing Interaction Traces to improve Reflexivity in Synchronous Collaborative e-Learning Activities. In Limited, A. C., editor, *Proceedings of the 6th European Conference on e-Learning*, pages 147–158, Copenhaguen. 137

[Craw et al., 2006] Craw, S., Wiratunga, N., and Rowe, R. C. (2006). Learning Adaptation Knowledge to Improve Case-Based Reasoning. *Artificial Intelligence*, 170(16-17):1175–1192. 34

[Dalal, 1988] Dalal, M. (1988). Investigations into a theory of Knowledge Base Revision: Preliminary Report. In *Proceedings of the 7th National Conference on Artificial Intelligence*, pages 475–479, Saint Paul, Minnesota, USA. 96

[d'Aquin et al., 2007] d'Aquin, M., Badra, F., Lafrogne, S., Lieber, J., Napoli, A., and Szathmary, L. (2007). Case Base Mining for Adaptation Knowledge Acquisition. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI'07)*, pages 750–755. 34, 129, 134

[d'Aquin et al., 2004] d'Aquin, M., Brachais, S., Lieber, J., and Napoli, A. (2004). Vers une acquisition automatique de connaissances d'adaptation par examen de la base de cas – une approche fondée sur des techniques d'extraction de connaissances dans des bases de données. In *Actes du 12ème Atelier de Raisonnement à Partir de Cas (RàPC'04)*, pages 41–52, Université Paris Nord, Villetaneuse, France. 34

[d'Aquin et al., 2006] d'Aquin, M., Lieber, J., and Napoli, A. (2006). Adaptation Knowledge Acquisition: a Case Study for Case-Based Decision Support in Oncology. *Computational Intelligence*, 22(3 4):161–176. 34, 90, 97, 108

[Dubois et al., 1997] Dubois, D., Esteva, F., Garcia, P., Godo, L., Màntaras, R. L. d., and Prade, H. (1997). Fuzzy Modelling of Case-Based Reasoning and Decision. In Leake, D. and Plaza, E., editors, *Proceedings of the 2nd International Conference on Case-Based Reasoning (ICCBR'97)*, pages 599–610. Springer Berlin / Heidelberg. 59, 92

[Fox, 1995] Fox, S. (1995). *Introspective Learning for Case-Based Planning*. PhD thesis, Indiana University. 20

[Fox and Leake, 1994a] Fox, S. and Leake, D. (1994). Introspective Reasoning in a Case-based Planner. In *Proceedings of the 12th National Conference on Artificial Intelligence*, page 1446 , Seattle, Washington, United States. 19

[Fox and Leake, 1994b] Fox, S. and Leake, D. (1994). Using Introspective Reasoning to Guide Index Refinement in Case-Based Reasoning. In *Proceedings of the 16th Annual Conference of the Cognitive Science Society*, pages 324–329, Atlanta, GA. 19

[Fuchs, 1997] Fuchs, B. (1997). *Representation des connaissances pour le raisonnement à partir de cas. Le système ROCADE*. PhD thesis, Université Jean Monnet de Saint-Etienne. 141

[Fuchs et al., 1999] Fuchs, B., Lieber, J., Mille, A., and Napoli, A. (1999). Towards a Unified Theory of Adaptation in Case-Based Reasoning. In Schmitt, S. and Vollrath, I., editors, *Proceedings of the 3rd International Conference on Case-Based Reasoning (ICCBR'99)*, pages 104–117, Munich, Germany. LSA, University of Kaiserslautern. 24

[Fuchs et al., 2000] Fuchs, B., Lieber, J., Mille, A., and Napoli, A. (2000). An Algorithm for Adaptation in Case-Based Reasoning. In Horn, W., editor, *Proceedings of the 14th European Conference on Artificial Intelligence (ECAI'2000)*, pages 45–49, Berlin, Germany. IOS Press. 18, 24, 25, 62

[Fuchs et al., 2006a] Fuchs, B., Lieber, J., Mille, A., and Napoli, A. (2006). A general strategy for adaptation in Case-Based Reasoning. Research report - RR-LIRIS-2006-016. 24, 62

[Fuchs et al., 2006b] Fuchs, B., Lieber, J., Mille, A., and Napoli, A. (2006). Une première formalisation de la phase d'élaboration du raisonnement à partir de cas. In *Actes du 14ème atelier de raisonnement à partir de cas (RàPC'06)*, Besancon. 16

[Fuchs and Mille, 1999] Fuchs, B. and Mille, A. (1999). Une modélisation au niveau connaissance de la tâche d'adaptation en raisonnement à partir de cas. In Trousse, B. and Mille, A., editors, *Actes du 7ème atelier de raisonnement à partir de cas (RàPC'99)*, pages 27–36, Palaiseau, France. 23

[Fuchs and Mille, 2000] Fuchs, B. and Mille, A. (2000). Une modélisation au niveau connaissance du raisonnement à partir de cas. In Aussenac-Gilles, N., editor, *Actes des journées francophones d'acquisition des connaissances (IC'00)*, pages 3–10, Toulouse. 23

[Fuchs et al., 1995] Fuchs, B., Mille, A., and Chiron, B. (1995). Operator decision aiding by adaptation of supervision strategies . In *Proceedings of the 1st International Conference on Case-Based Reasoning (ICCBR'95)*, pages 23–32, Sesimba, Portugal. Springer. 141

[Gärdenfors, 1992] Gärdenfors, P. (1992). *Belif Revision*. Cambridge Univerity Press. 126, 127

[Gentner and Forbus, 1991] Gentner, D. and Forbus, K. (1991). MAC/FAC: A model of similarity-based retrieval. In Erlbaum, L., editor, *Proceedings of the 13th Annual Conference of the Cognitive Science Society*, pages 504–509, Hillsdale, NJ. 16

[Gentner et al., 2001] Gentner, D., Holyoak, K., and Kokinov, B., editors (2001). *The Analogical Mind: Perspectives from Cognitive Science*, Cambridge, MA. MIT Press. 11

[Georgeon, 2008] Georgeon, O. (2008). *Analyse de traces d'activité pour la modélisation cognitive : application à la conduite automobile*. PhD thesis, Université Lumière Lyon 2. 137

[Georgeon et al., 2007] Georgeon, O., Henning, M., Bellet, T., and Mille, A. (2007). Creating Cognitive Models from Activity Analysis: A Knowledge Engineering Approach to Car Driver Modeling. In Taylor and Francis, editors, *Proceedings of the International Conference on Cognitive Modeling*, pages 43–48, Ann Arbor, Michigan. 137

[Gick and Holyoak, 1980] Gick, M. and Holyoak, K. (1980). Analogical Problem Solving. *Cognitive Psychology*, 12:306–355. 11

[Goel and Chandrasekaran, 1989] Goel, A. and Chandrasekaran, B. (1989). Use of device models in adaptation of design cases. In *Proceedings of the DARPA Workshop on Case-Based Reasoning*, pages 100–109, Pensacola Beach, Florida. Morgan Kaufmann. 18

[Gómez-Albarrán et al., 1999] Gómez-Albarrán, M., González-Calero, P. A., Díaz-Agudo, B., and Fernández-Conde, C. (1999). Modelling the CBR Life Cycle Using Description Logics. In Schmitt, S. and Vollrath, I., editors, *Proceedings of the 3rd International*

Conference on Case-Based Reasoning (ICCBR'99), pages 147–161, Munich, Germany. LSA, University of Kaiserslautern. 23

[Gonzalez-Calero et al., 1999] Gonzalez-Calero, P. A., Gomez-Albarran, M., and Belén, D.-A. (1999). A substitution-based Adaptation Model. In Schmitt, S. and Vollrath, I., editors, *Challenges for Case-Based Reasoning - Proceedings of the ICCBR'99 Workshops*, pages 17–26, Seeon Monastery, Germany. University of Kaiserslautern, Computer Science. 23

[Hammond, 1986] Hammond, K. (1986). CHEF: A model of case-based planning. In Press, A., editor, *Proceedings of the 5th National Conference on Artificial Intelligence*, pages 267–271, Menlo Park, CA. 10, 16, 18, 25, 32, 40

[Hammond, 1989] Hammond, K. (1989). *Inside Case-Based Reasoning*, chapter 6. Chef, pages 165–212. Lawrence Erlbaum, Riesbeck, C. and Schanl, R. C. edition. 27, 32

[Hammond, 1990] Hammond, K. (1990). Explaining and Repairing Plans that Fail. *Artificial Intelligence*, 45(1-2):173–228. 32, 109

[Hanney and Keane, 1996] Hanney, K. and Keane, M. T. (1996). Learning Adaptation Rules from a Case-Base. In *Proceedings of the 3rd European Workshop on Case-Based Reasoning (EWCBR'96)*, pages 179–192, London, UK. Springer-Verlag. 33

[Hanney and Keane, 1997] Hanney, K. and Keane, M. T. (1997). The adaptation knowledge bottleneck: How to ease it by learning from cases. In Leake, D. and Plaza, E., editors, *Proceedings of the 2nd International Conference on Case-Based Reasoning (ICCBR'97)*, pages 359–370. Springer Berlin / Heidelberg. 33

[Hanney et al., 1995] Hanney, K., Keane, M. T., Smyth, B., and Cunningham, P. (1995). What Kind of Adaptation do CBR Systems Need?: A Review of Current Practice. In Aha, D. and Ram, A., editors, *Adaptation of Knowledge for Reuse: Proceedings of the 1995 AAAI Fall Symposium*. AAAI Press. 18

[Hastings et al., 1995] Hastings, J. D., Karl, B. L., and Lockwood, J. A. (1995). Case Adaptation Using an Incomplete Causal Model. In *Proceedings of the 1st International Conference on Case-Based Reasoning (ICCBR'95)*, pages 181–192, Sesimbra, Portugal. Springer. 32

[Heraud and Mille, 2000] Heraud, J.-M. and Mille, A. (2000). Pixed : vers le partage et la réutilisation d'expériences pour assister l'apprentissage. In *Actes du congrès Technologies de l'information et de la communication dans les enseignements d'ingénieurs et dans l'industrie (TICE'00)*, pages 237–244. 141

[Herbeaux, 2000] Herbeaux, O. (2000). *ACCELERE : aide à la conception de caoutchouc cellulaire exploitant la remémoration d'expériences*. PhD thesis, Université Jean Monnet, Saint Étienne. 16, 141

[Herbeaux and Mille, 2007] Herbeaux, O. and Mille, A. (2007). *ACCELERE : système d'aide à la conception de caoutchouc cellulaire exploitant la remémoration d'expérience*, volume 1 of *Traité IC2 - Informatique et systèmes d'information*, chapter Raisonnement à partir de cas 1 : conception et configuration de produits. Hermes Science. 141

[Iglezakis et al., 2004] Iglezakis, I., Reinartz, T., and Roth-Berghofer, T. (2004). Maintenance Memories: Beyond Concepts and Techniques for Case Base Maintenance. In *Proceedings of the 7th European Conference on Case-Based Reasoning (ECCBR'04)*, pages 227–241, Madrid, Spain. Springer Berlin / Heidelberg. 14

[Jarmulak et al., 2001] Jarmulak, J., Craw, S., and Rowe, R. (2001). Using Case-Based Data to Learn Adaptation Knowledge for Design. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI'01)* , pages 1011–1016. Morgan Kaufmann. 34

[Jo et al., 2000] Jo, T. C., Seo, J. H., and Hyeon, K. (2000). Topic Spotting on News Articles with Topic Repository by Controlled Indexing. In *Intelligent Data Engineering and Automated Learning (IDEAL'00). Data Mining, Financial Engineering, and Intelligent Agents*. Springer Berlin / Heidelberg. 122

[Kass et al., 1986] Kass, A., Leake, D., and Owens, C. (1986). *Explanation patterns: Understanding mechanically and creatively.*, chapter SWALE: A program that explains, pages 232–254. Lawrence Erlbaum, Hillsdale, NJ, Schank, R. edition. 10

[Katsuno and Mendelzon, 1991] Katsuno, H. and Mendelzon, A. O. (1991). Propositional knowledge base revision and minimal change. *Artificial Intelligence*, 52(3):263–294. 94, 96

[Kendall and Stuart, 1969] Kendall, M. G. and Stuart, A. (1969). *The advanced theory of statistics: distribution theory*, volume 1. New York: Hafner. 72

[Khan, 2003] Khan, A. S. (2003). *Incremental Knowledge Acquisition for Case-Based Reasoning*. PhD thesis, The University of New South Wales. 35

[Kinley, 2001] Kinley, A. (2001). *Learning to improve case adaptation*. PhD thesis, Indiana University. 19, 25, 33, 47

[Kolodner, 1988] Kolodner, J., editor (1988). *Workshop on case-based Reasoning, DARPA 88*, Clearwater, Florida. Morgan Kaufmann, San Mateo. 10

[Kolodner, 1993] Kolodner, J. (1993). *Case-Based Reasoning*. Morgan Kaufmann, San Mateo, CA. 10, 18, 42, 78

[Konieczny, 1999] Konieczny, S. (1999). *Sur la logique du changement : Révision et Fusion de bases de connaissances*. PhD thesis, Université des Sciences et Technologies de Lille. 92

[Konieczny et al., 2004] Konieczny, S., Lang, J., and Marquis, P. (2004). DA^2 merging operators. *Artificial Intelligence*, 157(1-2):49–79. 108

[Koton, 1988] Koton, P. (1988). Reasoning about evidence in causal explanations. In Press, A., editor, *Proceedings of the Seventh National conference on Artificial Intelligence*, pages 256–261, Menlo Park, CA. 10, 18

[Laflaquière et al., 2006] Laflaquière, J., Settouti, L. S., Prié, Y., and Mille, A. (2006). A trace-based system framework for experience management and engineering. In *Proceedings of the 2nd International Workshop on Experience Management and Engineering (EME'06, held in conjunction with KES'06)*, Bournemouth, United Kingdom. 137

[Leake, 1996] Leake, D., editor (1996). *Case-Based Reasoning: Experiences, Lessons, and Future Directions*. AAAI Press/MIT Press. 27

[Leake and Dial, 2008] Leake, D. and Dial, S. (2008). Using Case Provenance to Propagate Feedback to Cases and Adaptations. In Althoff, K.-D., Bergmann, R., Minor, M., and Hanft, A., editors, *Proceedings of the 9th European Conference on Case-Based Reasoning (ECCBR'08)*, pages 255–268. Springer - LNAI 5239. 139

[Leake et al., 1999] Leake, D., Hammond, K., Birnbaum, L., Marlow, C., and H., Y. (1999). Task-based knowledge management. In *Exploring Synergies of Knowledge Management and Case-Based Reasoning, Proceedings of The American Association of Artificial Intelligence (AAAI'99) Workshop*, pages 35–39, Orlando, Florida. AAAI Press. 24, 25

[Leake and Kendall-Morwick, 2008] Leake, D. and Kendall-Morwick, J. (2008). Towards Case-Based Support for e-Science Workflow Generation by Mining Provenance. In Althoff, K.-D., Bergmann, R., Minor, M., and Hanft, A., editors, *Proceedings of the 9th European Conference on Case-Based Reasoning (ECCBR'08)* , pages 269–283. Springer - LNAI 5239. 139

[Leake et al., 1996a] Leake, D., Kinley, A., and Wilson, D. (1996). Acquiring Case Adaptation Knowledge: A Hybrid Approach. In *Proceedings of the 13th National Conference on Artificial Intelligence*, pages 684–689, Portland, Oregon. AAAI Press, Menlo Park, CA. 33

[Leake et al., 1996b] Leake, D., Kinley, A., and Wilson, D. (1996). Multistrategy Learning to Apply Cases for Case-Based Reasoning. In *Proceedings of the third International Workshop on Multistrategy Learning*, pages 155–164, Menlo Park, CA. AAAI Press. 33

[Leake et al., 1997] Leake, D., Kinley, A., and Wilson, D. (1997). Case-Based Similarity Assessment: Estimating Adaptability from Experience. In *Fourteenth National Conference on Artificial Intelligence*, pages 674–679, Menlo Park, CA. AAAI Press. 16, 22

[Leake and Powell, 2007] Leake, D. and Powell, J. (2007). Mining Large-Scale Knowledge Sources for Case Adaptation Knowledge. In Weber, R. and Richter, M., editors, *Proceedings of the 7th International Conference on Case-Based Reasoning (ICCBR'07)*, pages 209–223, Belfast, UK. 35, 139

[Leake and Powell, 2008] Leake, D. and Powell, J. (2008). Knowledge Planning and Learned Personalization for Web-Based Case Adaptation. In Althoff, K.-D., Bergmann, R., Minor, M., and Hanft, A., editors, *Proceedings of the 9th European Conference on Case-Based Reasoning (ECCBR'08)*, pages 284–298. Springer - LNAI 5239. 139

[Leake and Whitehead, 2007] Leake, D. and Whitehead, M. (2007). Case Provenance: The Value of Remembering Case Sources. In Weber, R. and Richter, M., editors, *Proceedings of the 7th International Conference on Case-Based Reasoning (ICCBR'07)*, pages 194–208, Belfast, UK. Springer-Verlag Berlin Heidelberg, LNAI 4626. 139

[Leake and Wilson, 2000] Leake, D. and Wilson, D. (2000). Remembering Why to Remember: Performance-Guided Case-Base Maintenance. In *Proceedings of the 5th European Workshop on Case-Based Reasoning (EWCBR'00)*, pages 161–172, Berlin. Springer. 20

[Lee, 2003] Lee, M. (2003). A study of an automatic learning model of adaptation knowledge for case-based reasoning. *Information Science*, 155(1-2):61–78. 35

[Lenz and Burkhard, 1996] Lenz, M. and Burkhard, H.-D. (1996). Case Retrieval Nets: Basic Ideas and Extensions. In Görz, G. and Hölldobler, S., editors, *Proceedings of the 20th Annual German Conference on Artificial Intelligence (KI'96)*, pages 227–239, London, UK. Springer-Verlag - LNAI 1137. 17

[Lieber, 1997] Lieber, J. (1997). *Raisonnement à partir de cas et classification hiérarchique. Application à la planification de synthèse en chimie organique*. PhD thesis, Université Henri Poincaré, Nancy 1. 117, 119

[Lieber, 1999] Lieber, J. (1999). Reformulations and Adaptation Decomposition. In Schmitt, S. and Vollrath, I., editors, *Proceedings of the 3rd International Conference on Case-Based Reasoning (ICCBR'99)*, Munich, Germany. LSA, University of Kaiserslautern. 23

[Lieber, 2006] Lieber, J. (2006). A Definition and a Formalization of Conservative Adaptation for Knowledge-Intensive Case-Based Reasoning Application to Decision Support in Oncology (A Preliminary Report). Research Report. 24, 81, 92

[Lieber, 2007a] Lieber, J. (2007). Application de la révision et de la fusion des connaissances à l'adaptation et à la combinaison de cas. In Cordier, A. and Fuchs, B., editors, *Actes du 15ème atelier de raisonnement à partir de cas (RàPC'07)*, pages 119–129, Grenoble. Plateforme AFIA. 92

[Lieber, 2007b] Lieber, J. (2007). Application of the Revision Theory to Adaptation in Case-Based Reasoning: the Conservative Adaptation. In Weber, R. and Richter, M., editors, *Proceedings of the 7th International Conference on Case-Based Reasoning (ICCBR'07)*, pages 239–253, Belfast, UK. Springer-Verlag Berlin Heidelberg, LNAI 4626. 25, 81, 84, 92, 97

[Lieber et al., 2001] Lieber, J., Bey, P., Boisson, F., Bresson, B., Falzon, P., Lesur, A., Napoli, A., Rios, M., and Sauvagnac, C. (2001). Acquisition et modélisation de connaissances d'adaptation, une étude pour le traitement du cancer du sein. In *Actes des Journées d'ingénierie des connaissances (IC'01)*, pages 409–426, Grenoble, France. Presses Universitaires de Grenoble. 34

[Lieber et al., 2005] Lieber, J., d'Aquin, M., Badra, F., and Napoli, A. (2005). Case-Based Treatment Recommendations for Breast Cancer. Research Report. 34, 107, 141

[Lieber et al., 2002] Lieber, J., d'Aquin, M., Bey, P., Bresson, B., Croissant, O., Falzon, P., Lesur, A., Lévêque, J., Mollo, V., Napoli, A., Rios, M., and Sauvagnac, C. (2002). The Kasimir Project: Knowledge Management in Cancerology. In *Proceedings of the 4th International Workshop on Enterprise Networking and Computing in Health Care Industry (HealthComm'02)*, pages 125–127, Nancy. 141

[Lieber et al., 2004] Lieber, J., d'Aquin, M., Brachais, S., and Napoli, A. (2004). Une étude comparative de quelques travaux sur l'acquisition de connaissances d'adaptation pour le raisonnement à partir de cas. In *Actes du 12ème atelier de raisonnement à partir de cas (RàPC'04)*, pages 53–60, Université Paris Nord, Villetaneuse, France. 31

[Lieber and Napoli, 1998] Lieber, J. and Napoli, A. (1998). Correct and Complete Retrieval for Case-Based Problem-Solving. In Prade, H., editor, *Proceedings of the 13th European Conference on Artificial Intelligence (ECAI'98)*, pages 68–72, Brighton, United Kingdom. 16, 77

[López de Mántaras et al., 2005] López de Mántaras, R., McSherry, D., Bridge, D., Leake, D., Smyth, B., Craw, S., Falting, B., Maher, M. L., Cox, M. T., Forbus, K., Keane, M. T., Aamodt, A., and Watson, I. (2005). Retrieval, reuse, revision and retention in case-based reasoning. *Knowledge Engineering Review*, 20(3):215–240. 7, 15, 131

[Malek, 1996] Malek, M. (1996). *Un modèle hybride de mémoire pour le raisonnement à partir de cas*. PhD thesis, Université Joseph Fourier, Grenoble. 17

[Mascret, 2008] Mascret, B. (2008). Apprendre à assister l'utilisateur à partir de l'expérience tracée ? Master's thesis, Université Claude Bernard Lyon 1. 135, 137

[Massie et al., 2004] Massie, S., Craw, S., and Wiratunga, N. (2004). Visualisation of Case-Base Reasoning for Explanation. In *Proceedings of the 7th European Conference on Case-Based Reasoning (ECCBR'04)*, pages 135–144, Madrid, Spain. Springer Berlin / Heidelberg. 13

[McSherry, 2003] McSherry, D. (2003). Similarity and compromise. In *Proceedings of the 5th International Conference on Case-Based Reasoning (ICCBR'03)*, pages 291–305, Berlin. Springer. 17

[Melis et al., 1998] Melis, E., Lieber, J., and Napoli, A. (1998). Reformulation in Case-Based Reasoning. In Smyth, B. and Cunningham, P., editors, *Proceedings of the 4th European Workshop on Case-Based Reasoning (EWCBR'98)*, pages 172–183. Springer. 54

[Mille, 2006a] Mille, A. (2006). From case-based reasoning to traces-based reasoning. *Annual Reviews in Control*, 30(2):223–232. 137

[Mille, 2006b] Mille, A. (2006). *Le storytelling : concepts, outils et applications*, chapter Raisonner à partir de l'expérience tracée. Traité IC2 - Informatique et systèmes d'information. Soulier, E., Hermes Science edition. 137

[Mille et al., 1999] Mille, A., Fuchs, B., and Chiron, B. (1999). Raisonnement fondé sur l'expérience : un nouveau paradigme en supervision industrielle ? *Revue d'intelligence artificielle*, 13:97–128. 141

[Mille et al., 1996] Mille, A., Fuchs, B., and Herbeaux, O. (1996). A unifying framework for Adaptation in Case-Based Reasoning. In Voss, A., Bergmann, R., and Bartsch-Sporl, B., editors, *Workshop on Adaptation in Case-Based Reasoning, 12th European Conference on Artificial Intelligence (ECAI'96)*, pages 22–28, Budapest, Hungary. Springer. 141

[Minton, 1990] Minton, S. (1990). Qualitative results concerning the utility of explanation-based learning. *Artificial Intelligence*, 42:363–391. 20

[Napoli, 1997] Napoli, A. (1997). Une introduction aux logiques de description. Technical Report 3314, INRIA. 108

[Newell, 1982] Newell, A. (1982). The Knowledge Level. *Artificial Intelligence*, 19(2):87–127. 7, 23, 25

[Py, 1994] Py, M. (1994). Un modèle conceptuel de raisonnement par analogie. *Revue d'Intelligence Artificielle*, 8:63–99. 11

[Ram, 1993] Ram, A. (1993). Indexing, Elaboration and Refinement: Incremental Learning of Explanatory Cases. *Machine Learning*, 10:201–248. 35

[Richter, 1992] Richter, M. M. (1992). Classification and Learning of Similarity Measures. In *Studies in Classification, Data Analysis and Knowledge Organisation*, pages 1–8. Springer. 16, 35

[Richter, 1995] Richter, M. M. (1995). The Knowledge Contained in Similarity Measures. Invited Talk of the First International Conference on Case-Based Reasoning (ICCBR'95). 12, 21, 25

[Richter and Aamodt, 2005] Richter, M. M. and Aamodt, A. (2005). Case-based reasoning foundations. *Knowledge Engineering Review*, 20:203–207. 15, 131

[Riesbeck and Schank, 1989] Riesbeck, C. K. and Schank, R. C. (1989). *Inside Case-Based Reasoning*. Lawrence Erlbaum, Hillsdale, NJ. 7, 12, 13, 15, 25, 40, 76, 92

[Rifqi, 1996] Rifqi, M. (1996). *Mesures de Comparaison, Typicalité et Classifications D'objets Flous : Théorie et Pratique*. PhD thesis, Université Pierre et Marie Curie, Paris 6. 16

[Rocchio, 1966] Rocchio, J. J. (1966). *Document Retrieval Systems – Optimization and Evaluation*. PhD thesis, Harvard University. 109

[Salotti and Ventos, 1998] Salotti, S. and Ventos, V. (1998). Study and Formalization of a Case-Based Reasoning System Using a Description Logic. In Smyth, B. and Cunningham, P., editors, *Proceedings of the 4th European Workshop on Case-Based Reasoning (EWCBR'98)*, pages 286–297. Springer. 23

[Sanchez-Ruiz et al., 2008] Sanchez-Ruiz, A., Gomez-Martin, P. P., Diaz-Agudo, B., and Gonzalez-Calero, P. A. (2008). Adaptation through Planning in Knowledge Intensive CBR. In Althoff, K.-D., Bergmann, R., Minor, M., and Hanft, A., editors, *Proceedings of the 9th European Conference on Case-Based Reasoning (ECCBR'08)*, pages 503–517. Springer - LNAI 5239. 24

[Schanck et al., 1994] Schanck, R. C., Kass, A., and Riesbeck, C. K., editors (1994). *Inside Case-Based Explanation*. Lawrence Erlbaum Associates. 10

[Schank, 1982] Schank, R. C. (1982). *Dynamic Memory: A Theory of Reminding and Learning in Computers and People*. Cambridge University Press, New York, NY. 10

[Schank, 1986] Schank, R. C. (1986). *Explanation Patterns: Understanding Mechanically and Creatively*. Lawrence Erlbaum, Hillsdale, NJ. 10

[Schank and Abelson, 1977] Schank, R. C. and Abelson, R. P. (1977). *Scripts, Plans, Goals and Understanding: An Inquiry into Human Knowledge Structures*. Lawrence Erlbaum, Hillsdale, NJ. 10

[Settouti et al., 2007] Settouti, L. S., Prié, Y., Marty, J.-C., and Mille, A. (2007). Vers des systèmes à base de traces modélisées pour les EIAH. Research report - RR-LIRIS-2007-016. 137

[Simpson, 1985] Simpson, R. L. (1985). A Computer Model of Case-Based Reasoning in Problem Solving: An Investigation in the Domain of Dispute Mediation. Technical report GIT-ICS-85/18, Georgia Institue of Technology, Atlanta, GA. 40

[Slade, 1991] Slade, S. (1991). Case-Based Reasoning: A Research Paradigm. *AI Magazine*, 12(1):42–55. 11

[Smyth, 1996a] Smyth, B. (1996). Case Adaptation and Reuse in Déjà Vu. Research report. 18

[Smyth, 1996b] Smyth, B. (1996). *Case-Based Design*. PhD thesis, Trinity College, Dublin, Ireland. 22

[Smyth and Cunningham, 1993] Smyth, B. and Cunningham, P. (1993). Complexity of Adaptation in Real-World Case-Based Reasoning Systems. In *Proceedings of the 6th Irish Conference on Artificial Intelligence and Cognitive Systems*, pages 228–240. 17

[Smyth and Keane, 1995a] Smyth, B. and Keane, M. T. (1995). Experiments on Adaptation-Guided Retrieval in Case-Based Design. In *Proceedings of the 1st International Conference on Case-Based Reasoning (ICCBR'95)*, pages 313–324, Sesimbra, Portugal. Springer. 19

[Smyth and Keane, 1995b] Smyth, B. and Keane, M. T. (1995). Remembering To Forget. A Competence-Preserving Deletion Policy for Case-Based Reasoning Systems. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI'95)*, pages 377–382. 20

[Smyth and Keane, 1995c] Smyth, B. and Keane, M. T. (1995). Retrieval and Adaptation in Déjà Vu, a Case-Based Reasoning System for Software Design. In *Adaptation of Knowledge for Reuse: AAAI Fall Symposium*, pages 228–240. AAAI Press. 18, 25

[Smyth and Keane, 1996] Smyth, B. and Keane, M. T. (1996). Using adaptation knowledge to retrieve and adapt design cases. *Knowledge-Based Systems*, 9(2):127–135. 19

[Smyth and Keane, 1998] Smyth, B. and Keane, M. T. (1998). Adaptation-Guided Retrieval: Questioning the Similarity Assumption in Reasoning. *Artificial Intelligence*, 102(2):249–293. 16, 17, 25, 54, 60

[Smyth and McKenna, 1998] Smyth, B. and McKenna, E. (1998). Modelling the competence of case-bases. In Smyth, B. and Cunningham, P., editors, *Proceedings of the 4th European Workshop on Case-Based Reasoning (EWCBR'98)*, pages 208–220. Springer. 65, 78

[Sormo et al., 2005] Sormo, F., Cassens, J., and Aamodt, A. (2005). Explanation in Case-Based Reasoning - Perspectives and Goals. *Artificial Intelligence Review*, 24:109–143. 13, 109

[Steels, 1992] Steels, L. (1992). Reusability and Configuration of Applications by Non-programmers. In *Proceedings of Artificial Intelligence from th Information Processing Perspective*, Madrid. 23

[Strunk and White, 1979] Strunk, W. and White, E. B. (1979). *The Elements of Style*. Macmillan Publishing Co., Inc. 5

[Stuber, 2007] Stuber, A. (2007). *Co-construction de sens par négociation pour la réutilisation en situation de l'expérience tracée - Vers le partage et l'échange d'expérience collective*. PhD thesis, Université Claude Bernard Lyon 1. 137

[Sycara, 1992] Sycara, K. (1992). The Persuader. *Encyclopedia of Artificial Intelligence*, Shapiro Ed. 40

[Tversky, 1977] Tversky, A. (1977). Features of similarity. *Psychological Review*, 84:327–352. 16

[Uren et al., 2006] Uren, V., Cimiano, P., Iria, J., Handschuh, S., Vargas-Vera, M., Motta, E., and Ciravegna, F. (2006). Semantic Annotation for Knowledge Management: Requirements and a Survey of the State of the Art. *Journal of Web Semantics: Science, Services and Agents on the World Wide Web*, 4(1):14–28. 122

[Veloso et al., 1995] Veloso, M., Carbonell, J., Pérez, A., Borrajo, D., Fink, E., and Blythe, J. (1995). Integrating Planning and Learning: The PRODIGY Architecture. *Journal of Experimental and Theoretical Artificial Intelligence*, 7(1):81–120. 18

[Watson and Marir, 1994] Watson, I. and Marir, F. (1994). Case-Based Reasoning: A Review. *Knowledge Engineering Review*, 9(4):355–381. 15

[Wilcoxon, 1945] Wilcoxon, F. (1945). Individual Comparisons by Ranking Methods. *Biometrics Bulletin*, 1(6):80–83. 72

[Wilke and Bergmann, 1998] Wilke, W. and Bergmann, R. (1998). Techniques and knowledge used for adaptation during case-based problem solving. In *Proceedings of the 11th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems: Tasks and Methods in Applied Artificial Intelligence*, pages 497–506. 18

[Wilke et al., 1996] 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. 35

[Wiratunga et al., 2002] Wiratunga, N., Craw, S., and Rowe, R. (2002). Learning to adapt for case-based design. In *Proceedings of the 6th European Conference on Case-Based Reasoning (ECCBR'02)*, pages 423–437, Berlin. Springer. 34

[Zehraoui, 2004] Zehraoui, F. (2004). *Systèmes d'apprentissage connexionnistes et raisonnement à partir de cas pour la classification et le classement de séquences*. PhD thesis, Université Paris Nord. 35

Résumé

Jeune discipline à la croisée de l'informatique, de l'intelligence artificielle et des sciences cognitives, l'ingénierie des connaissances vise à modéliser les connaissances d'un domaine pour les opérationnaliser dans un outil informatique. Pour cela, elle propose des outils théoriques, des modèles et des méthodologies empiriques pour instrumenter les systèmes et permettre l'échange de connaissances entre l'utilisateur et l'outil informatique.

Le travail développé ici traite de l'ingénierie des connaissances d'une catégorie de systèmes en particulier : les systèmes de raisonnement à partir de cas (RÀPC). Un système de RÀPC assiste un utilisateur dans sa tâche de résolution de problème en lui proposant une adaptation à la situation courante d'une précédente expérience. C'est en particulier au système en interaction "utilisateur - outil de RÀPC" que nous nous intéressons ici.

La problématique étudiée peut donc être exprimée ainsi : quelles méthodes et outils développer pour instrumenter efficacement le système apprenant "utilisateur - outil de RàPC" ? Cette problématique soulève un questionnement sur les connaissances du raisonnement et conduit à une analyse au niveau connaissance de tels systèmes. Un autre volet d'analyse porte sur les interactions entre l'utilisateur et l'artefact informatique pendant les phases de résolution de problème. Ces aspects sont étudiés à plusieurs niveaux dans les différentes contributions présentées dans cette thèse.

Nos différentes expériences et expérimentations nous ont conduits à proposer, comme première contribution, une formalisation à un niveau général de l'apprentissage interactif de connaissances en RÀPC (FIKA). Cette formalisation repose sur les échecs de raisonnement qui, puisqu'ils permettent de mettre en évidence les lacunes dans les connaissances disponibles, sont utilisés pour guider le processus d'apprentissage. Deux extensions de ce modèle général ont été proposées : IAKA et FRAKAS.

IAKA raffine les principes proposés par FIKA pour permettre leur mise en œuvre immédiate dans une certaine catégorie de systèmes où les connaissances peuvent être représentées selon un modèle donné (cas et connaissances d'adaptation représentées par des opérateurs d'adaptation). Ces principes ont été implantés et expérimentés dans une application développée à des seules fins expérimentales.

FRAKAS propose des méthodes et outils similaires pour une autre catégorie de systèmes ou les connaissances du domaines sont utilisées pour guider l'adaptation. Ces principes ont, quant à eux, été implantés dans un prototype inspiré d'une application réelle.

IAKA et FRAKAS, les deux extensions de l'approche FIKA, présentent des forces et des limites, une réflexion quant à leur intégration possible à donc été menée. Une première tentative pratique d'intégration a été mise en œuvre dans une application de RÀPC permettant l'adaptation de recettes de cuisine : le logiciel TAAABLE.

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). A CBR system assists a user in his problem solving task by retrieving a previous successful problem solving experience and by adapting it to the current situation. In this work, we are mainly interested in the interacting system "user - CBR tool".

The main research question we address here can be formulated as: what methods and tools have to be developed to support knowledge acquisition in the learning system "user - CBR tool". This issue raises the question of the knowledge of the reasoning process and leads to an analysis at the knowledge level of CBR systems. Another part of the analysis aims at studying the interactions between the user and the CBR tool during the problem solving phases. These aspects are studied at several levels in the different contributions presented in this thesis.

Our different experiences and experiments lead us to propose, as a first contribution, a formalisation at general level of interactive knowledge learning in CBR (FIKA). This formalisation relies on the reasoning failures which, as they allow to highlight the gaps in the available knowledge, are used to guide the learning process. Two extensions of this general model have been proposed: IAKA and FRAKAS.

IAKA refines the principles proposed in FIKA to permit their immediate implementation in a particular type of system where knowledge can be represented according to a given model (cases and adaptation knowledge in the form of adaptation operators). These principles have been implemented and experimented with in an application developed exclusively for this purpose. FRAKAS proposes similar methods and tools for another type of system where domain knowledge is used to guide adaptation. As for these principles, they have been implemented in a prototype inspired by a real world application.

We have conducted a study of strengths and limits of FRAKAS and IAKA and we have investigated possible ways to combine them. A first practical implementation has been made in a CBR application allowing the adaptation of cooking recipes, the project TAAABLE.